Formalising the Homotopy Extension Property in Lean

Mara Masike Silge

Born on 24. April 2003 in Werne an der Lippe

September 2, 2026

Bachelor's Thesis Mathematics

Advisor: Prof. Dr. Floris van Doorn

Second Advisor: Prof. Dr. Markus Hausmann

Mathematisches Institut

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Contents

1 Introduction

This thesis is about the formalisation of the homotopy extension property (HEP) in the formal language of the proof assistant Lean. We work towards the result that every relative CW-complex satisfies the HEP. A proof assistant is a software system that checks every single step of a mathematical proof, so that an argument is accepted only once it has been justified completely. The process of writing a mathematical proof in the formal language of a proof assistant is called formalisation. Lean comes with a community-driven mathematical library called Mathlib [The20], which allows the user to build on previously formalised results. Relative CW-complexes were contributed to Mathlib by Hannah Scholz [Sch24] as part of her bachelor's thesis in 2024, which was likewise supervised by Floris van Doorn. The homotopy extension property, however, is not yet a part of it. Mathlib contains the class of cofibrations `HomotopicalAlgebra.CategoryWithCofibrations`, which is a closely related concept, but no formalisation of the HEP that would allow us to work with it explicitly and to prove it for concrete pairs of spaces. Providing such a formalisation is the subject of this thesis.

Motivation

The motivation for this project has two sides: why the result is mathematically interesting, and what is gained by formalising it. Our mathematical motivation for studying the HEP is that it is needed for two major theorems in algebraic topology. The first is the *cellular approximation theorem*.

Theorem 1.1 (Cellular Approximation Theorem, [Hat02, Theorem 4.8])**.** *Every map $f\colon X \to Y$ of CW-complexes is homotopic to a cellular map. If f is already cellular on a subcomplex $A \subseteq X$, the homotopy may be taken to be stationary on A.*

Here a map is called *cellular* if it sends the n-skeleton of X to the n-skeleton of Y for all $n \in \mathbb{N}$. An easy but important corollary of the cellular approximation theorem is the following. Equipping S^k with the CW structure consisting of one 0-cell and one k-cell, it has no cells in dimensions $0 < n < k$. Hence any map $S^n \to S^k$ with $n < k$ is homotopic to one landing in the n-skeleton, which is a single point, and is therefore null-homotopic. We see that $\pi_n(S^k) = 0$ for all $n < k$. Secondly, the result that relative CW complexes satisfy the HEP is one of the main steps towards the proof of *Whitehead's theorem*.

Theorem 1.2 (Whitehead's Theorem, [Hat02, Theorem 4.5])**.** *If a map $f\colon X \to Y$ between connected CW-complexes induces isomorphisms $f_*\colon \pi_n(X) \to \pi_n(Y)$ for all n, then f is a homotopy equivalence. In case f is the inclusion of a subcomplex $X \hookrightarrow Y$, the conclusion is stronger: X is a deformation retract of Y.*

A map between topological spaces is called a weak homotopy equivalence if it induces isomorphisms on all homotopy groups. Showing that two spaces are homotopy equivalent is in general a difficult task: it requires maps in both directions together with homotopies witnessing that both composites are homotopic to the identity, and outside of easy examples such data is hard to produce explicitly. For CW-complexes, Whitehead's theorem says that it is enough to find a map which induces isomorphisms on all homotopy groups, which is a much easier task, and hence provides a powerful tool for such problems.

Formalising a proof has several benefits, and they carry different weight here. The first is verification: a proof assistant accepts an argument only once every step has been carried out, so that no case can be overlooked and no hypothesis can be used without having been stated. For the results formalised here this aspect is secondary: the HEP is a well understood concept, and the correctness of the statements was never in doubt. The second benefit, and the one that motivates this project, is reusability. A formalised statement is available to everyone who works with the library, and anyone who uses it may rely on its proof being free of gaps. Formalising well understood material therefore serves the results that build on it. Eventually these may include results whose correctness is in doubt.

Outline

Section 2 gives an overview of the mathematics behind the formalisation. We give the definition of the homotopy extension property in several equivalent phrasings and prove the retraction criterion, which is the tool we use to show that a pair of topological spaces satisfies the property, together with two results on its preservation. Afterwards we turn to the instance we are interested in, namely that relative CW-complexes satisfy the homotopy extension property.

Section 3 is about Lean and Mathlib. Apart from a short general overview, we highlight three concepts that are relevant for this project, namely the difference between sets and subtypes, structures, and junk values. We briefly introduce Mathlib itself and the part of it concerning relative CW-complexes, which our own formalisation builds on.

The formalisation is the subject of Section 4 and makes up the main part of this thesis. Based on short excerpts of the Lean code, we explain and assess the decisions made along the way. We formalise the results of section 2 up to the homotopy extension property for finite dimensional relative CW-complexes, but not the general case. Section 5 documents how AI was used for this project, and Section 6 summarises what was achieved and indicates where the project could be continued.

The complete Lean code accompanying this thesis can be found at `https://github.com/marasilge/Bachelorarbeit`.

2 Mathematics of the HEP

Throughout this work, unless specified otherwise, every space carries a topology and every map is continuous. We write $I := [0, 1]$ for the unit interval.

The overall structure of the formalisation follows the lecture notes of the course Topology I, taught by Markus Hausmann at the University of Bonn in the winter term 2025/26. These notes are not publicly available. Since the chapter on the homotopy extension property closely follows the (German) lecture notes Algebraische Topologie by F. Waldhausen (Bielefeld) [Wal] from page 38 on, we cite this instead wherever a specific argument is referenced.

2.1 Homotopy Extension Property

There are several equivalent ways to define the homotopy extension property, and we begin this section with the definition that is closest to the one we formalised in Lean.

Definition 2.1 (cf. [Wal, Bemerkung 1])**.** A pair of topological spaces (X, A) with $A \subseteq X$ has the *homotopy extension property (HEP)* if for every topological space Y, every map $f\colon X \to Y$, and every homotopy $H\colon A \times I \to Y$ satisfying $H(a, 0) = f(a)$ for all $a \in A$, there exists a homotopy $H'\colon X \times I \to Y$ such that

$$\begin{aligned} H'(x, 0) &= f(x) && \text{for all } x \in X, \\ H'(a, t) &= H(a, t) && \text{for all } (a, t) \in A \times I. \end{aligned}$$

Remark 2.2 (cf. [Str11, Definition 5.1.1])*.* An inclusion $A \hookrightarrow X$ is called a *cofibration* if the pair (X, A) has the HEP. Cofibration and HEP are thus two names for the same notion, and the results of this section can be read in either language: Proposition 2.11 says that a composite of cofibrations is again a cofibration, and Proposition 2.9 that cofibrations are preserved under homeomorphisms of pairs.

The two extreme cases are immediate, but we record them separately because we will use one of them later as the base case of an induction.

Example 2.3. For every topological space X, both (X, X) and $(X, \emptyset)$ have the HEP.

Proof. For (X, X), the homotopy H is already defined on all of $X \times I$ and satisfies $H(x, 0) = f(x)$, so we may take $H' \coloneqq H$. For $(X, \emptyset)$, the second condition of Definition 2.1 is vacuous, so the constant homotopy $H'(x, t) \coloneqq f(x)$ has the required properties. □

Remark 2.4 (cf. [Wal, Bemerkung 2 und 3]). The HEP can equivalently be phrased as follows: (X, A) has the HEP if and only if every map defined on the pushout $X \cup_A (A \times I)$ extends to a map on $X \times I$. Here the pushout is taken along the inclusion $A \hookrightarrow X$ and the map $A \to A \times I, a \mapsto (a, 0)$. This is, in general *not* the same as requiring that every map defined on the subspace $(X \times \{0\}) \cup (A \times I) \subseteq X \times I$ extends to $X \times I$, because the canonical map

$$X \cup_A (A \times I) \to (X \times \{0\}) \cup (A \times I)$$

need not be a homeomorphism. In general, it is only a continuous bijection. If A is closed in X, however, both $X \times \{0\}$ and $A \times I$ are closed in $X \times I$, the canonical map is a closed map and hence a homeomorphism, and the two formulations agree. This is why the closedness assumption appears in exactly one direction of Lemma 2.5 below.

Lemma 2.5. *Let A be a subset of a topological space X and consider the following two statements*

(i) The pair (X, A) has the HEP.

(ii) For every topological space Y and every continuous map $g\colon (X \times \{0\}) \cup (A \times I) \to Y$ there exists an extension of g to a map $G\colon X \times I \to Y$.

Then (i) implies (ii). If also A is closed in X, then also (ii) implies (i).

Proof. (i) $\Rightarrow$ (ii). Let $g\colon (X \times \{0\}) \cup (A \times I) \to Y$ be a map. Define

$$f\colon X \to Y,\ x \mapsto g(x, 0) \qquad \text{and} \qquad H\colon (A \times I) \to Y,\ (x, t) \mapsto g(x, t).$$

Both maps are continuous: H is the restriction of g to the subspace $A \times I$, and f is the restriction of g to $X \times \{0\}$, composed with the homeomorphism $X \to X \times \{0\}$, $x \mapsto (x, 0)$. They are compatible in the sense of Definition 2.1, since $H(a, 0) = g(a, 0) = f(a)$ for all $a \in A$. Applying the HEP of (X, A) yields a homotopy $H'\colon X \times I \to Y$ with $H'(x, 0) = f(x) = g(x, 0)$ and $H'|_{A \times I} = H = g|_{A \times I}$. Hence $G := H'$ restricts to the map g on all of $(X \times \{0\}) \cup (A \times I)$.
(ii) $\Rightarrow$ (i), assuming A is closed. Let $f\colon X \to Y$ and $H\colon A \times I \to Y$ be maps with $H(a, 0) = f(a)$ for all $a \in A$. We define

$$g\colon (X \times \{0\}) \cup (A \times I) \to Y, \qquad (x, t) \mapsto \begin{cases} f(x) & t = 0, \\ H(x, t) & \text{otherwise.} \end{cases}$$

The map g is well defined: the two cases overlap precisely on $A \times \{0\}$, where $H(a, 0) = f(a)$ by assumption. For continuity we use that A is closed in X: then $A \times I$ is closed in $X \times I$, and so is $X \times \{0\}$. Both are therefore closed in the subspace $(X \times \{0\}) \cup (A \times I)$, which they cover, and the restrictions of g to them are continuous. By assumption there is an extension $G\colon X \times I \to Y$ of g, and $H' := G$ satisfies both conditions of Definition 2.1 by construction. $\square$

In preparation for the retraction criterion, we first define a retraction.

Definition 2.6. Let X be a topological space and $A \subseteq X$ a subset with inclusion $\iota : A \hookrightarrow X$. A *retraction* of X onto A is a map $r\colon X \to A$ with $r \circ \iota = id_A$. If such an r exists, we call A a *retract* of X.

Lemma 2.7 (cf. [Wal, Hilfssatz, page 39])**.** *Let A be a subset of a topological space X. The following are equivalent:*

(i) For every topological space Y and every map $g\colon A \to Y$, there exists an extension $G\colon X \to Y$.

(ii) A is a retract of X.

Proof. (i) $\Rightarrow$ (ii). Apply (i) to the special case $Y := A$ and $g := id_A$. The resulting extension G restricts to the identity on A and is therefore a retraction.
(ii) $\Rightarrow$ (i). Let $g\colon A \to Y$ be a map and let $r\colon X \to A$ be a retraction. Then set $G := g \circ r$ so that for $a \in A$ we have $G(a) = g(r(a)) = g(a)$. Hence, G is a map extending g. □

Combining the two lemmas gives the *retraction criterion*, which is the tool we use in several of the following proofs.

Proposition 2.8 (Retraction Criterion, cf. [Tom08, (5.1.2 Proposition)])**.** *Let X be a topological space and $A \subseteq X$ a subspace. Consider the following two statements:*

(i) The pair (X, A) has the HEP

(ii) The subspace $(X \times \{0\}) \cup (A \times I)$ is a retract of $X \times I$

Then (i) implies (ii). If in addition A is closed in X, then (ii) also implies (i).

Proof. (i) $\Rightarrow$ (ii). By Lemma 2.5, every map defined on $(X \times \{0\}) \cup (A \times I)$ extends to $X \times I$. Applying Lemma 2.7 to the subspace $(X \times \{0\}) \cup (A \times I) \subseteq X \times I$ yields the retraction.
(ii) $\Rightarrow$ (i), assuming A is closed. By Lemma 2.7, every map defined on $(X \times \{0\}) \cup (A \times I)$ extends to $X \times I$. Since A is closed in X, the second direction of Lemma 2.5 gives us the HEP for (X, A). □

Several formulations of the retraction criterion appear in the literature, differing in the closedness assumption. The lecture notes state it for closed subspaces $A \subseteq X$ throughout, and [Tom08] follows the same structure as above, assuming closedness for the second implication only. Hatcher [Hat02, Chapter 0] states the equivalence without any closedness hypothesis, but proves the general case only in the appendix [Hat02, Appendix], following an

argument of [Str68]. The proof given in Chapter 0 itself assumes A closed. Our formulation follows the same pattern as Proposition 2.8: closedness is imposed only on the implication that uses it. For our purposes this is harmless, since in all applications needed to prove the HEP for relative CW-complexes the subspace in question is closed anyway.

The remaining results of this section are preservation properties: they allow us to build new pairs with the HEP out of known ones. The first one states that the HEP depends on the pair (X, A) only up to homeomorphism of pairs. We record this explicitly, since it is used in Corollary 2.17 to transfer the HEP from balls to cubes. The second result is the inductive step in the proof for relative CW-complexes.

