Formalising CW complexes in Lean

Hannah Scholz

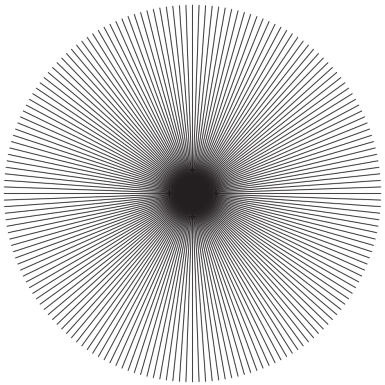
University of Bonn

23.06.2026

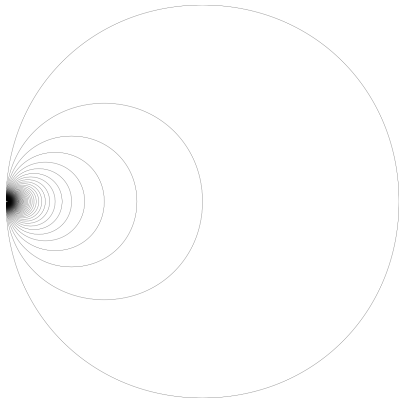
Joint work with and supervised by Prof. Floris van Doorn

Why CW complexes?

- Very general class of spaces
  - Examples of CW complexes: $\mathbb{R}^n$, S^n, $\mathbb{CP}^n$, $\mathbb{RP}^\infty$
  - Homotopy type of CW complexes: differentiable manifolds
  - Not a CW complex: hedgehog space
  - Not homotopy equivalent to a CW complex: Hawaiian earring



hedgehog space



Hawaiian earring

Why CW complexes?

- A lot of strong results about CW complexes

Theorem (Whitehead theorem, 1949)

A continuous map between two CW complexes that induces isomorphisms on all homotopy groups is a homotopy equivalence.

Theorem (Cellular homology)

Let X be a CW complex. Then the cellular and singular homology of X agree.

Intuition: What is a CW complex?

- Glue n-cells (i.e. continuous images of n-discs) together along their boundaries

Intuition: What is a CW complex?

- Glue n-cells (i.e. continuous images of n-discs) together along their boundaries

0-cells

Intuition: What is a CW complex?

- Glue n-cells (i.e. continuous images of n-discs) together along their boundaries

0-cells

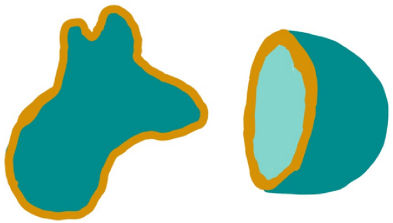
1-cells

Intuition: What is a CW complex?

- Glue n-cells (i.e. continuous images of n-discs) together along their boundaries

0-cells

1-cells



2-cells

Examples: What is a CW complex?

Interval

Examples: What is a CW complex?

Interval

Real line

Examples: What is a CW complex?

Interval

Real line

2-sphere

Definition: What is a CW complex?

Let X be a Hausdorff space. An *(absolute) CW complex* on X consists of a family of indexing sets $(I_n)_{n\in\mathbb{N}}$ and a family of continuous maps $(Q_i^n \colon D^n \to X)_{n\in\mathbb{N}, i\in I_n}$ called *characteristic maps* with the following properties:

(i) $Q_i^n|_{\mathrm{int}(D^n)} : \mathrm{int}(D^n) \to Q_i^n(\mathrm{int}(D^n))$ is a homeomorphism for every $n \in \mathbb{N}$ and $i \in I_n$. We call $e_i^n := Q_i^n(\mathrm{int}(D^n))$ an *(open) n-cell* and $\overline{e}_i^n := Q_i^n(D^n)$ a *closed n-cell*.

(ii) Two different open cells are disjoint.

(iii) For each $n \in \mathbb{N}$ and $i \in I_n$ the *cell frontier* $\partial e_i^n := Q_i^n(\partial D^n)$ is contained in the union of a finite number of closed cells of a lower dimension.

