Synthetic Cohomology Theory in Cubical Agda

Guillaume Brunerie
Independent researcher, Sweden

Axel Ljungström
Department of Mathematics, Stockholm University, Sweden

Anders Mörtberg
Department of Mathematics, Stockholm University, Sweden

Abstract

This paper discusses the formalization of synthetic cohomology theory in a cubical extension of `Agda` which natively supports univalence and higher inductive types. This enables significant simplifications of many proofs from Homotopy Type Theory and Univalent Foundations as steps that used to require long calculations now hold simply by computation. To this end, we give a new group structure for cohomology with $\mathbb{Z}$-coefficients, optimized for efficient computations. We also invent an optimized definition of the cup product which allows us to give the first complete formalization of the axioms needed to turn the $\mathbb{Z}$-cohomology groups into a graded commutative ring. Using this, we characterize the cohomology groups of the spheres, torus, Klein bottle and real/complex projective planes. As all proofs are constructive we can then use `Cubical Agda` to distinguish between spaces by computation.

2012 ACM Subject Classification Theory of computation → Constructive mathematics; Theory of computation → Type theory

Keywords and phrases Synthetic Homotopy Theory, Cohomology Theory, Cubical Agda

1 Introduction

Homotopy Type Theory and Univalent Foundations (HoTT/UF) [38] extends Martin-Löf type theory [30] with Voevodsky's univalence axiom [41] and higher inductive types (HITs). This is based on a close correspondence between types and topological spaces represented as Kan simplicial sets [24]. With this interpretation, points in spaces correspond to elements of types, while paths and homotopies correspond to identity types between these elements [3]. This enables homotopy theory to be developed *synthetically* using type theory. Many classical results from homotopy theory have been formalized in HoTT/UF this way: the definition of the Hopf fibration [38], the Blakers-Massey theorem [22], the Seifert-van Kampen theorem [23] and the Serre spectral sequence [39], among others. Using these results, many homotopy groups of spaces—represented as types—have been characterized. However, just like in classical algebraic topology, these groups tend to be complicated to work with. Because of this, other topological invariants like cohomology have been invented.

Informally, the cohomology groups $H^n(X)$ of a space X describe its n-dimensional holes. For instance, the n-dimensional hole in the n-sphere $\mathbb{S}^n$ corresponds to $H^n(\mathbb{S}^n) \simeq \mathbb{Z}$. These holes constitute a topological invariant, making cohomology a powerful technique for establishing which spaces cannot be homotopy equivalent. The usual formulation of singular cohomology using cochain complexes relies on taking the underlying set of topological spaces when defining the singular cochains [19]. This operation is *not* invariant under homotopy equivalence, which makes it impossible to use when formalizing cohomology synthetically. Luckily, there is an alternative definition of cohomology using Eilenberg-MacLane spaces which is homotopy invariant [26]. This was initially studied at the IAS special year on HoTT/UF in 2012–2013 [33] and has since been used to develop the Eilenberg-Steenrod axioms [11] and cellular cohomology [8]. This paper builds on this prior work, but uses `Cubical Agda`—a recent *cubical* extension of the dependently typed programming language `Agda` [35].

The `Cubical Agda` system is based on a variation of cubical type theory formulated by Coquand et al. [14]. These type theories can be seen as refinements of HoTT/UF where the homotopical intuitions are taken very literally and made part of the theory. Instead of relying on the inductively defined identity type [29] to define paths and homotopies, a primitive interval type I is added. Paths and homotopies are then represented as functions out of I, just like in traditional topology. This has some benefits compared to HoTT/UF. First, many proofs become simpler. For instance, function extensionality becomes trivial to prove, as opposed to in HoTT/UF where it either has to be postulated or derived from the univalence axiom [42]. Second, it gives computational meaning to HoTT/UF, which makes it possible to use the system to do computations using univalence and HITs. Finally, it makes it possible to formulate a general schema for HITs where the eliminators compute definitionally for higher constructors [12, 15]. This is still an open problem for HoTT/UF, and HITs have to be added axiomatically, which leads to bureaucratic transports that complicate proofs.

Mörtberg and Pujet explored practical implications of formalizing synthetic homotopy theory in `Cubical Agda` in [31]. This work provided empirical evidence that formalizing synthetic homotopy theory in cubical type theory can lead to significant simplifications of the corresponding formal HoTT/UF proofs. For instance, the proof of the 3×3 lemma for pushouts was shortened from 3000 lines of code (LOC) in `HoTT-Agda` [7] to only 200 in `Cubical Agda`. Another proof that becomes substantially shorter is the proof that the torus is equivalent to the product of two circles. This elementary result in topology turned out to have a surprisingly non-trivial proof in HoTT/UF because of the lack of definitional computation rules for higher constructors [25, 34]. With the additional computation rules of `Cubical Agda`, this proof is now trivial [40, Sect. 2.4.1].