Proposition 2.9. *Let (X, A) and (Y, B) be pairs of topological spaces with B closed in Y, and let $\varphi\colon X \to Y$ be a homeomorphism with $\varphi(A) = B$. If (X, A) has the HEP, then so does (Y, B).*

Proof. Applying Proposition 2.8 to (X, A) gives a retraction $r_X\colon X \times I \to (X \times \{0\}) \cup (A \times I)$. Let $pr_X\colon X \times I \to X$ and $pr_I\colon X \times I \to I$ denote the projections onto the first and second component, respectively, and define

$$\begin{aligned} r_Y\colon Y \times I &\longrightarrow (Y \times \{0\}) \cup (B \times I) \\ (y,t) &\longmapsto \Big(\varphi \circ pr_X \circ r_X\big(\varphi^{-1}(y), t\big),\ pr_I \circ r_X\big(\varphi^{-1}(y), t\big)\Big). \end{aligned}$$

We show that r_Y is a retraction. By Proposition 2.8 this yields the HEP for the pair (Y, B), since B is closed in Y. The map r_Y:

- *takes values in* $(Y \times \{0\}) \cup (B \times I)$: let $(y, t) \in Y \times I$. The point $r_X(\varphi^{-1}(y), t)$ lies in the image of r_X, which is $(X \times \{0\}) \cup (A \times I)$. Hence either its second coordinate is 0, in which case the second component of $r_Y(y, t)$ is 0 as well, or its first component lies in A, in which case the first component of $r_Y(y, t)$ lies in $\varphi(A) = B$.
- *is continuous*, since both component functions are composites of continuous maps.
- *restricts to the identity on* $(Y \times \{0\}) \cup (B \times I)$: let (y, t) be a point of this subspace. Since φ^{-1} maps Y to X and B to A, the point $(\varphi^{-1}(y), t)$ lies in $(X \times \{0\}) \cup (A \times I)$, on which r_X is the identity. Therefore $r_X(\varphi^{-1}(y), t) = (\varphi^{-1}(y), t)$, and applying φ to the first component gives $r_Y(y, t) = (y, t)$. □

Remark 2.10. The assumption that B be closed in Y is not needed for the statement itself: the HEP is preserved under homeomorphisms of pairs without any further hypotheses. It is used only in the proof given above, where closedness is required to apply the version of the retraction criterion given in Proposition 2.8. We nevertheless give this proof, since it is the one we formalised.

The following proposition states that the HEP behaves transitively for two pairs. In the language of cofibrations, this statement says that a composite of cofibrations is again a cofibration. It is stated as such in [Str11, Problem 5.2].

Proposition 2.11. *Let $C \subseteq B \subseteq X$ be topological spaces such that both (X, B) and (B, C) have the HEP. Then (X, C) has the HEP.*

Proof. Let Z be a topological space, let $f\colon X \to Z$ be a map and let $H\colon C \times I \to Z$ be a homotopy with $H(c, 0) = f(c)$ for all $c \in C$. Applying the HEP of (B, C) to the restriction $f|_B$ and to H yields a homotopy $H'\colon B \times I \to Z$ with

$$H'(b, 0) = f(b) \quad \text{for all } b \in B \qquad \text{and} \qquad H'|_{C\times I} = H.$$

In particular $H'(b, 0) = f(b)$, so we may apply the HEP of (X, B) to f and H' and obtain a homotopy $H''\colon X \times I \to Z$ with $H''(x, 0) = f(x)$ for all $x \in X$ and $H''|_{B\times I} = H'$. The first condition of Definition 2.1 is thus satisfied, and since $C \subseteq B$ we also get $H''|_{C\times I} = H'|_{C\times I} = H$. □

2.2 HEP for relative CW-complexes

To prove the HEP for relative CW-complexes, we generalise the pair of spaces in three steps, proving the HEP at each stage:

1. Every closed ball relative to its boundary has the HEP.
2. For every relative CW-complex, two consecutive skeleta form a pair with the HEP, that is, (X_{n+1}, X_n) has the HEP for all n.
3. Every relative CW-complex (X, C) has the HEP.

Before turning to the proofs we introduce the definitions and notations we use for CW-complexes. The following definition of relative CW-complexes is a modified version of the definition of CW-complexes, given in the bachelor thesis of Hannah Scholz [Sch24]. She formalised the historical definition of CW-complexes, presented by Whitehead in [Whi18], and this formalisation is now available in Mathlib as *Classical CW-Complex*. This definition provides exactly the structures our formalised proofs rely on: explicit characteristic maps and the weak topology, rather than an inductive description via attaching cells.

Definition 2.12. We write $D^n := \{x \in \mathbb{R}^n \mid \|x\| \leq 1\}$ for the n-dimensional closed ball. Its boundary $\partial D^n = S^{n-1} := \{x \in \mathbb{R}^n \mid \|x\| = 1\}$ is the $(n-1)$-dimensional sphere.

Definition 2.13 (cf. [Sch24, Definition 1.1.2.])**.** Let X be a Hausdorff space and $C \subseteq X$ a closed subspace. A *relative CW-complex* on (X, C) consists of a family of indexing sets $(I_n)_{n\in\mathbb{N}}$ together with a family of maps $(Q_i^n\colon D_i^n \to X)_{n\geq 0, i\in I_n}$ such that:

1. $Q_i^n|_{\mathring{D}_i^n} : \mathring{D}_i^n \to Q_i^n(\mathring{D}_i^n)$ is a homeomorphism. We call $e_i^n \coloneqq Q_i^n(\mathring{D}_i^n)$ an *(open) n-cell* (or a cell of dimension n) and $\overline{e}_i^n \coloneqq Q_i^n(D_i^n)$ a *closed n-cell*.

2. For all $n, m \in \mathbb{N}$, $i \in I_n$ and $j \in I_m$ with $(n, i) \neq (m, j)$ the cells e_i^n and e_j^m are disjoint.

3. For all $n \in \mathbb{N}$ and $i \in I_n$, the set $Q_i^n(\partial D_i^n)$ is contained in the union of C and finitely many closed cells of dimension less than n.

4. A subset $S \subseteq X$ is closed if and only if $S \cap C$ is closed and $S \cap \overline{e}_i^n$ is closed for all $n \in \mathbb{N}$ and $i \in I_n$.

5. $C \cup \bigcup_{n \geq 0} \bigcup_{i \in I_n} Q_i^n(D_i^n) = X$.

We call Q_i^n a *characteristic map*, refer to C as the *base* of the complex and call $\partial e_i^n \coloneqq Q_i^n(\partial D_i^n)$ the *frontier of the n-cell* for any i and n. For $-1 \leq n \leq \infty$ we define the *n-skeleton* of X as

$$X_n \coloneqq C \cup \bigcup_{m \leq n} \bigcup_{i \in I_m} \overline{e}_i^m.$$

Note that $X_{-1} = C$ and $X_\infty = X$. Moreover, every skeleton is closed in X. This follows from condition (4), since the intersection of X_n with C and with every closed cell is closed. We use this below when we apply the converse direction of the retraction criterion Proposition 2.8 for skeleta.

Definition 2.14. If $C = \emptyset$, then $(X, \emptyset)$, or simply X, is an *(absolute) CW-complex*.

Definition 2.15. A relative CW-complex (X, C) is called *finite dimensional* if there exists an $n \in \mathbb{N}$ such that $I_m = \emptyset$ for all $m \geq n$.

2.2.1 HEP for balls

We begin the HEP proof for balls.

Theorem 2.16 (cf. [Boa16, Lemma 11])**.** *For every $m \in \mathbb{N}$, the pair $(D^m, \partial D^m)$ has the HEP.*

There are two different attempts to prove this. One argues visually via the retraction criterion, projecting the cylinder $D^m \times I$ from a point above it onto $(D^m \times \{0\}) \cup (\partial D^m \times I)$. The other constructs the extended homotopy explicitly. We give the constructive proof presented in [Boa16, Lemma 11], since it was the easier one to formalise in Lean.

Proof. Let Y be a topological space, let $f\colon D^m \to Y$ be a map and let $H\colon \partial D^m \times I \to Y$ be a homotopy with $H(a,0) = f(a)$ for all $a \in \partial D^m$. We assemble these into a map $h\colon D_2^m \to Y$, where D_2^m denotes the closed ball of radius 2 in $\mathbb{R}^m$, by setting

$$h(x) = \begin{cases} f(x) & \|x\| \leq 1 \\ H\left(\dfrac{x}{\|x\|}, \|x\| - 1\right), & \|x\| \geq 1 \end{cases}$$

The two cases agree on the sphere $\|x\| = 1$, where this is exactly the compatibility assumption on f and H. Both domains are closed in D_2^m, so h is continuous by the pasting lemma. Using h we define

$$H'\colon D^m \times I \to Y, \qquad H'(x,t) = h\big((1+t)x\big),$$

which is well defined and continuous, and which satisfies $H'(x,0) = h(x) = f(x)$ for all $x \in D^m$ as well as $H'(a,t) = H(a,t)$ for all $a \in \partial D^m$ and $t \in I$, since $\|(1+t)a\| = 1 + t \geq 1$. □

In Lean the term $x/\|x\|$ can be written down without any hypothesis on x, since division by zero is defined to be zero. Here this is a convenience rather than a necessity, as the term only occurs where $\|x\| \geq 1$.
As we will discuss in Section 3.3.1, the definition of CW-complexes in Mathlib uses n-dimensional cubes rather than closed balls as the domain of the characteristic maps. We therefore state and prove the HEP for cubes relative to their boundary. It is stated without proof as [Tom08, Example 5.1.4], there for the pair $(I^n, \partial I^n)$.

Corollary 2.17. *For every $n \in \mathbb{N}$, the pair $([-1,1]^n, \partial[-1,1]^n)$ has the HEP.*

Proof. The ball D^n is homeomorphic to the cube $[-1,1]^n$ via radial scaling, which maps each ray from the origin to itself and takes ∂D^n to $\partial[-1,1]^n$. The claim now follows from Theorem 2.16 together with Proposition 2.9. □

2.2.2 HEP for consecutive skeleta

In the next step we prove that for every relative CW-complex (X, C) and every $m \in \mathbb{N}$, the pair (X_m, X_{m-1}) has the HEP. We do so by applying the retraction criterion, that is, by constructing a retraction

$$r_m^{\mathrm{Skel}}\colon X_m \times I \to (X_m \times \{0\}) \cup (X_{m-1} \times I).$$

This map is obtained from the universal property of the quotient. To use it we need to know that $X_m \times I$ carries the quotient topology, and for this it suffices to know that X_m does, by the following theorem:

Theorem 2.18 (3.3.17. The Whitehead Theorem, [Eng89])**.** *Let Z be a locally compact space and let $g\colon Y \to Y'$ be a quotient map. Then*

$$g \times id_Z \colon Y \times Z \to Y' \times Z$$

is a quotient map.

This theorem is also known as Whitehead's theorem but has nothing to do with Theorem 1.2 above. Both results are named after J. H. C. Whitehead. Since the unit interval I is compact, and in particular locally compact, Theorem 2.18 applies to $Z \coloneqq I$ throughout this section. Let us prove that every skeleton indeed is a quotient space. We set $Z_m \coloneqq C \amalg \coprod_{n \leq m} \coprod_{i \in I_n} D_i^n$ and define

$$\begin{aligned} P_m : Z_m &\to X_m \\ x &\mapsto \begin{cases} Q_i^n(x) & x \in D_i^n, \\ x, & x \in C. \end{cases} \end{aligned}$$

Lemma 2.19 (cf. [SZ88, Satz 4.1.11])**.** *Let (X, C) be a relative CW-complex. For every $m \in \mathbb{N}$ the map $P_m : Z_m \to X_m$ is a quotient map.*

Proof. Let $m \in \mathbb{N}$ be a natural number. The map P_m is well defined, since the images of the characteristic maps up to dimension m lie in X_m and the base C is contained in every skeleton. It is continuous because its restriction to each summand of the coproduct is, and it is surjective by the definition of X_m.
It remains to show that a subset $S \subseteq X_m$ is closed whenever $P_m^{-1}(S)$ is closed. So let $P_m^{-1}(S)$ be closed. By the definition of the coproduct topology this means that

$$P_m^{-1}(S) \cap C = S \cap C \qquad \text{and} \qquad P_m^{-1}(S) \cap D_i^n = (Q_i^n)^{-1}(S)$$

are closed for all $n \leq m$ and $i \in I_n$. Since X_m is closed in X, it suffices to show that S is closed in X. By condition 4 of Definition 2.13 this amounts to showing that $S \cap C$ and $S \cap \overline{e}_i^n$ are closed for all $n \in \mathbb{N}$ and $i \in I_n$.
Note that $S \cap C$ is closed by the above. For $n \leq m$ we have $S \cap \overline{e}_i^n = Q_i^n\big((Q_i^n)^{-1}(S)\big)$, which is closed as the image of a closed subset of the compact space D_i^n under a continuous map into a Hausdorff space. For $n > m$ we use a variant of the closedness characterisation, which is available in Mathlib as `Topology.RelCWComplex.isClosed_of_disjoint_openCell_or_isClosed_inter_closedCell`. By this, it suffices to consider the intersection of S with the open cells e_i^n. Since $S \subseteq X_m$ is disjoint from every open cell of dimension greater than m, these intersections are empty. □

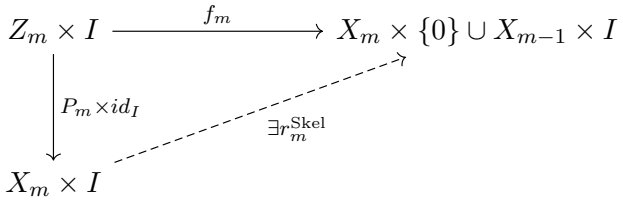
Remark 2.20. The reference given for Lemma 2.19 does not prove this exact statement, but the closely related result that a map defined on a CW-complex

is continuous if and only if its restriction to each closed cell is continuous. Moreover, we have adapted the statement to relative CW-complexes by imposing an additional condition on the restriction to the base.