(iv) A set $A \subseteq X$ is closed if the intersections $A \cap \overline{e}_i^n$ are closed for all $n \in \mathbb{N}$ and $i \in I_n$.

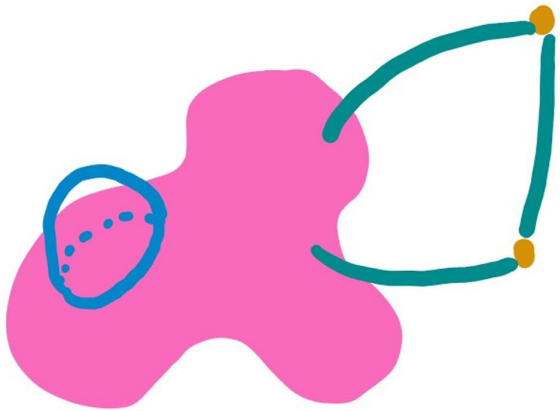
(v) The union of all closed cells is X.

Lean: What is a CW complex?

```
class CWComplex.{u} {X : Type u} [TopologicalSpace X] (C : Set X) where
  cell (n : ℕ) : Type u
  map (n : ℕ) (i : cell n) : PartialEquiv (Fin n → ℝ) X
  source_eq (n : ℕ) (i : cell n) : (map n i).source = ball 0 1
  continuousOn (n : ℕ) (i : cell n) : ContinuousOn (map n i) (closedBall 0 1)
  continuousOn_symm (n : ℕ) (i : cell n) : ContinuousOn (map n i).symm (map n i).target
  pairwiseDisjoint' :
    (univ : Set (Σ n, cell n)).PairwiseDisjoint (fun ni ↦ map ni.1 ni.2 '' ball 0 1)
  mapsTo' (n : ℕ) (i : cell n) : ∃ I : Π m, Finset (cell m),
    MapsTo (map n i) (sphere 0 1) (⋃ (m < n) (j ∈ I m), map m j '' closedBall 0 1)
  closed' (A : Set X) (hAC : A ⊆ C) :
    (∀ n j, IsClosed (A ∩ map n j '' closedBall 0 1)) → IsClosed A
  union' : ⋃ (n : ℕ) (j : cell n), map n j '' closedBall 0 1 = C
```

Intuition: What is a relative CW complex?

A relative CW complex additionally has a base set that the boundaries can attach to.



An example of a relative CW complex

Lean: What is a relative CW complex?

```
class RelCWComplex.{u} {X : Type u} [TopologicalSpace X] (C : Set X) (D : outParam (Set X)) where
  cell (n : ℕ) : Type u
  map (n : ℕ) (i : cell n) : PartialEquiv (Fin n → ℝ) X
  source_eq (n : ℕ) (i : cell n) : (map n i).source = ball 0 1
  continuousOn (n : ℕ) (i : cell n) : ContinuousOn (map n i) (closedBall 0 1)
  continuousOn_symm (n : ℕ) (i : cell n) : ContinuousOn (map n i).symm (map n i).target
  pairwiseDisjoint' :
    (univ : Set (Σ n, cell n)).PairwiseDisjoint (fun ni ↦ map ni.1 ni.2 '' ball 0 1)
  disjointBase' (n : ℕ) (i : cell n) : Disjoint (map n i '' ball 0 1) D
  mapsTo (n : ℕ) (i : cell n) : ∃ I : Π m, Finset (cell m),
    MapsTo (map n i) (sphere 0 1) (D ∪ ⋃ (m < n) (j ∈ I m), map m j '' closedBall 0 1)
  closed' (A : Set X) (hAC : A ⊆ C) :
    ((∀ n j, IsClosed (A ∩ map n j '' closedBall 0 1)) ∧ IsClosed (A ∩ D)) → IsClosed A
  isClosedBase : IsClosed D
  union' : D ∪ ⋃ (n : ℕ) (j : cell n), map n j '' closedBall 0 1 = C
```

