Blah Blah

Blah Blah Blah
University of Bonn

6th June, 2025

Contents

1 Introduction **2**

2 Preliminaries **2**

2.1 Lean and `Mathlib` 2

2.2 Preliminary Mathematics in Lean 3

3 Definition of CW complexes **3**

4 Constructions **6**

5 Examples **6**

6 Conclusion **6**

1 Introduction

This is the introduction. It could explain the following:

- Mathematical relevance of CW complexes
- What is Lean in mathlib and why is it relevant
- Any related work (do CW complexes exist in other proof assistants?)

I think Floris was maybe going to write this?

2 Preliminaries

2.1 Lean and `Mathlib`

I could also do a subsection here explaining what Lean and mathlib is.

Talk about typeclass inference (cite paper), describe outParam, example (one from topology?) CharP

Lean and `Mathlib` make use of *typeclasses* to provide definitions on various types with potentially different behaviour. This is called *ad-hoc polymorphism*. For example, this means that there can be a general notion of a topological space on an arbitrary type. One can then provide specific *instances* of a typeclass, for example the metric topology on the reals, or assume that an instance exists for a declaration, for example assuming a topology on a type `X` by writing `[TopologicalSpace X]`. Instances can then be combined into new ones: `Mathlib` has an instance providing the subspace topology on a subtype of a type with a `TopologicalSpace` instance and for two types with each a topology there is an instance providing the product topology on the product of the types. Additionally, instances can be used to construct another instance of a different typeclass: `Mathlib` for example provides an instance expressing that a uniform space is a topological space. This creates a complicated graph of typeclass instances that can then be traversed by a search algorithm to automatically infer instances. This *typeclass inference* ensures, for example, that when a type is assumed to be a metric space, lemmas about topological spaces can be used even though `Mathlib` does not have a declaration stating explicitly that a metric space has an associated topology. Instead, this fact is a combination of different instances that are automatically combined. More about typeclasses in Lean can be found in [SUM20].

This isn't really a good example, should I do Addition instead?

This feels way too long.

There are different ways to modify the behaviour of typeclass inference for specific typeclasses. One such way, which we make use of in the definition of CW complexes, is ascribing the property `outParam` to a parameter of the typeclass. An instance of a typeclass depends on its input parameters: for `TopologicalSpace X` there is one parameter `X`; for `Membership α γ`, a typeclass stating that objects of type α can be considered as elements of objects of type γ, there are the two parameters α and γ. Looking for an instance when some of the input parameters are not known would mean significant slowdowns and potentially unwanted results of the search which is why Leans typeclass inference requires all input parameters to run. But sometimes this behaviour is not desired. As the typical examples for the membership instance include things like `Membership α (List α)` and `Membership α (Set α)`, it makes sense to assume that

2.2 Preliminary Mathematics in Lean

There aren't really a lot of basics that I need. I could write how to get a topological space, continuous maps, open sets... Not sure if I need anything else.

add PartialEquiv

In `Mathlib`, a topological space is a type `X` together with a topology `TopologicalSpace X` on it. This then allows you to describe whether a set `A : Set X` is open or closed by writing `IsOpen A` and `IsClosed A`. A function `f : X → Y` between two topological spaces `X` and `Y` can be described as being continuous and as being continuous on a set `A : Set X` which is expressed by writing `Continuous f` and `ContinuousOn f A`. `Mathlib` also implements various separation axioms: to specify that a topological space `X` is Hausdorff one can write `T2Space X`.

I am not sure if this all isn't a bit too basic and whether I should even include how to write things in Lean

3 Definition of CW complexes

In this section I should first describe mathematically what a CW complex is. Then I show the Lean definition and explain why/how Maybe this can also have proofs of basic lemmas (e.g. a CW complex is closed in its ambient space)

A *(absolute) CW complex* is a topological space that can be constructed by glueing images of closed discs of different dimensions together along the images of their boundaries. These images of closed discs in the CW complex are called *cells*. To specify that a cell is the image of an n-dimensional disc, one can call it an n-cell. The cells up to dimension n make up what is called the *n-skeleton*. In a relative CW complex these discs can additionally be attached to a specified base set.

The different definitions of CW complexes present in the literature can be broadly categorized into to approaches: firstly there is the "classical" approach that sticks closely in style to Whitehead's original definition in [Whi18]. This definition assumes the cells to all lie in one topological space and then describes how the cells interact with each other and the space. Secondly, there is a popular approach that is more categorical in nature. In this approach the skeletons are regarded as different spaces and the definition describes how to construct the $(n+1)$-skeleton from the n-skeleton. The CW complex is then defined as the colimit of the skeletons.