By Lemma 2.19 and Theorem 2.18, the map $P_m \times id_I \colon Z_m \times I \to X_m \times I$ is a quotient map. We now define for all $m \in \mathbb{N}$

$$f_m : Z_m \times I \to (X_m \times \{0\}) \cup (X_{m-1} \times I),$$
$$(z,t) \mapsto \begin{cases} (P_m \times id_I) \circ r_i^m(z,t), & z \in D_i^m \text{ for some } i \in I_m, \\ (P_m \times id_I)(z,t) & \text{otherwise.} \end{cases}$$

Here $r_i^m \colon D_i^m \times I \to (D_i^m \times \{0\}) \cup (S_i^{m-1} \times I)$ is the retraction obtained from Theorem 2.16 and the retraction criterion Proposition 2.8. Note that the case distinction is unambiguous: the summands of Z_m are disjoint, so z lies in a ball of dimension exactly m for at most one index i. Once we know that f_m is continuous and factors through $P_m \times id_I$, the universal property of the quotient yields a map r_m^{Skel} making the following diagram commute:



It is not clear a priori that this map is a retraction. We verify this separately in Theorem 2.22.

Lemma 2.21. *For every $m \in \mathbb{N}$, the map f_m*

(i) takes values in $(X_m \times \{0\}) \cup (X_{m-1} \times I)$,

(ii) is continuous, and

(iii) factors through $P_m \times id_I$

Proof. (i). We treat the two cases of the definition separately. First let $(z,t) \in D_i^m \times I$ for some $i \in I_m$. Since r_i^m is a retraction,

$$r_i^m(z,t) \in (D_i^m \times \{0\}) \cup (S_i^{m-1} \times I).$$

If the second coordinate of $r_i^m(z,t)$ is 0, then $f_m(z,t) \in X_m \times \{0\}$. Otherwise the first coordinate lies in $S_i^{m-1} = \partial D_i^m$, and by condition (3) of Definition 2.13 its image under Q_i^m lies in X_{m-1}, so $f_m(z,t) \in X_{m-1} \times I$. Now let $(z,t) \in Z_m \times I$ be a point of the second case, so $z \in C$ or $z \in D_i^n$ for some $n < m$. In either case $P_m(z) \in X_{m-1}$, hence $f_m(z,t) \in X_{m-1} \times I$.

(ii). The space $Z_m \times I$ is canonically homeomorphic to $(C \times I) \amalg \coprod_{n \le m} \coprod_{i \in I_n} (D_i^n \times I)$, so that f_m can be regarded as a map out of a disjoint

union. On each summand it restricts to a composite of continuous maps, and a map out of a disjoint union is continuous if and only if all its restrictions to the summands are. Hence f_m is continuous.

(iii). Let (z,t) and $(z',t') \in Z_m \times I$ with $P_m \times id_I(z,t) = P_m \times id_I(z',t')$. Since $P_m \times id_I$ is the identity on the second coordinate, we directly get $t = t'$ and $P_m(z) = P_m(z')$. We must show that $f_m(z,t) = f_m(z',t)$.
If neither z nor z' lies in a ball of dimension m, both are in the second case of the definition and $f_m(z,t) = (P_m \times id_I)(z,t) = (P_m \times id_I)(z',t) = f_m(z',t)$. The same conclusion holds if z or z' lies in ∂D_i^m for some $i \in I_m$ since r_i^m, being a retraction, is the identity on the boundary and f_m again agrees with $P_m \times id_I$. So we may assume that $z \in \mathring{D}_i^m$ for some $i \in I_m$, and it suffices to show $z = z'$.
We have $P_m(z) = Q_i^m(z) \in e_i^m$. Suppose z' did not lie in $\mathring{D}_i^m$. If $z' \in \mathring{D}_j^m$ for some $j \neq i$, then $P_m(z') \in e_j^m$, which is disjoint from e_i^m by condition (2) of Definition 2.13. In all remaining cases $z' \in C$, or z' lies in a ball of dimension less than m, or in the boundary of a ball of dimension m. In each of these cases, condition (3) gives $P_m(z') \in X_{m-1}$, which is disjoint from e_i^m as well. Both contradict $P_m(z) = P_m(z')$, so $z' \in \mathring{D}_i^m$. Since $Q_i^m|_{\mathring{D}_i^m}$ is a homeomorphism onto e_i^m and hence is injective, the fact that $P_m(z) = P_m(z')$ forces $z = z'$. □

The idea of the following proof is taken from [Wal, page 40].

Theorem 2.22. *Let (X,C) be a relative CW-complex with skeleta $(X_m)_{m\in\mathbb{N}}$. Then (X_m, X_{m-1}) has the HEP for every $m \in \mathbb{N}$.*

Proof. By Lemma 2.21 and the universal property of the quotient we have a continuous map $r_m^{\mathrm{Skel}}\colon X_m \times I \to (X_m \times \{0\}) \cup (X_{m-1} \times I)$ with $r_m^{\mathrm{Skel}} \circ (P_m \times id_I) = f_m$. By Proposition 2.8 it remains to show that r_m^{Skel} restricts to the identity on $(X_m \times \{0\}) \cup (X_{m-1} \times I)$. The pair then has the HEP, since X_{m-1} is closed in X_m.
Because f_m factors through $P_m \times id_I$, the value $r_m^{\mathrm{Skel}}(x,t)$ does not depend on the choice of a preimage of (x,t), and we may therefore choose a convenient one in each case.
Points of $X_{m-1} \times I$. Let $(x,t) \in X_{m-1} \times I$. Then x lies in C or in a closed cell of dimension less than m, so we may choose a preimage (z,t) with $z \in C$ or $z \in D_i^n$ for some $n < m$. Such a z falls under the second case of the definition of f_m. Therefore

$$r_m^{\mathrm{Skel}}(x,t) = r_m^{\mathrm{Skel}}\big((P_m \times id_I)(z,t)\big) = f_m(z,t) = (P_m \times id_I)(z,t) = (x,t).$$

Points of $X_m \times \{0\}$. Let $(x,0) \in X_m \times \{0\}$. If x lies in X_{m-1}, the previous case applies. Otherwise x lies in an open cell of dimension m, so there are $i \in I_m$ and $z \in D_i^m$ with $P_m(z) = x$. Since r_i^m is a retraction and $(z,0) \in D_i^m \times \{0\}$, we get $r_i^m(z,0) = (z,0)$ and therefore

$$r_m^{\mathrm{Skel}}(x,0) = f_m(z,0) = (P_m \times id_I)\big(r_i^m(z,0)\big) = (P_m \times id_I)(z,0) = (x,0).$$ □

2.2.3 HEP for relative CW-complexes

In this final section we prove that every relative CW-complex (X, C) has the HEP. For a finite dimensional complex this follows quickly from what we established so far.

Lemma 2.23. *Let (X, C) be a relative CW-complex. Then (X_n, C) has the HEP for every $n \in \mathbb{N} \cup \{-1\}$.*

Proof. We proceed by induction on n.
Base case $n = -1$: Note that $X_{-1} = C$, so the pair in question is $(X_{-1}, C) = (C, C)$, which has the HEP by Example 2.3.
Inductive step $n \to n+1$. Assume we know that (X_n, C) has the HEP. By Theorem 2.22 the pair (X_{n+1}, X_n) has the HEP as well. Since $C \subseteq X_n \subseteq X_{n+1}$, the transitivity shown in Proposition 2.11 yields the HEP for (X_{n+1}, C). □

Corollary 2.24. *Let (X, C) be a finite dimensional relative CW-complex. Then (X, C) has the HEP.*

Proof. Since (X, C) is finite dimensional, there is an $n \in \mathbb{N} \cup \{-1\}$ such that X is equal to X_n. The claim therefore follows from Lemma 2.23 applied to X_n. □

This is the point at which our formalisation stops. For the sake of completeness we sketch the proof of the general case, which we did not formalise in Lean.

Theorem 2.25 (cf. [Wal, page 40 f.])**.** *Let (X, C) be a relative CW-complex. Then (X, C) has the HEP.*

Proof sketch. We use the retraction criterion, which applies since C is closed in X. In a first step the maps r_m are constructed inductively for all $m \in \mathbb{N} \cup \{-1\}$.

$$r_m \colon X_m \times I \to (X_m \times \{0\}) \cup (C \times I)$$

For $m = -1$ we take the identity, and for $m \geq 0$ we let r_m be the composite

$$X_m \times I \xrightarrow{r_m^{\mathrm{Skel}}} (X_m \times \{0\}) \cup (X_{m-1} \times I) \xrightarrow{id \cup r_{m-1}} (X_m \times \{0\}) \cup (C \times I),$$

where the second map is the identity on $X_m \times \{0\}$ and r_{m-1} on $X_{m-1} \times I$. A straightforward induction shows that these two prescriptions agree on the overlap and that $r_m|_{X_{m-1} \times I} = r_{m-1}$ for all m. This compatibility allows us to define

$$\begin{aligned} r \colon X \times I &\longrightarrow (X \times \{0\}) \cup (C \times I), \\ r(x,t) &\coloneqq r_m(x,t) \quad \text{for any } m \text{ with } x \in X_m, \end{aligned}$$

which is independent of the choice of m. It restricts to the identity on $(X \times \{0\}) \cup (C \times I)$ since each r_m does. Continuity follows from Proposition 2.26 below, applied to $Z \coloneqq I$. The restriction of r to $X_n \times I$ is r_n and hence continuous for every n. Thus r is a retraction, and the retraction criterion gives the HEP for (X, C). □

The following proposition is what the last step of the sketch appeals to.

Proposition 2.26. *Let (X, C) be a relative CW-complex, let Z be a locally compact space and let Y be a topological space. A map $f\colon X \times Z \to Y$ is continuous if and only if the restriction $f|_{X_n \times Z}$ is continuous for every $n \geq -1$.*

We state Proposition 2.26 without proof. It says that X carries the colimit topology of its skeleta and that this is preserved under taking the product with a locally compact space. This is the same phenomenon that Theorem 2.18 describes for a single quotient map.

3 Lean and Mathlib

Having discussed the mathematics behind our formalisation, we now give a brief overview of Lean and of the conventions and concepts that are relevant to this project.

Lean is a proof assistant, that is, a tool that checks the individual reasoning steps of a proof written in its formal language. It verifies that assumptions, axioms, functions and theorems are applied only where permitted and that new results follow by valid logical inference. According to its main designers, this verification rests upon a "small trusted kernel based on dependent type theory" [Mou+15], a notion we explain in the next section. Lean is one of several proof assistants, alongside Rocq and Isabelle. As Kontorovich describes in *The Shape of Math to Come* [Kon25], it is remarkable how quickly Lean has gained relevance not only in computer science and mathematical logic, but in mainstream mathematical research.

The first version of Lean was created by Leonardo de Moura in 2013, and the first stable release of Lean 4 appeared in September 2023. Since July 2023, the development of the language has been coordinated by the non-profit *Lean Focused Research Organization*, whose stated goal is "enhanced scalability, usability, documentation, and proof automation while guiding Lean toward long-term self-sustainability" [FRO]. Alongside the language, the mathematical library *Mathlib* has grown considerably: as of August 2026 it comprises almost 2,500,000 lines of code contributed by over 770 authors [Mat].

3.1 Lean's Dependent Type Theory

The following description is based on the chapter *Dependent Type Theory* in the digital book *Theorem Proving in Lean 4* [Avi+24].
Every expression in Lean has a type, which can be displayed with the command `#check`. The familiar mathematical objects $\mathbb{N}$, $\mathbb{R}$, $\mathbb{C}$ and the Euclidean spaces all have type `Type`, and a function from `α` to `β` has type `α → β`. Types themselves are expressions and therefore have types again: `Type` has type `Type 1`, and in general `Type k` has type `Type (k+1)`. This hierarchy of *universes* is why declarations in later sections carry universe variables such as `u`. An arbitrary universe is introduced automatically when we write `Type*` instead.

```
#check ℕ
    ℕ : Type
#check EuclideanSpace ℝ (Fin 3)
    EuclideanSpace ℝ (Fin 3) : Type
#check Type
    Type : Type 1
#check 2 = 1
    2 = 1 : Prop
```

A distinguished type is `Prop`. Lean uses the word in the sense it has in logic: a proposition is a statement that may be true or false, and `Prop` is the type of such statements. Note that `2 = 1` is a well-formed term of type `Prop`; a term of this type need not be a true statement.

What makes this a *dependent* type theory is that a type may depend not only on other types, but on arbitrary terms. We have already seen an instance of this: `Fin : ℕ → Type` assigns to every natural number n the type with n elements, so that `Fin 3` and `Fin 5` are distinct types determined by the values 3 and 5.

3.2 Relevant Concepts of Lean

Formalising in Lean involves a number of concepts and conventions. We briefly explain three of them that are particularly relevant to this project.