Lean: outParam

- *Typeclass inference* has input and output parameters

Lean: outParam

- *Typeclass inference* has input and output parameters
- It doesn't run until all input parameters are provided

Lean: outParam

- *Typeclass inference* has input and output parameters
- It doesn't run until all input parameters are provided
- By default all parameters are input parameters

Lean: outParam

- *Typeclass inference* has input and output parameters
- It doesn't run until all input parameters are provided
- By default all parameters are input parameters
- `outParam` declares a parameter to be an output parameter

Lean: outParam

- *Typeclass inference* has input and output parameters
- It doesn't run until all input parameters are provided
- By default all parameters are input parameters
- `outParam` declares a parameter to be an output parameter
- output parameters cannot be provided manually

Lean: outParam

- *Typeclass inference* has input and output parameters
- It doesn't run until all input parameters are provided
- By default all parameters are input parameters
- `outParam` declares a parameter to be an output parameter
- output parameters cannot be provided manually
- typeclass inference picks the first instance where all input parameters match regardless of output parameters

Lean: outParam

- Example: `Membership α (Set α)`

Lean: outParam

- Example: `Membership α (Set α)`
- Summary: You should only use `outParam` if for every combination of the input parameters there is at most one instance

Implementation: general situation

Situation: We have a general and a specific definition where

- the specific definition is a lot more commonly used
- the specific case provides significant simplifications
- the differentiating parameter is an `outParam`

Implementation: general situation

Situation: We have a general and a specific definition where

- the specific definition is a lot more commonly used
- the specific case provides significant simplifications
- the differentiating parameter is an `outParam`

Aims:

- make using the specific definition as comfortable as possible
- make use of typeclass inference as much as possible

Implementation: Issues with naive definition

- Naive approach: define an absolute CW complex as a relative one with empty base
- Issues with naive approach:
  - repeated simplifications
  - instances where the base is provably but not definitionally equal to empty set

Implementation: Issues with naive definition

- Product of two relative CW complexes $(C, \varnothing)$ and $(E, \varnothing)$ has type:

```
RelCWComplex (C ×ˢ E) (∅ ×ˢ E ∪ C ×ˢ ∅)
```

- Product of two absolute CW complexes C and E has type:

```
CWComplex (C ×ˢ E)
```

- With the naive approach this would be definitionally the same as:

```
RelCWComplex (C ×ˢ E) ∅
```

Implementation: Solution

- `CWComplex` and `RelCWComplex` are separate classes
- there is an instance from `CWComplex` to `RelCWComplex`
- there is a definition in the other direction

General Situation

Situation:

- a lot of lemmas about relative CW complexes have simpler versions for absolute CW complexes

Typical approach:

- two namespaces `RelCWComplex` and `CWComplex`
- simplified versions live the `CWComplex` namespace and get modified names
- rest of the lemmas lives exlusively in the `RelCWComplex` namespace
- both namespaces are opened at the same time for absolute CW complexes

Consequences of typical approach

Consequences:

- lemmas for absolute CW complexes have worse names
- it is harder to search for lemmas about absolute CW complexes

This violates my aim to make absolute CW complexes as convenient as possible.

Consequences of typical approach

Consequences:

- lemmas for absolute CW complexes have worse names
- *it is harder to search for lemmas about absolute CW complexes*

This violates my aim to make absolute CW complexes as convenient as possible.