At the start of this project neither of the approaches had been formalized in Lean. The authors chose to proceed with the former approach because it avoids having to deal with different topological spaces and inclusions between them. At the time of writing both approaches have been formalized and are part of `Mathlib`.

make it clear that the other one was done by someone else

The definition chosen for formalization is the following:

Definition 1. Let X be a Hausdorff space and $D \subseteq X$ be a subset of X. A (relative) CW complex on X consists of a family of indexing sets $(I_n)_{n\in\mathbb{N}}$ and a family of continuous maps $(Q_i^n \colon D_i^n \to X)_{n\in\mathbb{N}, i\in I_n}$ called *characteristic maps* with the following properties:

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

n-cell.

(ii) Two different open cells are disjoint.

(iii) Every open cell is disjoint with D.

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

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

(vi) D is closed.

(vii) The union of D and all closed cells is X.

It is important to notice that an open cell is not necessarily open and that the cell frontier is not necessarily the frontier of the corresponding cell.

In Mathlib this definition translates to the following:

do this as a figure maybe?

```
  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
```

I feel like the code is a bit much at once

One obvious change in the Lean definition is that instead of talking about the topological space X being a CW complex, it talks about the set C being a CW complex in the ambient space X. This eases working with constructions and examples of CW complexes. For constructions it allows you to avoid dealing with constructed topologies, for example the disjoint union topology, and for examples it allows you to use the possibly nicer topology of the ambient space that is often already naturally present. It is however derivable from the definition that C is closed in X. So while a closed interval in the real line can be considered as a CW complex in its natural ambient space, the open interval cannot and needs to be considered as a CW complex in itself. This approach is inspired by [Gon+13], where the authors notice that it is helpful to consider subsets of an ambient group to avoid having to work with different group operations and similar issues.

Even though the behaviour of a CW complex depends strongly on its data and there can be different "non-equivalent" CW structures on the same space, we have chosen to make it a `class`, effectively treating it more like a property than a structure. This is to be able to make use of Lean's typeclass inference.

cite some paper on typeclass inference, in the preliminaries secction

We omit the requirement for X to be a Hausdorff space and instead naturally require it for most of the lemmata.

Is this too basic?

The base D is an `outParam` which means that Lean will look for a defined instance of a CW complex even if just C and not D is provided. Because of this option Lean automatically assumes that there is only one possible base for the specified set C and only searches for the first instance that matches C even if the base set does not match.

Terrible explanation. Make this better

Reason: The lemmas mainly refer to C

While we do not expect there to be instances on the same set with a different base, we have encountered instances where the base is provably equal but not definitionally so. Lean's type theory only treats definitionally equal instances as truly equal which means that typeclass inference can fail in certain situations and the instances need to be provided manually.

Firstly: I can't really provide an example right now as I have neither absolute CW complexes or products at this point. Secondly: This sounds a bit pessemistic. How much should one talk about unsolved problems?

In topology most CW complexes that are considered have empty base and often the term "CW complex" refers to this type of complex. Those CW complexes are called absolute CW complexes.

The obvious way to define absolute CW complexes in Lean would be to extend the definition of a relative CW complex by the fact that the base D is empty. However, this leads to two issues: firstly, when defining a CW complex there are now trivial proofs that need to be provided and some simplifications that need to be performed for every new absolute CW complex that is defined. This produces a lot of duplicate code. Secondly, with absolute CW complexes one does encountered instances on the same set with probably but not definitionally equal base sets more often. Constructions for relative CW complexes, e.g. products, produce instances for which in the case of an absolute CW complex the base is provably equal to the empty set but not definitionally so. In that case we define an instance specifically for absolute CW complexes and want this to be inferred over the relative version. But since D is an `outParam`, we cannot specify typeclass inference to be looking for a base that is definitionally equal to the empty set.

The solution is to have absolute CW complexes be their own definitions that agrees with relative CW complexes except for the trivial proofs and simplifications. The type of absolute CW complexes on the set C in Lean in `CWComplex C`.

talk about the instance from CWComplex to RelCWComplex here.

talk about the definition of openCell, ... here and how they all go through RelCW-Complex

talk about cellular maps here

talk about finiteness notions here, subsection? next section?

4 Constructions

Talk about:

(i) subcomplexes

(ii) attaching cells (briefly)

(iii) disjoint unions? (briefly)

(iv) transporting along partial homeomorphisms? (briefly maybe)

(v) product (detailed mathematical proof)

5 Examples

Should I talk about examples? I think the spheres would be nice. But the code is far from being polished...

6 Conclusion

Write what an impact this has made (?). Describe further possible research.

References

[Gon+13] Georges Gonthier et al. “A Machine-Checked Proof of the Odd Order Theorem”. In: *Interactive Theorem Proving*. Ed. by Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 163–179. ISBN: 978-3-642-39634-2.

[SUM20] Daniel Selsam, Sebastian Ullrich, and Leonardo de Moura. *Tabled Typeclass Resolution*. 2020. arXiv: `2001.04301 [cs.PL]`. URL: `https://arxiv.org/abs/2001.04301`.

[Whi18] J. H. C. Whitehead. “Combinatorial homotopy. I”. In: *Bulletin (new series) of the American Mathematical Society* 55.3 (2018), pp. 213–245. ISSN: 0273-0979.