Sets and Subtypes

Type theory allows us not only to define elements of a type, but to build new types from existing ones. The construction most relevant to this project is that of a subtype. Given a type `α` and a predicate `p : α → Prop`, the subtype `{x : α // p x}` is the type whose terms are pairs consisting of a term `x : α` together with a proof of `p x`. Constructing such a term therefore requires supplying both components. In the example below, the zero of the closed unit ball in $\mathbb{R}$ is given by the real number `0` together with a proof that it lies in the ball. While one would not usually distinguish between this point and the real number zero, in Lean they are terms of different types.

```
#check (⟨0 , Metric.mem_closedBall_self zero_le_one⟩ : Metric.closedBall (0 :
↪ ℝ) 1)
    ⟨0, …⟩ : { x // x ∈ Metric.closedBall 0 1 }
```

Closely related is the type `Set α` of subsets of `α`, which is defined as `α → Prop`. These two notions are connected by coercions. Every `S : Set α` can be read as a type `↑S`, namely the subtype `{x : α // x ∈ S}`, and conversely every term of `↑S` has an underlying term of `α`, obtained by `Subtype.val`. Both coercions are inserted automatically by Lean where needed.
The distinction matters for us because it offers different ways of formalising a pair of spaces (X, A) with $A \subseteq X$, two of which are used in this project. In the first, the larger space is a type `X` and the subspace is a term `A : Set X`. In the second, both are terms of `Set Y` for an ambient type `Y`. In either case a subtype appears as soon as we want to treat a set as a space in its own right. Much of the design of the definitions below is aimed at avoiding subtypes and coercions.

Structures

A *structure* in Lean bundles several components into a single type. Its declaration lists a number of fields, each with its own type, and a term of the structure is obtained by supplying a term for every field. Structures are how Mathlib organises mathematical objects: a topological space, for instance, is a structure whose fields consist of a collection of open sets together with proofs of the axioms it has to satisfy. We use this mechanism in three places, in each case as a structure whose fields are all propositions. Since proofs of a proposition are indistinguishable in Lean, such a structure carries no data beyond the fact that its conditions hold. It simply gives a name to a conjunction of conditions, so that a single hypothesis can be passed around instead of several separate ones, and each condition can be retrieved by its field name.

Junk Values

In Lean, division by zero is defined to yield 0.

```
example : (1 : ℚ) / 0 = 0 := by norm_num
```

This makes division a total function $\mathbb{Q} \to \mathbb{Q} \to \mathbb{Q}$, even though certain inputs have no genuine mathematical meaning. As explained in [MLS25], such *junk values* are common practice in Lean. The same convention applies, for instance, to the inversion of square matrices, which is defined for singular matrices as well, where it returns the zero matrix, or to the derivative of a function at a point where it is not differentiable, which is defined to be zero.

At first glance this may seem concerning, since one can state and prove propositions that in classical mathematics would look false or ill-defined. The issue is resolved by adding suitable hypotheses such as `n ≠ 0` or `Differentiable 𝕜 f` to any statement about the object in question. These ensure that a theorem can only be applied where its conclusion is meaningful. What is gained in return is that definitions and statements become simpler, because no side conditions have to be carried along merely to make an expression well formed.

We make frequent use of this convention. Functions that are mathematically defined only on a subset are defined on the entire ambient space, which allows us to extend them without subtype coercions and to apply them to a point without proving each time that it lies in the right subset. We benefit from this convention in the formalisation of Theorem 2.16: the term $x/\|x\|$ in the definition of h needs no hypothesis on x, so that no side condition $\|x\| \neq 0$ has to be carried through the proof.

3.3 Mathlib

Results in Mathlib are stated as generally as the setting allows, so that they apply in as many situations as possible. For this project we import, among others, the topology packages, which include the formalisation of CW-complexes and the basic results about them that our proofs rely on.

Names in Mathlib are chosen to be self-explanatory, in the sense that the name of a result describes its statement. The theorem that a composition of two continuous functions is continuous, for example, is called `Continuous.comp`. Given `f` and `g` together with proofs `hg : Continuous g` and `hf : Continuous f`, we obtain a proof of `Continuous (g ∘ f)` by writing `Continuous.comp hg hf`, or, in the shorter and more common form using dot notation, `hg.comp hf`. This works because `hg` has type `Continuous g` and the name of the theorem begins with `Continuous`, so that Lean resolves `hg.comp` to `Continuous.comp` with `hg` inserted as the first explicit argument of that type.

3.3.1 CW-complexes in Mathlib

We introduce those aspects of Mathlib's formalisation of CW-complexes that are relevant for us. The reference for this section is the documentation of CW-complexes in Mathlib. All notions discussed below live in the namespace `Topology.RelCWComplex`, which we open in our files, so that we write `cell`, `map` or `skeletonLT` without the prefix.

First of all, a relative CW-complex is not defined as a pushout, but instead lives in an ambient type. This is what motivated the definition `HEP'`. A relative CW-complex (X, C) is declared in the following way.

```
variable {Y : Type u} [TopologicalSpace Y] (X : Set Y) {C : Set Y}
↪  [RelCWComplex X C]
```

Here the complex `X` and its base space `C` are both sets of the ambient type `Y`. This is the third of the variable blocks fixed in Section 4.1. Mathlib's definition does not assume the ambient space to be Hausdorff. Accordingly, for many statements about complexes we have to add the instance `[T2Space Y]`.

In each dimension there is an index type `cell X n`, and for each cell a characteristic map `map (n : ℕ) (i : cell X n) : PartialEquiv (Fin n → ℝ) Y`. The image of the open ball is called `openCell n i` and that of the closed ball `closedCell n i`.

The characteristic maps have type `PartialEquiv (Fin n → ℝ) Y`. A partial equivalence is a structure containing a map between the two given types, an inverse map in the other direction and two sets, called `source` in the domain and `target` in the codomain. Even though the map is defined on the entire type, it is supposed to represent a map between these two subsets. That is, the map sends the source to the target and the inverse map sends the target to the source. On the source and the target respectively, the two maps are

inverse to one another.
The characteristic maps are partial equivalences with source the open ball and target the open cell, since these are the sets between which we have a bijection. They are continuous on the closed ball, since that is where their definition has a mathematical meaning.

When saying that the source of the partial equivalence is an open ball, one has to be careful. In the definition this ball lies in the type `Fin m → ℝ`, which is a formalisation of the m-dimensional real vector space carrying the maximum norm. Hence what the definition calls a ball is what we would usually call a cube. This is why it does not suffice to prove the HEP for a ball in an m-dimensional Euclidean space `EuclideanSpace ℝ (Fin m)`. The result has to be transferred to cubes, which we do in Section 4.5.
The topology of a complex is described through its closed sets. A subset of `X` is closed if and only if its intersection with the base space and with every closed cell is closed there. Note that with this definition, the description of the complex as a quotient of the disjoint union of its base space with the closed balls becomes a statement that has to be proved, whereas for a definition by a pushout it would be the definition itself. We do exactly that in Section 4.6.1.
Skeleta are not plain sets but subcomplexes, that is, terms of a structure whose underlying set is available as `(skeletonLT X n).carrier`. There are two related notions of skeleton, which differ by one index. The set `skeletonLT X n` is the union of the base space with all cells of dimension strictly less than n, whereas `skeleton X n` also includes the cells of dimension n. The gain of working with `skeletonLT` is that in an induction the base case is about the base space itself rather than about its union with the cells of dimension zero. The results for `skeleton` are then derived from those for `skeletonLT`. This is the procedure suggested in the documentation, and the one we follow throughout this project.
Finally, a complex is `FiniteDimensional` if the index types `cell X n` are empty from some dimension on. We use this property for our final result in Section 4.7.

4 Lean formalisation of the Homotopy Extension Property

4.1 Notation

The formalisation consists of the four files `HEP_definition.lean`, `HEP_ball_cube.lean`, `HEP_skeleton.lean` and `HEP_relCW.lean`. In all of them the same letter always denotes the same kind of object, so that a statement can be read without looking up where its variables were introduced. We fix this notation once and for all here and do not repeat it at the beginning of the following sections.

`X`	the larger space of a pair (X, A), a type or a set
`A`	the subspace of the pair, $A \subseteq X$
`Y`	the ambient type in which the sets `X` and `A` live
`Z`	the target type of the maps `f`, `H`, `H'`, `g`, `G` to be extended
`rangeH'`, `rangeG`	the subset of `Z` in which these maps take their values
`C`	the base of a relative CW-complex
`r`, `s`	retractions
`m`, `n`	dimensions

A pair of spaces occurs in two shapes in this project, which is what the definitions `HEP` and `HEP'` reflect. Either the larger space is a type and the subspace a set in it, or both are sets of a common ambient type. Accordingly the file `HEP_definition.lean` is split into the two sections `PairLargeType` and `PairLargeSet` with the variable blocks below. The two files about CW-complexes use the third block, which is the declaration of a relative CW-complex discussed in Section 3.3.1.

```
universe u

-- section `PairLargeType`, the setting of `HEP`:
variable {X Z : Type u} [TopologicalSpace X] [TopologicalSpace Z] {A : Set X}

-- section `PairLargeSet`, the setting of `HEP'`:
variable {Y : Type u} [TopologicalSpace Y]

-- `HEP_skeleton.lean` and `HEP_relCW.lean`:
variable {Y : Type u} [TopologicalSpace Y] [T2Space Y] (X : Set Y) {C : Set Y}
  [RelCWComplex X C]
```

Throughout the formalisation we use the following abbreviations.

```
abbrev cyl (X : Type*) : Set (X × ℝ) := {p | p.2 ∈ unitInterval}
abbrev smallcyl {X : Type*} (A : Set X) : Set (X × ℝ) := {p | p.1 ∈ A ∧
  p.2 ∈ unitInterval}
```

```
abbrev anchor {X : Type*} (A : Set X) : Set (X × ℝ) :=
  {p | p.2 = 0 ∨ p.1 ∈ A ∧ p.2 ∈ unitInterval}
abbrev smallanchor {Y : Type*} (X A : Set Y) : Set (Y × ℝ) :=
  {p | p.1 ∈ X ∧ p.2 = 0 ∨ p.1 ∈ A ∧ p.2 ∈ unitInterval}
```

These four abbreviations name the subsets that occur in the definition of the HEP and in the retraction criterion. The set `cyl X` is the cylinder of height one over the type `X`, whereas `smallcyl A` is the cylinder only over a set `A` of type `Set X`. The set `anchor A` is the union of the cylinder over `(A : Set X)` with the bottom of the cylinder `cyl X`. Finally, `smallanchor X A` is the corresponding set for the case that `(X A : Set Y)` are subsets of an ambient type `Y`. It is the union of the cylinder over `A` with the bottom of `smallcyl X`. All four are declared as `abbrev` rather than `def` and are therefore reducible, so that elaboration and `simp` see through them.

4.2 HEP Definition

Already in the mathematical part, the definition of the homotopy extension property was long and technical, and its formalisation is no shorter. We can now turn to it.

```
def agreeOn (f : X → Z) (H : X × ℝ → Z) (A : Set X) : Prop :=
    ∀ (a : X), a ∈ A → f a = H (a,0)

structure HomotopyExtension (H' : X × ℝ → Z) (f : X → Z) (H : X × ℝ → Z)
    (A : Set X) (rangeH' : Set Z) : Prop where
  ContinuousOn : ContinuousOn H' (cyl X)
  range : (cyl X).MapsTo H' rangeH'
  agreef : (∀ (x : X), f x = H' (x, 0))
  agreeH : (∀ (a : A) (t : unitInterval), H (a,t) = H' (a, t))

def HEP (X : Type u) [TopologicalSpace X] (A : Set X) : Prop :=
  ∀ (Z : Type u) [TopologicalSpace Z] (rangeH' : Set Z), ∀ (f : X → Z),
  ∀ (H : X × ℝ → Z), Continuous f → Set.range f ⊆ rangeH' → ContinuousOn H
  (smallcyl A) → (smallcyl A).MapsTo H rangeH' → agreeOn f H A →
  ∃ H' : X × ℝ → Z, HomotopyExtension H' f H A rangeH'
```

To shorten the definition and make it more readable, we first define `agreeOn`. It expresses that the map `f` and the homotopy `H` we wish to extend take the same values on `A`, respectively `A × {0}`, so that the two maps are compatible and asking for a common extension makes sense. Secondly, the structure `HomotopyExtension` bundles the properties the desired homotopy should have: it is continuous on the cylinder, its values lie in the prescribed set, and it agrees with the two given maps. Using a structure rather than a conjunction has the advantage that it comes with projections for the individual fields, which is what makes the applications readable.

This definition of `HEP` is close to the informal Definition 2.1, apart from three aspects elaborated below. Before turning to them, and as a first check

that the definition is usable at all, we prove that every space has the HEP relative to itself and relative to the empty set. The lemma `HEP_self` will serve as the base case of the induction in Section 4.7.

```
lemma HEP_self : HEP X (@Set.univ X) := by ...

lemma HEP_empty : HEP X ∅ := by ...
```

Homotopies on `X × ℝ`

The homotopies are defined on `X × ℝ` rather than on `X × unitInterval` or even `A × unitInterval`. The reason is that a homotopy defined on `↑A × ↑unitInterval` is applied to pairs `(⟨a, ha⟩, ⟨t, ht⟩)`. Every time we want to evaluate it we have to supply a proof that the point lies in the right subset, and every equation between such terms has to be passed through `Subtype.ext` or a congruence lemma before the underlying points can be compared. On `X × ℝ` none of this bookkeeping appears.