The present paper is a natural continuation of this prior work and the two main goals are to *characterize* $\mathbb{Z}$-cohomology groups of types and to *compute* using these groups. In classical algebraic topology, *characterize* and *compute* are often used interchangeably when discussing cohomology. We are careful to distinguish these two notions. When *characterizing* a cohomology group of some type, we prove that it is isomorphic to another group. As all of our proofs are constructive, we can then use `Cubical Agda` to actually *compute* with this isomorphism. Having the possibility of doing proofs simply by computation is one of the most appealing aspects of developing synthetic homotopy theory cubically. As this is not possible with pen and paper proofs, or even with many formalized proofs in HoTT/UF, one often has to resort to doing long calculations by hand. If proofs instead can be carried out using a computer, many of these long calculations become obsolete. This is a reason why many proofs from synthetic homotopy theory are substantially shorter in `Cubical Agda`. However, not everything has successfully been possible to reduce to computations. A famous example is the *Brunerie number*. This is a synthetic definition of a number $n : \mathbb{Z}$ such that $\pi_4(\mathbb{S}^3) = \mathbb{Z}/n\mathbb{Z}$. Brunerie proved in his PhD thesis [5] that $n = \pm 2$, but even though this is a constructive definition, it has thus far proved infeasible to compute using `Cubical Agda`, despite considerable efforts. In this paper, we construct a similar number, also inspired by [5], using the multiplicative structure on $H^n(\mathbb{C}P^2)$. This number was proved to be ± 1 using sophisticated techniques in [5, Chapter 6], but we have thus far been unable to verify this purely by computation. However, as this number is substantially simpler than the Brunerie number, it provides a new challenge for constructive implementations of HoTT/UF which should be more feasible.

Contributions: the main novel result of the paper is the first formalization of the graded commutative ring axioms for $\mathbb{Z}$-cohomology in HoTT/UF (Section 4). To this end, we first develop $\mathbb{Z}$-cohomology groups (Section 3). The definitions are inspired by [5], but the additive structure is new and optimized for efficient computations. The definition of the cup product is also new and provides significant simplifications compared to related proofs in `HoTT-Agda` [4]. We also characterize the cohomology groups of various types (Section 5); for instance, we give the first synthetic characterizations of the cohomology groups of the Klein bottle and real projective plane. In order to characterize $H^n(\mathbb{C}P^2)$, we verify that our definition of cohomology satisfies the Eilenberg-Steenrod axioms for cohomology theories and construct the Mayer-Vietoris sequence (Appendix B). We finally reap the fruits of our constructive definitions in Section 6 where we prove that $\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1$ and the torus are not equivalent by computing with `Cubical Agda`.

All results in the paper have been formalized in `Cubical Agda`, but we also provide written up proofs, with details in Appendix A. Much of the code in the paper is literal `Cubical Agda` code, but we have taken some liberties when typesetting, to closer resemble standard mathematical notations. In order to clarify the connections between the paper and formalization, we provide a summary file: **https://github.com/agda/cubical/blob/master/Cubical/Papers/ZCohomology.agda**. This file typechecks with the **-safe** flag, which ensures that there are no postulates or unfinished goals.

2 Homotopy Type Theory in Cubical Agda

The `Agda` system [35] is a dependently typed programming language in which both programs and proofs can be written using the same syntax. Dependent function types (Π-types) are written $(x : A) \to B$ while non-dependent function types are written $A \to B$. Implicit arguments to functions are written using curly braces $\{x : A\} \to B$ and function application is written using juxtaposition, so $f\,x$ instead of $f(x)$. Universes are written $\mathsf{Type}\,\ell$, where ℓ is a *universe level*. In order to ease notation, we omit universe levels in this paper. Readers familiar with `Agda` will also notice that we rename Set to Type. `Agda` supports many features of modern proof assistants and has recently been extended with an experimental *cubical* mode. The goal of this section is to introduce notions from HoTT/UF (including their formalizations in `Cubical Agda`) which the rest of the paper relies on. Due to space constraints, we omit many technical details and refer curious readers to the paper of Vezzosi et al. [40] for a comprehensive technical treatment of the features of `Cubical Agda`.