Our current approach

- allow naming lemmas for absolute CW complexes identically to their counterparts for relative CW complexes
- never open the two namespaces at the same time
- copy over all relevant declaration from the `RelCWcomplex` namespace to the `CWComplex` namespace

Lean: copying between namespaces

Two options:

- `export`: makes name available in different namespace, resolves to orginal name
- `alias`: creates a new declaration with the provided name, which can just include a change of namespace

Implementaion: Copying between namespaces

- we use `alias` for most declaration for discoverability

Implementaion: Copying between namespaces

- we use `alias` for most declaration for discoverability
- however, an `alias` doesn't change any part of the declaration (e.g. `RelCWComplex.cell` $\longrightarrow$ `CWComplex.cell`)

Implementaion: Copying between namespaces

- we use `alias` for most declaration for discoverability
- however, an `alias` doesn't change any part of the declaration (e.g. `RelCWComplex.cell` $\longrightarrow$ `CWComplex.cell`)
- thus, definitions should not be defined in `CWComplex` and instead copied over by `export` and used through the instance
- where a `CWComplex` definition is necessary (e.g. `CWComplex.cell`), this should be marked as `protected` as to not conflict with the `exported` name

Example: Copying between namespaces

```
class CWComplex.{u} {X : Type u} [TopologicalSpace X] (C : Set X) where
  protected cell (n : ℕ) : Type u
  protected map (n : ℕ) (i : cell n) : PartialEquiv (Fin n → ℝ) X
  protected source_eq (n : ℕ) (i : cell n) : (map n i).source = ball 0 1
  protected continuousOn (n : ℕ) (i : cell n) : ContinuousOn (map n i) (closedBall 0 1)
  protected continuousOn_symm (n : ℕ) (i : cell n) : ContinuousOn (map n i).symm (map n i).target
  protected pairwiseDisjoint' :
    (univ : Set (Σ n, cell n)).PairwiseDisjoint (fun ni ↦ map ni.1 ni.2 '' ball 0 1)
  protected mapsTo' (n : ℕ) (i : cell n) : ∃ I : Π m, Finset (cell m),
    MapsTo (map n i) (sphere 0 1) (⋃ (m < n) (j ∈ I m), map m j '' closedBall 0 1)
  protected closed' (A : Set X) (hAC : A ⊆ C) :
    (∀ n j, IsClosed (A ∩ map n j '' closedBall 0 1)) → IsClosed A
  protected union' : ⋃ (n : ℕ) (j : cell n), map n j '' closedBall 0 1 = C
```

Example: Copying between namespaces

```
namespace CWComplex

export RelCWComplex (cell map source_eq continuousOn continuousOn_symm isClosedBase openCell
  closedCell cellFrontier)

end CWComplex
```

```
@[alias_in CWComplex]
lemma RelCWComplex.openCell_subset_closedCell [RelCWComplex C D] (n : ℕ) (i : cell C n) :
    openCell n i ⊆ closedCell n i := image_mono Metric.ball_subset_closedBall

@[alias_in CWComplex]
lemma RelCWComplex.cellFrontier_subset_closedCell [RelCWComplex C D] (n : ℕ) (i : cell C n) :
    cellFrontier n i ⊆ closedCell n i := image_mono Metric.sphere_subset_closedBall
```

What has been done in Lean?

By other people (that I am aware of):

- Categorical definition `TopCat.RelativeCWComplex` by Jiazhen Xia and Elliot Dean Young and refactored by Joël Riou: in Mathlib
- Whitehead theorem in model categories by Joël Riou: in Mathlib
- Equivalence of the definitions by Robert Maxton: PRs
- Cellular Approximation Theorem by Mara Silge: in progress

What has been done in Lean?

By us:

- Definition and basic properties ($\sim$ 600 LOC): in Mathlib
- Finiteness notions ($\sim$ 300 LOC): in Mathlib
- Subcomplexes ($\sim$ 800 LOC): in Mathlib/PRs
- Compactly coherent spaces ($\sim$ 200 LOC): in Mathlib/PRs
- Product ($\sim$ 600 LOC): done
- Examples ($\sim$ 1000 LOC): almost done
- Rest of the Project ($\sim$ 3000 LOC)

Summary

- CW complexes are an important class of topological spaces

Summary

- CW complexes are an important class of topological spaces
- a CW complex is made up of a lot of discs glued together

Summary

- CW complexes are an important class of topological spaces
- a CW complex is made up of a lot of discs glued together
- the use of `outParam` should be limited to situations where there is exactly one instance for every combination of input parameters