This does not change the object being defined. The field `ContinuousOn H' (cyl X)` constrains `H'` only on the cylinder, so what `H'` does outside the unit interval is irrelevant. This field shifts the focus onto exactly what the definition needs. The choice not to work with subtypes is not free of cost. For example, in the backward direction of `lemma retraction_criterion_closed'` we will have to precompose with a projection onto I in order to make the `MapsTo` field applicable, as described in Section 4.3.

The set `rangeH'` and the variant `HEPY`

The set `rangeH'` has no equivalent in the informal Definition 2.1. It is a set in the target type `Z` in which the extended homotopy `H'` is required to take its values. It forces two additional hypotheses `Set.range f ⊆ rangeH'` and `(smallcyl A).MapsTo H rangeH'`. Without them there could be no such extension, since `H'` extends both `f` and `H`.

The gain of carrying this set along is that it lets us apply the HEP with a *subset* of a type as codomain without ever forming the corresponding subtype. The application we have in mind is the retraction criterion. There the HEP is applied with `Z := X × ℝ` and `rangeH' := anchor A`, so that the extension it produces is a self-map of `X × ℝ` whose image lies in the anchor. This is precisely what the formalisation of a retraction onto the anchor requires, as discussed later on. In most applications one takes `rangeH'` to be the whole target type, written `@ Set.univ Z` in Lean. The two additional hypotheses become statements that `tauto` solves. Since this extra work is negligible, we prefer this definition over the shorter variant discussed next.

Because `rangeH'` has no informal counterpart, one could ask whether it changes the notion being defined. To settle this we also formalise the literal

reading of Definition 2.1, in which the extension is only required to be a map into `Z`.

```
structure HomotopyExtensionY (H' : X × ℝ → Z) (f : X → Z) (H : X × ℝ → Z)
    (A : Set X) : Prop where
  ContinuousOn : ContinuousOn H' (cyl X)
  agreef : ∀ (x : X), f x = H' (x, 0)
  agreeH : ∀ (a : A) (t : unitInterval), H (a,t) = H' (a, t)

def HEPY (X : Type u) [TopologicalSpace X] (A : Set X) : Prop :=
  ∀ (Z : Type u) [TopologicalSpace Z], ∀ (f : X → Z), ∀ (H : X × ℝ → Z),
  Continuous f → ContinuousOn H (smallcyl A) → agreeOn f H A →
  ∃ (H' : X × ℝ → Z), HomotopyExtensionY H' f H A
```

The lemma `HEP_iff_HEPY` shows that the two definitions are equivalent, so that `rangeH'` is a matter of convenience and not of content. We will not use `HEPY` again in this project. Its only purpose is this equivalence.

Universes

In the definition the target type `Z` is quantified over Type u, the same universe as `X`. This is caused by the way we use the definition. A quantifier `∀ (Z : Type v)` can only be filled with codomains lying in exactly that universe. Every codomain we actually need, however, lies in the universe of `X`. In the retraction criterion, for instance, the codomain is `X × ℝ`. Quantifying over a smaller universe would make the definition inapplicable to precisely the instances we care about, and quantifying over a larger one would admit them only after inserting explicit lifts.

This raises the question whether restricting `Z` to a single universe weakens the property. There is a short argument why the definition is equivalent to a more general one: the retraction criterion shows that `HEP`, at least in the case that the subspace is closed, is equivalent to the existence of a retraction, and that statement does not mention the target type `Z` or its universe at all. Hence the property for target types in `Type u` already implies it for target types in any universe.

The variant `HEP'`

Let us now turn to the variant `HEP'`, which changes the shape of the pair of spaces.

```
open Set.Notation
def HEP' {Y : Type u} [TopologicalSpace Y] (X A : Set Y) : Prop :=
  HEP X (X ↓∩ A)
```

The notation `(X ↓∩ A)` means that we regard `(X ∩ A)` as a set of type `Set ↑X`. The difference is that in `HEP` the set `A` is a subset of the type `X`, whereas

in `HEP'` both `X` and `A` are subsets of one ambient type `Y`.
Two consequences of defining `HEP'` through `HEP` are worth noting. First, `HEP'` does not require A to be a subset of X. To be precise, we do not state that the pair (X, A) has the HEP, but that the pair $(X, X \cap A)$ does, and this is the intended statement only if $A \subseteq X$. In some proofs, such as `HEP_trans`, the inclusion therefore appears as an explicit hypothesis.
Secondly, `HEP` and `HEP'` are definitionally equal, so results transfer between them for free, as we will see for example when `HEP_ball_boundary'` is proved by `HEP_ball_boundary` alone.

We introduced `HEP'` when we came to pairs of consecutive skeleta, since a skeleton of a relative CW-complex is a subcomplex whose carrier is a subset of an ambient space. So this is the shape in which the objects actually arise. It also turned out to be the better fit for balls and spheres: the sphere is then simply the set of points of norm one in the ambient normed space, rather than a subset of the subtype of the closed ball.

Note that in this definition the maps `f` and `H` to be extended are defined on the subtype `↑X`, respectively `↑X × ℝ`. Working with these subtypes caused some extra work in the proof of `retraction_criterion_closed'`. One could avoid it by not defining `HEP'` in terms of `HEP` but writing it out with maps defined on `Y` and `Y × ℝ`. The definition would be longer, since for instance continuity of `f` would become a `ContinuousOn` statement. Also, the equivalence with `HEP` would have to be proven rather than holding by definition. Nevertheless, it is an interesting modification that should be considered when looking for the best formalisation, even though we do not pursue it further in this work.

4.3 Retraction Criterion

Before turning to the formalisation of the retraction criterion itself, we look at the definition of a retraction. Classically, a retraction r from B onto A is defined via precomposition with the inclusion $A \hookrightarrow B$, as in Definition 2.6. Formalising the retraction as a map from B to A would make the codomain a subtype, which is inconvenient to work with. Instead we let the retraction be a map from an ambient type X to itself. The structure `RetractionOn` bundles the properties that make such a map a retraction.

```
structure RetractionOn (r : X → X) (B A : Set X) : Prop where
  subset : A ⊆ B
  continuousOn : ContinuousOn r B
  mapsTo : B.MapsTo r A
  fixesOn : ∀ a ∈ A, r a = a
```

As the name `RetractionOn` suggests, the map `r` is a self-map of the type `X` that need not be a retraction on all of its domain. It only "retracts" a

subset `B` onto a subset `A`. The four fields translate the classical conditions as follows. The first property `subset` states that `A` is contained in `B`. Note that omitting this property would define a more general class of self-maps that need not satisfy $r(B) = A$. The field `continuousOn` requires continuity on the relevant subset B only. Since the codomain of a retraction is supposed to be `A`, the field `mapsTo` requires that `r` sends every element of `B` to an element of `A`. Finally, the classical definition demands that precomposition with the inclusion $A \hookrightarrow B$ gives the identity id_A. In our setting there is no inclusion map, since domain and codomain of `r` are the same type `X`, so the condition becomes the requirement that every element of `A` is fixed by `r`. This is the field `fixesOn`.

Recall that the retraction criterion was obtained from two lemmas, and the formalisation follows the same structure. Since Lemma 2.5 consists of two implications, only one of which requires closedness, we split it into two separate lemmas. The forward direction is called `if_HEP_then_extension`, and the backward direction under the closedness assumption is called `if_extension_then_HEP`. Their proofs are unspectacular and follow the argument in the mathematical section of Lemma 2.5 closely. The same holds for the formalisations of Lemma 2.7 and of the retraction criterion Proposition 2.8.

```
lemma retraction_criterion_closed (hA1 : IsClosed A) : HEP X A ↔
    ∃ r : X × ℝ → X × ℝ, RetractionOn r (cyl X) (anchor A) := by ...
```

Here the retraction is a map on the product `X × ℝ`, where `X` is the larger space of the pair. Although the lemma is phrased in terms of `HEP`, it can be applied to `HEP'` as well, since the two definitions are definitionally equal:

```
variable (X A : Set Y) (hHEP' : HEP' X A)
#check if_HEP_then_retraction hHEP'
if_HEP_then_retraction hHEP' : ∃ (r : ↑X × ℝ → ↑X × ℝ), RetractionOn r (cyl ↑X)
(anchor (X ↓∩ A))
```

While this does yield a retraction for the spaces in question, it is not formulated in the spirit of `HEP'`: the retraction still lives on `↑X × ℝ` for a subtype `↑X`, whereas `HEP'` is stated for subsets of an ambient type. We therefore also formalised the lemma `retraction_criterion_closed'`, in which the retraction is a self-map of `Y × ℝ`, where `Y` is the ambient space. As before, the Lean file contains the forward implication without the closedness assumption as well as the equivalence under the assumption.

```
lemma retraction_criterion_closed' (X A : Set Y) (hAX : A ⊆ X)
    (hA1 : IsClosed A) : HEP' X A ↔ ∃ s : Y × ℝ → Y × ℝ, RetractionOn s
    (smallcyl X) (smallanchor X A) := by ...
```

At this point the ambient-space formulation introduces a discrepancy with the informal statement, which went unnoticed during the formalisation.

This discrepancy can occur because a proof assistant is not able to guarantee that the formal statement says what one intended it to say. In Lemma 2.5 and Proposition 2.8 the hypothesis is that A is closed in X, that is, in the subspace topology of X. In `retraction_criterion_closed'`, however, both A and X are subsets of the ambient type `Y`, and `hA1 : IsClosed A` asserts that A is closed in `Y`. This is strictly stronger.

In practice this turned out to be convenient rather than restrictive, since all closedness facts we need are available in the ambient space anyway: the skeleta of a relative CW-complex are closed in the ambient type, as is the sphere S^{m-1} in $\mathbb{R}^m$. Ideally one would drop the assumption entirely. Hatcher [Hat02] states the retraction criterion without it, the general case being proved in the appendix following an argument of [Str68]. Formalising that argument would remove the discrepancy, but it is considerably more involved than the proof we give.

We proved `retraction_criterion_closed'` from `retraction_criterion_closed`. For the forward direction we obtain a retraction `r` as in the `#check` above. In order to extend this map `r : ↑X × ℝ → ↑X × ℝ` to a map `s : Y × ℝ → Y × ℝ`, we precompose in the first component with a map `mapYX : Y → X`, which is the identity on `X` and sends all other points of `Y` to an arbitrary junk value, and postcompose with `Subtype.val`, which sends `(r x).1 ∈ X` back to the corresponding point of `Y`. It remains to confirm that this newly defined map is again a retraction, that is, that it is continuous on the relevant set, maps into the required one, and leaves the necessary values unchanged.

For the backward direction we start with a retraction `s : Y × ℝ → Y × ℝ` and want to construct `r : ↑X × ℝ → ↑X × ℝ`, so that `retraction_criterion_closed` gives the claim. One might expect this direction to be the easier one, since no new values have to be constructed and `s` only needs to be restricted to the desired space. This expectation is disappointed at the point at which the codomain of `r` is required to be $X \times \mathbb{R} \subseteq Y \times \mathbb{R}$. All we know about the range of `s` is that its codomain is $Y \times \mathbb{R}$ and that it is a retraction on `smallcyl X`, so the field `RetractionOn.mapsTo` gives no information about where in $Y \times \mathbb{R}$ a point from `X × ℝ` is sent once its second coordinate lies outside the unit interval. We resolved this by precomposing the real coordinate with

```
ProjIcc : ℝ → ℝ := fun t ↦ Set.projIcc (0 : ℝ) 1 zero_le_one t,
```

where `Set.projIcc` is a continuous map from Mathlib that retracts a linearly ordered type onto a closed interval in it. After this precomposition the `mapsTo` field of `s` is applicable to every point we feed into it. This extra work is a consequence of working with types rather than subtypes; it could have been avoided by defining homotopies on `Y × unitInterval` instead of `Y × ℝ`.

The two versions of the criterion differ in two independent aspects, which are worth keeping apart. The first is which definition of HEP appears in the

statement. As `#check` above shows, this is a matter of style. `HEP` and `HEP'` are definitionally equal, so either criterion could have been stated in terms of either definition. The second aspect is where the retraction lives, on `X × ℝ` for the larger type `X` or on `Y × ℝ` for an ambient type `Y`, and here the two forms are not interchangeable. While we preferred working with the ambient type for the definition of the homotopy extension property, here we found both versions valuable in practice. We give an example for each:
The unprimed version `retraction_criterion_closed` is what the proof for two consecutive skeleta needs, where we construct a retraction `r : (skeletonLT X m) × ℝ → (skeletonLT X m) × ℝ`. The key ingredient for the continuity of this map is `Topology.IsQuotientMap.continuous_lift_prod_left`, Mathlib's formalisation of Theorem 2.18 on products of quotient maps with a locally compact factor. Working with an ambient space here would have destroyed the surjectivity of the quotient map, and the theorem could not have been applied in the form in which Mathlib provides it.
The primed version, on the other hand, is what `PartialHomeomorph_HEP'` needs, as we discuss in Section 4.4. A partial homeomorphism is a map between two ambient types, so a retraction defined on the ambient type rather than on the larger type can be used more directly.