2.1 Important notions in `Cubical Agda`

The first addition to make `Agda` *cubical* is an interval type I with endpoints $\mathsf{i0}$ and $\mathsf{i1}$. This corresponds to the real interval $[0,1] \subset \mathbb{R}$. However, in `Cubical Agda`, this is a purely formal object. A variable $i : \mathsf{I}$ represents a point varying continuously between the endpoints. There are three primitive operations on I: *minimum*/*maximum* ($\_\wedge\_$, $\_\vee\_$: $\mathsf{I} \to \mathsf{I} \to \mathsf{I}$) and *reversal* ($\sim\_$: $\mathsf{I} \to \mathsf{I}$). A function $\mathsf{I} \to \mathsf{Type}$ represents a line between two types. By iterating this, we obtain squares, cubes and hypercubes of types making `Agda` inherently *cubical*. In order to specify the endpoints of a line, we use *path types*:

$$\mathsf{PathP} : (A : \mathsf{I} \to \mathsf{Type}) \to A\ \mathsf{i0} \to A\ \mathsf{i1} \to \mathsf{Type}$$

As paths are functions, they are introduced as $\lambda\, i \to t$: $\mathsf{PathP}\ A\ t[\mathsf{i0}\,/\,i]\ t[\mathsf{i1}\,/\,i]$. Given p : $\mathsf{PathP}\ A\ a_0\ a_1$, we can apply it to $r : \mathsf{I}$ and obtain $p\ r : A\ r$. Also, we always have that $p\ \mathsf{i0}$ reduces to a_0 and $p\ \mathsf{i1}$ reduces to a_1. The PathP types should be thought of as representing heterogeneous equalities since the two endpoints are in different types; this is similar to dependent paths in HoTT/UF [38, Section 6.2]. Given A : Type, we define the type of non-dependent paths in A using PathP as follows:

```
_≡_ : A → A → Type
_≡_ x y = PathP (λ _ → A) x y
```

Representing equalities as paths allows us to directly reason about equality. For instance, the constant path $\lambda\ i \to x$ represents a proof of reflexivity refl : $\{x : A\} \to x \equiv x$. We can also directly apply a function to a path in order to prove that dependent functions respect path-equality, as shown in the definition of cong below:

```
cong : {B : A → Type} {x y : A} (f : (x : A) → B x) (p : x ≡ y) → PathP (λ i → B (p i)) (f x) (f y)
cong f p = λ i → f (p i)
```

We write cong$_2$ for the binary version of cong; its proof is equally direct. These functions satisfy the standard property that refl gets mapped to refl. They are also definitionally functorial. The latter is an important difference to the corresponding operations defined using path induction which only satisfy the functoriality equations up to a path. Path types also let us prove new things that are not provable in standard `Agda`, e.g. function extensionality:

```
funExt : {B : A → Type} {f g : (x : A) → B x} → ((x : A) → f x ≡ g x) → f ≡ g
funExt p i x = p x i
```

One of the key operations of type theoretic equality is *transport*: given an path between types, we get a function between these types. In `Cubical Agda`, this is defined using another primitive called transp. However, for this paper, the cubical transport function suffices:

```
transport : {A B : Type} → A ≡ B → A → B
transport p a = transp (λ i → p i) i0 a
```

The substitution principle, called "transport" in HoTT/UF, is an instance of cubical transport:

```
subst : (B : A → Type) {x y : A} → x ≡ y → B x → B y
subst B p b = transport (λ i → B (p i)) b
```

This function invokes transport with a proof that the family B respects the equality p. By combining transport and _∧_, we can define the induction principle for paths. However, an important difference between path types in `Cubical Agda` and HoTT/UF is that _≡_ does not behave like an inductive type. In particular, the cubical path induction principle does not definitionally satisfy the computation rule when applied to refl. Nevertheless, we can still prove that this rule holds up to a path. This is a subtle, but important, difference between cubical type theory and HoTT/UF. Readers familiar with HoTT/UF might be worried that the failure of this equality to hold definitionally complicates many proofs. However, in our experience, this is rarely the case, as many proofs that require path induction in HoTT/UF can be proved more directly using cubical primitives.

`Cubical Agda` also has a primitive operation hcomp for composing paths and, more generally, for composing higher dimensional cubes. An important special case is binary composition of paths _·_ : $x \equiv y \to y \equiv z \to x \equiv z$. By composing paths and higher cubes using hcomp, we can reason about paths in a very direct way, avoiding path induction.

2.2 Important concepts from HoTT/UF in `Cubical Agda`

Pointed types and functions will play an important role in this paper. Formally, a pointed type is a pair $(A\,,*_A)$ where A is a type with $*_A : A$. We write Type$_*$ for the universe of pointed types. Given $A, B : \mathsf{Type}_*$, a pointed function is a pair $(f , p) : A \to_* B$, where $f : A \to B$ and $p : f\ *_A\ \equiv *_B$. We often leave $*_A$ and p implicit and write simply $A : \mathsf{Type}_*$ and $f : A \to_* B$. We also sometimes just write A for the underlying type of $A : \mathsf{Type}_*$.