4.4 Preservation of the HEP

Besides the retraction criterion we formalised two more results that are useful for proving that a pair of topological spaces has the HEP. The first is that the HEP is preserved under partial homeomorphisms.

```
lemma PartialHomeomorph_HEP' {Y' : Type u} [TopologicalSpace Y'] {X A : Set Y}
  (hA : A ⊆ X) {X' A' : Set Y'} (hA' : A' ⊆ X') (hHEP : HEP' X A)
  (f : PartialHomeomorph Y Y') (source : X = f.source) (target : X'
  = f.target) (h2 : f '' A = A') (hA'closed : IsClosed A') :
  HEP' X' A' := by ...
```

The two pairs of this lemma are `(X,A)` in the ambient type `Y` and `(X',A')` in a second ambient type `Y'`. A `PartialHomeomorph Y Y'` in Mathlib is a map between the ambient types `Y` and `Y'` together with a distinguished source and target on which it is a homeomorphism. The hypotheses `source`, `target` and `h2` identify the two pairs of spaces with these data, and they are stated as equations so that they can be removed by `subst` in the first step of the proof. Afterwards the whole argument is phrased in terms of `f.source`, `f.target` and `f '' A`, and no transport along the equations is needed anywhere else. Here the ambient formulation of the retraction criterion pays off. We apply `if_HEP_then_retraction'` to `hHEP` and obtain a retraction of `smallcyl f.source` onto `smallanchor f.source A`. Since this retraction is a self-map of `Y × ℝ`, we can conjugate it with the partial homeomorphism, which is likewise a map between the ambient types. In the first coordinate we precompose with

`f.symm` and postcompose with `f`. If only the unprimed criterion were available, the retraction would live on `↑X × ℝ`, and we would first have to turn `f` into a homeomorphism `↑f.source ≃ₜ ↑f.target` and work with subtype coercions.

Note that only `A'` is required to be closed. The two subspaces enter the proof through opposite implications of the retraction criterion: from `hHEP` we extract a retraction, which is the forward implication and needs no closedness assumption, whereas for `A'` we argue in the other direction and construct the HEP from a retraction, which is where closedness is needed.

The second result is a transitivity property: if the pairs (A_1, A_2) and (A_2, A_3) both satisfy the HEP, then so does the pair (A_1, A_3). ↗

```
lemma HEP_trans (A1 A2 A3 : Set Y) (h21_sub : A2 ⊆ A1) (h12_hep : HEP' A1 A2)
  (h32_sub : A3 ⊆ A2) (h23_hep : HEP' A2 A3) : HEP' A1 A3 := by ...
```

For the formalisation of this proof we had a choice between two approaches. Instead of applying the retraction criterion to all three `HEP` statements and constructing the desired retraction from the two given ones, we proved the lemma directly from the definition. This turned out to be the shorter way, and a further advantage is that no closedness assumptions are needed, whereas applying the retraction criterion would have required them.

In retrospect, `PartialHomeomorph_HEP'` could be proved in the same manner, by checking the definition and constructing the extended homotopy explicitly rather than going through the retraction criterion. In that case the closedness assumption on `A'` could be dropped as well. With the current definition of `HEP'`, however, in which the maps we wish to extend live on subtypes, such a proof would be less pleasant to carry out.

4.5 HEP for balls and cubes

The goal of the file `HEP_ball_cube.lean` is to prove the homotopy extension property for balls and cubes relative to their boundary. These are not only the first interesting pairs of spaces for which we want to prove the HEP, but also the first step in the proof of the homotopy extension property for relative CW-complexes.

We start with the balls. As in the proof of Theorem 2.16, we first construct the auxiliary function `aux_fun`, which combines the information of the given maps `f` and `H`. This function is then used to define the desired homotopy `H'` explicitly, with which we can check the definition of `HEP`. Following the notation of Section 4.1, the target type is `Z` and `rangeH' ⊆ Z` bounds the range of the maps; here `Z` has to lie in `Type 0`, since the pair of this section does. The maps `f` and `H` are declared as variables as well, because most lemmas of the first half of the file speak about them. ↗

```
variable {m : ℕ} {Z : Type} [TopologicalSpace Z] (rangeH' : Set Z)
  (f : (Metric.closedBall (0 : EuclideanSpace ℝ (Fin m)) 1) → Z)
  (H : (Metric.closedBall (0 : EuclideanSpace ℝ (Fin m)) 1) × ℝ → Z)
```

```
def aux_fun : EuclideanSpace ℝ (Fin m) → Z := fun p ⇒
  if hp : norm p ≤ 1 then f ⟨p, by simp[hp]⟩
  else H (⟨((norm p)⁻¹ : ℝ) • p, inv_norm_smul_mem_unitClosedBall p⟩,
  norm p - 1)

def H' : (Metric.closedBall (0 : EuclideanSpace ℝ (Fin m)) 1) × ℝ → Z :=
  fun p ⇒ (aux_fun f H ) ((1 + p.2) • p.1)
```

We split the `ContinuousOn` proof of `H'` into three steps. We start with a lemma stating that `aux_fun` is continuous on the annulus $\{x \mid 1 \leq \|x\| \leq 2\} \subset \mathbb{R}^m$ and use it to prove continuity of `aux_fun` on the ball of radius two. That ball can be written as the union of the ball of radius one with the annulus, and continuity on both of these closed subsets yields continuity on their union. The continuity of `H'` then follows by composing two `ContinuousOn` functions. The proofs of the remaining properties of `H'` are not hard. Solving, rewriting and confirming inequalities make up a large part of them. All these results combined prove the lemma `HEP_ball_boundary`. The primed version `HEP_ball_boundary'` is definitionally equal.

```
lemma HEP_ball_boundary : ∀ (m : ℕ),
  HEP (Metric.closedBall (0 : EuclideanSpace ℝ (Fin m)) 1) (Metric.sphere
  ⟨0, by simp⟩ 1) := by ...

lemma HEP_ball_boundary' : ∀ (m : ℕ),
  HEP' (closedBall (0 : EuclideanSpace ℝ (Fin m)) 1) (sphere 0 1) :=
  HEP_ball_boundary
```

The `Metric.sphere` in Lean has type `Set α`, where `α` is the type of the center, in our case the zero of an m-dimensional Euclidean space. By the way `HEP` is defined, the sphere in `HEP_ball_boundary` is a subset of the closed ball, so we have to add the `by simp` proof in order to construct the element of the subtype. For `HEP'`, on the other hand, both spaces live in the same ambient space. No proof has to be supplied, and Lean interprets the zero as the origin of $\mathbb{R}^m$. Here we see an advantage of the `HEP'` definition in action, since we can reduce the use of subtypes.

In Section 3.3.1 we saw that the source of the characteristic map of a cell is a cube rather than a ball. To prove the `HEP` for these cubes, we want to use the fact that an m-dimensional ball is homeomorphic to a cube of the same dimension.

Instead of constructing this partial homeomorphism explicitly, we obtain it from the following Mathlib lemma. It states that for a convex bounded set with nonempty interior in a real normed space there is a homeomorphism of the ambient space with itself that carries the interior, the closure and the frontier of the set onto the open unit ball, the closed unit ball and the unit sphere respectively.

```
exists_homeomorph_image_interior_closure_frontier_eq_unitBall.{u_1}
  {E : Type u_1} [NormedAddCommGroup E][NormedSpace ℝ E] {s : Set E}
  (hc : Convex ℝ s) (hne : (interior s).Nonempty)
  (hb : Bornology.IsBounded s) :
  ∃ h, ↑h '' interior s = ball 0 1 ∧ ↑h '' closure s = closedBall 0 1 ∧
  ↑h '' frontier s = sphere 0 1
```

The `Metric.closedBall` for which we already proved the `HEP` is convex, bounded and has nonempty interior, so it satisfies all three assumptions. Applying the lemma to it directly would nevertheless gain us nothing: inside `EuclideanSpace ℝ (Fin m)` this set already is the closed unit ball, so the lemma would only produce a homeomorphism carrying it onto itself, and no cube would appear anywhere. What makes the cube appear is that in `Fin m → ℝ` with maximum norm the closed unit ball is the cube. We therefore first transport our closed ball into that space along the homeomorphism

```
def homeo_euclid_max : EuclideanSpace ℝ (Fin m)  ≃ₜ (Fin m → ℝ) where
  toFun := fun p ↦ p
  ...
```

The map `homeo_euclid_max` sends each point to itself and only changes the norm of the space. The image of the `Metric.closedBall` is therefore still the same round convex set, even though it is no longer the closed unit ball of the codomain. Applying the lemma to this image yields a homeomorphism of `Fin m → ℝ` to itself that carries the round set onto the cube:

```
def G : (Fin m → ℝ) ≃ₜ (Fin m → ℝ):=
  (exists_homeomorph_image_interior_closure_frontier_eq_unitBall
  convex_euclid_to_max_ball nonempty_euclid_to_max_ball
  bounded_euclid_to_max_ball).choose
```

Since the lemma only asserts that such a homeomorphism exists, we use `.choose` to fix and name one of them, and its three defining properties are then available through `.choose_spec`.
The composition of these two homeomorphisms is in particular a partial homeomorphism from `EuclideanSpace ℝ (Fin m)` to `Fin m → ℝ` with source the closed ball and target the closed cube. Applying the lemma `PartialHomeomorph_HEP'` now proves

```
lemma HEP_cube_boundary' (m : ℕ) : HEP' (closedBall (0 : (Fin m → ℝ)) 1)
  (sphere 0 1) := by ...

lemma HEP_cube_boundary (m : ℕ) : HEP (closedBall (0 : (Fin m → ℝ)) 1)
  (sphere ⟨0, by simp⟩ 1) := HEP_cube_boundary' m
```

Before concluding this section, we want to point out that here the definition of `HEP'` matches the structure of a partial homeomorphism particularly well. When we first proved this result for `HEP`, before `HEP'` was defined, we

had to show that the `Subtype.val` image of a sphere in the closed ball is the same as the sphere in the ambient space, and likewise for the closed ball itself. With `HEP'` these subtype coercions are avoided and the proof is more straightforward. From this result the statement for `HEP` follows directly.

4.6 HEP for consecutive skeleta

Now we continue with the second step in the proof of the HEP for relative CW-complexes, namely that every pair of two consecutive skeleta satisfies the HEP. The formalisation of this result builds on the following Mathlib lemma, which takes the place of the theorem stating that the product of a quotient map with the identity on a locally compact space is again a quotient map (Theorem 2.18).

```
theorem Topology.IsQuotientMap.continuous_lift_prod_left {X₀ : Type u_1}
  {X : Type u_2} {Y : Type u_3} {Z : Type u_4} [TopologicalSpace X₀]
  [TopologicalSpace X] [TopologicalSpace Y] [TopologicalSpace Z]
  [LocallyCompactSpace Y] {f : X₀ → X} (hf : IsQuotientMap f)
  {g : X × Y → Z} (hg : Continuous fun (p : X₀ × Y) ⇒ g (f p.1, p.2)) :
  Continuous g
```

The theorem tells us that a map `g` defined on the product of a quotient space with a locally compact space is continuous, provided that its composition with the quotient map in the first factor is. We want this `g` to be the map that yields the HEP for two consecutive skeleta. Mathematically it is the retraction of $X_m \times I$ onto $X_m \times \{0\} \cup X_{m-1} \times I$. In the formalisation it takes the shape `skeletonLT X (m+1) × unitInterval → skeletonLT X (m+1) × ℝ`, which will be discussed later on. For now, let us take a closer look at which functions we need to define and which results we have to prove in order to get there.

4.6.1 Skeleton is a quotient space

First of all we need a quotient map onto the skeleton `skeletonLT X m` called `f` in the theorem above. For this we use the fact that we can "unglue" the skeleton into the base space and the disjoint union of closed balls, or cubes to be precise, in all dimensions less than m. The identity map on the base space and the characteristic maps on the closed balls assemble into the map `SkeletonProjection`.

```
abbrev ungluedSkel (m : ℕ) :=
  C ⊕ (Σ (n : Fin m) (_ : cell X n), (closedBall (0 : Fin n → ℝ) 1))

def SkeletonProjection (m : ℕ) : ungluedSkel X m → skeletonLT X m :=
  Sum.elim (fun c ↦ ⟨c, by ... ⟩) (fun ⟨n, i, x⟩ ⇒ ⟨map n i x, by ...⟩)
```

To prove that this indeed defines a quotient map, we have to check that it is surjective, continuous and that it induces the topology on the skeleton. The skeleton is a subcomplex of the relative CW-complex, and its topology is described in terms of the closed sets: A subset of the subcomplex is closed precisely when its intersection with the base space is closed there and its intersection with each closed cell is closed in that cell.
The surjectivity proof is straightforward. A point of the skeleton lies either in the base space or in an open cell, and in the first case it has a preimage in `C`, in the second in the closed ball belonging to that cell. Continuity is not much work either, since `SkeletonProjection` is assembled from continuous functions on a disjoint union. Proving that the topology on the skeleton is the induced one requires more effort.

We start with a set `S` of type `Set (skeletonLT X ↑m)` and the assumption that its preimage under `SkeletonProjection` is closed in the unglued skeleton. The goal is to show that `S` is closed in the skeleton. In Mathlib the results about closed sets are always phrased for subsets of the ambient space, together with the assumption that the set is contained in the CW-complex or in the subcomplex. To get into this setting, we need the following lines in the proof: 🔗

```
set S' : Set Y := Subtype.val '' S with hS'def
have hS'sub : S' ⊆ (skeletonLT X m):= by
  rintro _ ⟨a, _ , rfl⟩;
  exact a.2
```

Now we can apply the lemma `Topology.RelCWComplex.isClosed_of_disjoint_openCell_or_isClosed_inter_closedCell` 🔗, which states that `S'` is closed in the relative CW-complex if its intersection with the base space is closed and if for every cell either `S'` and `openCell n j` are disjoint or `S' ∩ closedCell n j` is closed. We prove the case of the base space in `hClosedBase` and the case of cells of dimension less than m in `hClosedLT`. For higher dimensions `S'` is disjoint from the open cell, which is proved in `SkeletonProjectionIsQuotientMap` itself.

4.6.2 Construction of the retraction

The next preparation is the construction of the map `r`, which is to be the retraction for two consecutive skeleta. As in the mathematical proof, we do not define the retraction map `r` on `skeletonLT X (m + 1) × I` directly. Instead we say what it should do on the unglued skeleton, where the base space and the individual cubes are still separate, and then push the resulting map down along the quotient map `SkeletonProjection`. We build this up from the bottom: first the map on all single cubes, called `cubesretract`, then the map on the product of the whole unglued skeleton and the unit interval, which is

called `sumMap` and was called f_m in the mathematical section, and finally `r` itself.

On a cube of top dimension the retraction we need is already available. It is the map `r_cube` that the retraction criterion provides from `HEP_cube_boundary`, the HEP of a cube relative to its boundary proved in Section 4.5. On cubes of lower dimension nothing should happen. The map `cubesretract` combines the two cases. Apart from the case distinction it only coerces the second coordinate from the unit interval to $\mathbb{R}$.

```
def cubesretract {m : ℕ} (n : Fin (m + 1)) :
    (closedBall (0 : Fin (n : ℕ) → ℝ) 1) × unitInterval →
    (closedBall (0 : Fin (n : ℕ) → ℝ) 1) × ℝ := fun p ↦
  if n = m then (r_cube n (p.1, Subtype.val p.2))
  else (p.1, Subtype.val p.2)
```

For `cubesretract` we proved lemmas about its continuity, about which points it fixes, and about the set in which its image lies. The last one comes in a version for the lower dimensions and one for the top dimension, matching the two branches of the definition.

```
lemma cubesretract_continuous {m : ℕ} (n : Fin (m + 1)) : Continuous
  (cubesretract n) := by …

lemma cubesretract_fixedOn {m : ℕ} {n : Fin (m + 1)}
  (x : (closedBall (0 : Fin n → ℝ) 1) × unitInterval) :
  (n < m → cubesretract n x = (x.1, x.2.1)) ∧
  (n = m → x ∈ {z | z.2 = 0 ∨ z.1 ∈ sphere ⟨0, by simp⟩ 1 } →
  cubesretract n x = (x.1, x.2.1)) := by …

lemma cubesretract_mapsto_lt {m : ℕ} (n : Fin (m + 1)) (i : cell X n)
  (p : (closedBall 0 1) × ↑unitInterval) (hn : n < m) :
  (map n i) (cubesretract n p).1 ∈ skeletonLT X m ∧
  (cubesretract n p).2 ∈ unitInterval := by …

lemma cubesretract_mapsto_top {m : ℕ} (n : Fin (m + 1)) (i : cell X n)
  (p : (closedBall 0 1) × ↑unitInterval) (hn : n = m) :
  (cubesretract n p).2 = 0 ∨ (map n i) (cubesretract n p).1
  ∈ skeletonLT X m ∧ (cubesretract n p).2 ∈ unitInterval := by …
```

Next we assemble these maps into a map on the product of the whole unglued skeleton with the unit interval.

```
def sumMap (m : ℕ) :
  (ungluedSkel X (m + 1)) × unitInterval → skeletonLT X (m+1) × ℝ :=
  fun p ↦ p.1.elim
    (fun c ⇒ (⟨c.val, memBase_memSkeletonLT X c.prop (m + 1)⟩, p.2))
    (fun ⟨n, i, x⟩ ⇒ (⟨map n i (cubesretract n (x,p.2)).1,
    cubesretract_mapsto_weak n i (x,p.2)⟩, (cubesretract n (x,p.2)).2))
```

The map `sumMap` is defined using `Sum.elim`, which performs a case analysis on a sum type and applies the corresponding function depending on which constructor is present (cf. `Init.Data.Sum.Basic` [The]), that is, on whether the first coordinate of the point `p` lies in the base space `C` or in a closed ball. In the first case we want to do nothing except apply `SkeletonProjection` to that coordinate, which on the base space is just the identity. We send the point `(c : C)` to itself, but regarded as an element of the skeleton. Since the proof that a point of the base space lies in any given skeleton is needed repeatedly, we made it a lemma of its own, called `memBase_memSkeletonLT`.
In the second case, where the first coordinate of `p` comes from a closed ball, we first apply `cubesretract` to the point and then postcompose with what `SkeletonProjection × id` would do to such a point. Since the first coordinate still lies in a closed ball after `cubesretract` has been applied, `SkeletonProjection` amounts to applying the characteristic map of that closed ball. We have to proof that the image of this characteristic map really lies in the skeleton. This proof is again a lemma of its own, called `cubesretract_mapsto_weak`, which is derived from the two `mapsto` statements above.

It remains to push `sumMap` down from the unglued skeleton to the skeleton itself, which is what `Function.extend` does.

```
def r (m : ℕ) : skeletonLT X (m + 1) × unitInterval → skeletonLT X (m + 1) × ℝ
  := Function.extend (fun p ↦ ((SkeletonProjection X (m + 1) p.1), p.2))
  (sumMap X m) (Prod.map id Subtype.val)
```

`Function.extend` works in such a way that whenever a point `x ∈ skeletonLT X (m + 1)` has a preimage `p ∈ ungluedSkel × I` under the map `(fun p ↦ ((SkeletonProjection X (m + 1) p.1), p.2))`, then `r x = sumMap X m p`, and otherwise `r x = (Prod.map id Subtype.val) x`. This is not captured in the definition but we know that `SkeletonProjection × id` is surjective, so the second case of `Function.extend` could be filled with any junk function, since it will never be applied. One can therefore read the definition of `r` simply as: take any preimage under `SkeletonProjection` and apply `sumMap` to it. That this is well defined is the content of the lemma `factors` in the next section.

It may seem odd that `r`, which is supposed to be a retraction, is formalised with the unit interval as the second factor of its domain, where we would usually take the real numbers, since our retractions are defined on a product with $\mathbb{R}$. But observe that the theorem `Topology.IsQuotientMap.continuous_lift_prod_left` yields `Continuous` and not `ContinuousOn`. We should therefore not make the domain any bigger than what is mathematically relevant. In the codomain we do not have this kind of problem and can simply choose `Z` to be the skeleton times the entire real line.

Taking a closer look at the definition of `sumMap`, one sees that in the case where `p` comes from the base space or from a closed ball of dimension less than `m`, it does the same as `SkeletonProjection × id`, at least if one ignores the subtype coercions. It would have been inconvenient to write this application into the definition of `sumMap`, so instead we proved the two lemmas that state exactly this equality.

```
lemma sumMap_eq_SkelProj_C (m : ℕ) (x : C) (t : unitInterval) : sumMap X m
  (Sum.inl x, t) = Prod.map id (Subtype.val) (SkeletonProjection X
  (m + 1) (Sum.inl x), t) := by …

lemma sumMap_eq_SkelProj_LtOrSphere {m : ℕ} (n : Fin (m + 1))
  (i : cell X n) (x' : closedBall (0 : Fin n → ℝ) 1)
  (t : unitInterval) (hx : n < m ∨ n = m ∧ x' ∈ sphere
  ⟨0, by simp⟩ 1) : sumMap X m (Sum.inr ⟨n, ⟨i, x'⟩⟩, t) =
  Prod.map (SkeletonProjection X (m + 1)) Subtype.val
  (Sum.inr ⟨n, ⟨i, x'⟩⟩, t) := by …
```

4.6.3 Continuity of the retraction

Now that all relevant maps are defined, we can turn to the proof that `r` is continuous. As announced, the continuity comes from the theorem `Topology.IsQuotientMap.continuous_lift_prod_left`. Let us see what is left to prove:

```
lemma continuousRetract (m : ℕ) : Continuous (r (X := X) m) := by
  refine (SkeletonProjectionIsQuotientMap X (m +1)).continuous_lift_prod_left ?_
  ...
```

After this first line of the proof the current goal becomes

```
Continuous fun (p : (ungluedSkel X m) × I) ⇒ r (SkeletonProjection X
  (m + 1) p.1, p.2)
```

Showing this comes down to two major steps. The first is that `sumMap` factors through `(fun p ↦ ((SkeletonProjection X (m + 1) p.1), p.2))`, from which it follows easily that the continuity goal is equivalent to the continuity of `sumMap`. That continuity is then the second step. We start with

```
lemma factors (m : ℕ) : (sumMap X m).FactorsThrough (fun p ↦
  ((SkeletonProjection X (m + 1) p.1), p.2)) := by
  intro x y hxy
  ...
```

Here `x` and `y` are two points of `ungluedSkel × I` whose second coordinates agree and which have the same image under `SkeletonProjection`. We have to show that `sumMap X m x = sumMap X m y`. In `sumMap_eq_SkelProj_C` and `sumMap_eq_SkelProj_LtOrSphere` we already have useful tools for this goal. One part that required some work is the following:

```
lemma SkeletonProj_inj_top {m : ℕ} {n : Fin (m + 1)} (hnm : ↑n = m)
  (i : cell X n) (x : closedBall (0 : Fin n → ℝ) 1) (x_ball : x.1 ∈ ball
  (0 : Fin n → ℝ) 1) (y : ungluedSkel X (m + 1))
  (hskel : SkeletonProjection X (m + 1) (Sum.inr ⟨n, ⟨i, x⟩⟩) =
  SkeletonProjection X (m + 1) y) : y = Sum.inr ⟨n, i, x⟩ := by ...
```

The lemma `SkeletonProj_inj_top` says that if `x` lies in a specific open ball of top dimension and `y` is any other point of `ungluedSkel` with the same image under `SkeletonProjection`, then `y` is that very point `x`.
The proof of `factors` is long because it splits into many case distinctions, beginning with whether `x` lies in the base space or in a closed ball, and considering in each case the same two options for `y` separately. In the case where `x` lies in a closed ball we distinguish further whether it lies in the top-dimensional open cell or not. Most of these cases are quite short in themselves.
In the mathematical argument it was enough to distinguish whether a point lies in the top-dimensional cell or not. So one may ask why the formalisation splits the situation up so much further. The reason is that we cannot simply say that a point lies in the base space. A point of `ungluedSkel` is given by a constructor, either as `Sum.inl c` for some `c : C` or as `Sum.inr ⟨n, i, x⟩`, so the case analysis over the constructors has to be made anyway, and it is then easier to treat the resulting combinations separately. For the continuity of `sumMap` we precompose the map with the homeomorphism

```
def homeoProd (m : ℕ) :
  (C × unitInterval) ⊕ (Σ (n : Fin (m + 1)) (_ : cell X n),
  (closedBall (0 : Fin n → ℝ) 1) × unitInterval) ≃ₜ
  (C ⊕ (Σ (n : Fin (m + 1)) (_ : cell X n), closedBall (0 : Fin n → ℝ) 1))
  × unitInterval := …
```

The domain of this composite is no longer a product, but a disjoint union, and on each summand the continuity goal is proved by combining results established earlier, such as `cubesretract_continuous` and the continuity of the characteristic maps. Since continuity is preserved under precomposition with a homeomorphism, this yields that `sumMap`, and finally also `r`, is continuous.

4.6.4 Properties of the retraction

In this section we conclude the proof of the HEP for consecutive skeleta. For the proof via the retraction criterion we have to make a small adjustment to the map `r`. As mentioned above, the desired retraction should have `(skeletonLT X (m + 1)) × ℝ` as its domain, whereas `r` is only defined on the unit interval in the second component. The easiest way to extend `r` turned out to be precomposing the second coordinate with the map `ProjIcc`, which was defined in the first file `HEP_definition.lean`. The final definition therefore looks as follows.

```
def retr_skeleton (m : ℕ) : (skeletonLT X (m + 1)) × ℝ → (skeletonLT X
  (m + 1)) × ℝ := fun p ↦ (r X m) (p.1, ⟨ProjIcc p.2, proj_mem p.2 ⟩)
```

It only remains to show that `retr_skeleton` is continuous, maps into the right set and fixes the points it has to fix. The continuity follows directly from all the work done before, and the claims about `mapsTo` and `fixesOn` are first proved for `sumMap` and then transferred to `retr_skeleton`. Altogether this results in the following lemma.

```
lemma HEP'_skeleton (m : ℕ) : HEP' (skeletonLT X (m + 1)).carrier
    (skeletonLT X m).carrier := by
  apply (retraction_criterion_closed (skeletonLT X ↑m).closed'.preimage_val).2
  use retr_skeleton X m
  ...
```

Note that although the statement is phrased in terms of `HEP'`, since both skeleta live in the ambient type, the proof applies the unprimed `retraction_criterion_closed`, which is possible because the two definitions are definitionally equal.

4.7 HEP for finite dimensional relative CW-complexes

In this last part of the formalisation we arrive at the result that finite dimensional relative CW-complexes satisfy the HEP. As the mathematical chapter already showed, the finite dimensional case follows from what we have proved so far by a straightforward induction, whereas the general case requires considerably more work. Given the time available for this thesis, we had to stop at this point.
We first run the induction over the skeleta, and then show that a finite dimensional complex is one of its own skeleta. In Lean the induction looks as follows.

```
lemma HEP_skelLT_base (m : ℕ) : HEP' (skeletonLT X m) C := by
  induction m with
  | zero ⇒
    rw [ENat.coe_zero, skeletonLT_zero_eq_base, HEP', Subtype.coe_preimage_self]
    exact HEP_self
  | succ n n_ih ⇒
    apply HEP_trans (skeletonLT X (n + 1)).carrier  (skeletonLT X n) C
      (skeletonLT_mono le_self_add)  ?_ (Subcomplex.base_subset
      (skeletonLT X ↑n)) n_ih
    exact HEP'_skeleton X n
```

The base case is the lemma `HEP_self`, which we proved at the very beginning, just after defining the HEP. Getting there requires rewriting with the fact that the base space is the same as the (-1)-skeleton. For the induction step we combine the induction hypothesis with the HEP for two

consecutive skeleta via the transitivity lemma `HEP_trans`.
The corresponding statement for `skeleton` then follows from the one for `skeletonLT`, since `skeleton X m` is by definition `skeletonLT X (m+1)`.

```
lemma HEP_skel_base (m : ℕ) : HEP' (skeleton X m) C := by
  rw [skeleton.congr_simp X m m rfl]
  exact HEP_skelLT_base X (m + 1)
```

It remains to show that for a finite dimensional complex there is an $m \in \mathbb{N}$ with `X = skeletonLT X m`. This is the content of the following lemma.

```
lemma RelCWFinite_eq_SkelLT (hX : FiniteDimensional X) : ∃ (m : ℕ),
  X = skeletonLT X m := by ...
```

The idea of the proof is simple. Being finite dimensional means that the index types of the cells are empty from some dimension m on. The complex itself can be written as the union of the base space with all open cells, and in `RelCW_eq_UnionSkelLtCell` we show that it can equally be written as the union of `skeletonLT X m` with the open cells of dimension at least m. For the m given by finite dimensionality the second part of this union is empty, which gives the claim.

Putting the two steps together, the final proof is short.

```
lemma HEP_RelCWFinite (hX : FiniteDimensional X) : HEP' X C := by
  obtain ⟨m, hm⟩ := RelCWFinite_eq_SkelLT X hX
  rw[hm]
  exact HEP_skelLT_base X m
```

This completes the three steps announced at the beginning: the HEP for a ball and a cube relative to their boundary, for two consecutive skeleta, and finally for a finite dimensional relative CW-complex relative to its base space.

5 Usage of AI

The use of AI as an aid for this thesis was agreed upon in advance with my supervisor, Floris van Doorn, on the condition that it be made transparent here and that any code not written by myself be marked as such in the formalisation. My methods, and with them the usage of AI, differed between the formalisation project and the writing of the text. In both cases, whenever AI was used, it was the latest version of Anthropic's Claude Opus model (as of August 2026: Opus 5).

All submitted Lean code was written by me, except two short definitions together with two related simp lemmas in the file `HEP_skeleton.lean`. These approximately thirty lines are marked accordingly in the formalisation. They were tools that could have been part of Mathlib. They are not specific to the constructions of this project.

The rest of the formalisation is my own work, but it would in some parts look different without the use of AI. I frequently asked Claude for help when a proof got stuck or an error message was difficult to interpret. One example is the tactic `subst`, which simplified the proof of `SkeletonProjectionIsQuotientMap` considerably and later allowed me to shorten several earlier drafts by a noticeable number of lines. Another is the observation that consecutive `exact` statements can be combined into a single `exacts [...]`.

For larger design questions I found AI less helpful. The pushout construction for the retraction of consecutive skeleta used in the lecture notes, for instance, did not match the formalisation of CW-complexes in Mathlib that I wanted to work with. I asked Claude for help at this point. Instead of working within the subset setting of the formalised CW-complexes, which is the solution presented earlier, it suggested a longer and more technical way that did not seem realisable to me. The biweekly meetings with Floris van Doorn were the place where such decisions were actually resolved.

I wrote the text of this thesis in full myself in a first step. In a second pass I went through it paragraph by paragraph and improved the language and corrected some inaccuracies, with the help of Claude. For the comments in my formalisation the procedure was partly the same. In other places Claude added a comment based on the code, which I then revised and adapted. Every reformulation was reviewed before I adopted it, and I discarded those that shifted the meaning of what I had written. AI was also used for parts of the LaTeX setup of this document, among them the configuration for typesetting Lean code.

6 Conclusion

In this thesis we formalised the homotopy extension property and developed tools to prove it for pairs of spaces, most importantly the retraction criterion and the preservation of the HEP under partial homeomorphisms. Building on these, we proved the property for balls and cubes relative to their boundary, for two consecutive skeleta of a relative CW-complex, and finally for finite dimensional relative CW-complexes.

One result deserves to be mentioned separately. The lemma `SkeletonProjectionIsQuotientMap` states that every skeleton of a relative CW-complex is a quotient of the disjoint union of its base space with the closed cubes of the cells it contains. Mathlib describes the topology of a complex through its closed sets rather than defining the complex as a pushout, so this description is not available there. It adds a second point of view on the existing definition, and with it the possibility of constructing a map out of a skeleton by prescribing it on the base space and on each cube separately. This is precisely how the retraction `r` of Section 4.6.2 is built.

The question that recurred throughout the formalisation was whether to work in an ambient type. It appears in the two definitionally equal versions of the homotopy extension property, `HEP` and `HEP'`. `HEP` is applied to a type together with a subset of it, whereas for `HEP'` both spaces are subsets of the same ambient type. Both are used in this project, and `HEP'` was the more convenient of the two to work with. Whether the further modification sketched in Section 4.2, spelling `HEP'` out with maps defined on the ambient space, would be better still, we cannot judge from the experience gathered here. What we did see is that working in an ambient type is not without cost. Statements become longer since, for example, `ContinuousOn` obligations are more work to prove than plain `Continuous` ones.
Both versions of the retraction criterion proved worthwhile, and we would recommend establishing both for whichever definition of the HEP one chooses. In `retraction_criterion_closed` the retraction lives on the product with the larger space of the pair, in `retraction_criterion_closed'` on the product with the ambient type. Each was clearly preferable at the place where we used it, as discussed in Section 4.3.

In this project, and we suspect not only here, the effort a step required correlates poorly with its mathematical difficulty. `PartialHomeomorph_HEP'` and its application to obtain `HEP_cube_boundary'` are mathematically unremarkable, yet obtaining the right partial homeomorphism and checking all the required properties took a notable amount of work. The proof for two consecutive skeleta consists of a large number of individual steps, none of them surprising, but all of them to be carried out. The induction showing that finite dimensional CW-complexes have the HEP, on the other hand, was short and caused no difficulties at all, which we take as evidence that the preparatory work was cut to the right shape.

In Section 4.3 we ran into a statement that turned out to say something slightly different from what was intended. In `retraction_criterion_closed'` the hypothesis `IsClosed A` asks for closedness in the ambient type rather than in the larger space of the pair, which is strictly stronger. The same hypothesis is inherited by `PartialHomeomorph_HEP'`, which is proved through the criterion. One could restate the assumption in the subspace topology, but in practice it slightly simplified the applications rather than causing any trouble, since every closedness fact we needed holds in the ambient space anyway.

The most obvious point to continue this project would be to generalise the HEP to all relative CW-complexes. For the inductive construction of the retraction maps in Theorem 2.25 we would recommend working with retractions on the ambient space, which compose well without any coercions. This would give a further application of `retraction_criterion_closed'`, where we benefit from the formulation using the ambient type. That result could then be used towards proving the theorems mentioned in the introduction, such as Whitehead's theorem Theorem 1.2. A second direction is to formalise the retraction criterion without the closedness assumption, following the argument of [Str68] given in the appendix of [Hat02]. Since the assumption appears in a number of other results, dropping it there would remove it also from `PartialHomeomorph_HEP'` and everything else proved in the same way.

I want to end on a personal note. Formalising these arguments made me understand them far more deeply and precisely than following them in a lecture had. The retraction criterion is my clearest example. Only while formalising it did I notice that the closedness assumption in the argument we follow is needed in one of the two directions and not in the other. This is something I had not been aware of before, and it is why the corresponding lemma is split in two.

7 German Summary

In dieser Arbeit formalisieren wir die Homotopieerweiterungseigenschaft (HEE) im Beweisassistenten Lean. Ziel ist der Beweis, dass relative CW-Komplexe diese Eigenschaft besitzen. Dieses bekannte Resultat der algebraischen Topologie ist unter anderem ein wesentlicher Schritt im Beweis des Satzes von Whitehead.

Das erste Kapitel gibt einen Überblick über die Arbeit und motiviert das Projekt von zwei Seiten: warum die HEE mathematisch interessant ist und was durch ihre Formalisierung gewonnen wird. Im zweiten Kapitel wird die Mathematik hinter der Formalisierung ausführlich dargestellt. Nach der Definition der HEE folgt das Retraktionskriterium, mit dem wir zeigen können, dass ein Paar topologischer Räume die HEE hat. Außerdem wird gezeigt, dass die HEE unter Homöomorphismen erhalten bleibt und dass aus der HEE für (X, A) und für (A, B) auch die HEE für (X, B) folgt.

Im dritten Kapitel wird Lean vorgestellt und die für diese Arbeit wichtigsten Konzepte werden zusammengefasst. Die mathematische Bibliothek Mathlib wird eingeführt und die dortige Formalisierung von CW-Komplexen wird in den Aspekten erläutert, die für die eigene Formalisierung relevant sind.

Im vierten Kapitel wird die Formalisierung anhand von Auszügen aus dem Lean-Code erklärt. Es wird auf Designentscheidungen eingegangen und das Vorgehen begründet. Das Kapitel folgt der Struktur der Formalisierung, welche vier Dateien umfasst. Die erste enthält die Definition und allgemeine Aussagen über die HEE. Die Dateien zwei bis vier beweisen schrittweise die HEE für verschiedene Paare. Diese Schritte stammen aus dem Beweis der HEE für relative CW-Komplexe. Zuerst wird die Eigenschaft für Bälle und Würfel relativ zu ihren Rändern gezeigt, anschließend für zwei aufeinanderfolgende Skeleta eines relativen CW-Komplexes. Die letzte Datei zeigt mithilfe dieser Ergebnisse die HEE für endlichdimensionale relative CW-Komplexe. Der allgemeine Fall relativer CW-Komplexe ist nicht abgedeckt.

Anschließend wird die Nutzung von KI für diese Arbeit erläutert. Zuletzt werden die wichtigsten Ergebnisse der Arbeit zusammengefasst und in einem Ausblick wird beschrieben, wie das Projekt fortgesetzt werden kann.

References

[Avi+24] Jeremy Avigad et al. *Theorem Proving in Lean 4*. 2024. URL: https://leanprover.github.io/theorem_proving_in_lean4/ (visited on 07/02/2026).

[Boa16] J. Michael Boardman. *The Homotopy Extension Property*. 2016. URL: https://math.jhu.edu/~jmb/note/hep.pdf (visited on 07/13/2026).

[Eng89] Ryszard Engelking. *General topology*. eng. Rev. and completed ed. Sigma series in pure mathematics 6. Berlin: Heldermann, 1989. ISBN: 3885380064.

[FRO] Lean FRO. *FRO - About us*. URL: https://lean-lang.org/fro/about/ (visited on 07/02/2026).

[Hat02] Allen Hatcher. *Algebraic Topology*. Cited according to the electronic version, which contains material not included in any printing. Cambridge: Cambridge University Press, 2002. ISBN: 0-521-79540-0. URL: https://pi.math.cornell.edu/~hatcher/AT/AT.pdf (visited on 08/24/2026).

[Kon25] Alex Kontorovich. *The Shape of Math To Come*. 2025. arXiv: 2510.15924 [math.HO]. URL: https://arxiv.org/abs/2510.15924.

[Mat] Mathlib Community. *Mathlib Statistics*. URL: https://leanprover-community.github.io/mathlib_stats.html (visited on 07/02/2026).

[MLS25] Alex Meiburg, Leonardo A. Lessa, and Rodolfo R. Soldati. *A Formalization of the Generalized Quantum Stein's Lemma in Lean*. 2025. arXiv: 2510.08672 [quant-ph]. URL: https://arxiv.org/abs/2510.08672.

[Mou+15] Leonardo de Moura et al. "The Lean Theorem Prover (System Description)". eng. In: *Automated Deduction - CADE-25* (2015), pp. 378–388. ISSN: 0302-9743.

[Sch24] Hannah Scholz. "Formalisation of CW-complexes". Bachelor's Thesis Mathematics. Rheinischen Friedrich-Wilhelms-Universität Bonn, 2024. URL: https://florisvandoorn.com/theses/HannahScholz.pdf.

[Str11] Jeffrey Strom. *Modern classical homotopy theory*. eng. Graduate studies in mathematics 127. Providence, RI: American Math. Soc., 2011. ISBN: 9780821852866.

[Str68] Arne Strom. "Note on Cofibrations II". eng. In: *Mathematica scandinavica* 22 (1968), pp. 130–142. ISSN: 0025-5521.

[SZ88] Ralph Stöcker and Heiner Zieschang. *Algebraische Topologie : eine Einführung ; mit zahlreichen Bildern, Beispielen und Übungsaufgaben*. ger. Mathematische Leitfäden. Stuttgart: Teubner, 1988. ISBN: 3519022265.

[The] The Mathlib Community. *Mathlib: The Lean Mathematical Library*. URL: https://github.com/leanprover-community/mathlib4 (visited on 08/24/2026).

[The20] The mathlib Community. "The Lean Mathematical Library". In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. CPP 2020. New Orleans, LA, USA: ACM, Jan. 2020. DOI: 10.1145/3372885.3373824. URL: https://doi.org/10.1145/3372885.3373824.

[Tom08] Tammo Tom Dieck. *Algebraic topology*. eng. EMS textbooks in mathematics. Zürich: European Math. Soc., 2008. ISBN: 9783037190487.

[Wal] Friedhelm Waldhausen. *Algebraische Topologie*. URL: https://www.math.uni-bielefeld.de/~fw/ (visited on 08/14/2026).

[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.