Most HITs in [38] can be defined directly using the general schema of `Cubical Agda`. For example, the circle and suspension HITs can be written as:

```
data S¹ : Type where            data Susp (A : Type) : Type where
  base : S¹                       north : Susp A
  loop : base ≡ base              south : Susp A
                                  merid : (a : A) → north ≡ south
```

Functions out of HITs are written using pattern-matching equations, just like regular `Agda` functions. When typechecking the cases for path constructors, `Cubical Agda` checks that the endpoints of what the user writes match up. We could directly define specific higher spheres as HITs with a base point and a constructor for iterated paths. However, the following definition is often easier to work with, as one can reason inductively about it:

▶ **Definition 1** ($\mathbb{S}^n$). *The n-spheres are pointed types defined by recursion:*

$$\mathbb{S}^n = \begin{cases} (\mathsf{Bool} , \mathsf{true}) & \text{if } n = 0 \\ (\mathbb{S}^1 , \mathsf{base}) & \text{if } n = 1 \\ (\mathsf{Susp}\ \mathbb{S}^{n-1} , \mathsf{north}) & \text{if } n \geq 2 \end{cases}$$

We could equivalently have defined $\mathbb{S}^1 = (\mathsf{Susp}\ \mathsf{Bool} , \mathsf{north})$, but in our experience, the base/loop-construction is often easier to work with and gives faster computations.

Consistent with the intuition that types correspond to topological spaces (up to homotopy equivalence), we may consider loop spaces of pointed types.

▶ **Definition 2** (Loop spaces). *Given a pointed type $A : \mathsf{Type}_*$, we define its loop space as the pointed type $\Omega\, A = (*_A \equiv\ *_A\ , \mathsf{refl})$. For $n : \mathbb{N}$, we let $\Omega^{n+1}\ A = \Omega\,(\Omega^n\ A)$.*

As an example of a non-trivial result which is proved using path induction in HoTT/UF, but which can be proved very concisely in `Cubical Agda`, consider the Eckmann-Hilton argument. It says that path composition in higher loop spaces is commutative and can be proved using a single transport with the unit laws for _·_ and some interval operations.

```
EH : {n : ℕ} (p q : Ω^ (2 + n) A) → p · q ≡ q · p
EH p q = transport (λ i → (λ j → rUnit (p j) i) · (λ j → lUnit (q j) i)
                        ≡ (λ j → lUnit (q j) i) · (λ j → rUnit (p j) i))
                   (λ i → (λ j → p (j ∧ ∼ i) · q (j ∧ i)) · (λ j → p (∼ i ∨ j) · q (i ∨ j)))
```

A type A is not uniquely determined by its points—also (higher) paths over A have to be taken into account. However, for some types, these paths become trivial at some point. We define what this means formally as follows.

▶ **Definition 3** (n-types). *Given $n \geq -2$, a type A is:*

- *a (-2)-type if A is contractible (i.e. A is pointed by a unique point).*
- *an $(n + 1)$-type if for all $x, y : A$, $x \equiv y$ is an n-type.*

We write n-Type for the universe of n-types (at some level ℓ).

Equivalently, we could have said that, for $n \geq -1$, A is an n-type if $\Omega^{n+1}A$ is contractible for any choice of base point $a : A$. We follow HoTT/UF terminology and refer to (-1)-types as *propositions* and 0-types as *sets*. A type is a proposition iff all of its elements are path-equal.

Sometimes we are only interested in the structure of a type A and its paths up to a certain level n. That is, we want to turn A into an n-type while preserving the structure of A for levels less than or equal to n. This can be achieved using the n-truncation HITs $\| A \|_n$. Just like for $\mathbb{S}^n$, these are easily defined in `Cubical Agda` for fixed n, but for general $n \geq -2$ we rely on the "hub and spoke" construction [38, Section 7.3].[1] This construction introduces an injection $| \_ | : A \to \| A \|_n$ and path constructors hub and spoke ensuring that any map $\mathbb{S}^{n+1} \to \| A \|_n$ is constant (thus contracting $\Omega^{n+1} \| A \|_n$). Using pattern-matching, we can define the usual elimination principle which says: given $B : \| A \|_n \to n\text{-Type}$, in order to construct an element of type $B\ x$, we may assume that x is of the form $| a |$ for some $a : A$. This extends to paths $p : | x | \equiv | y |$ in $\| A \|_{n+1}$. Suppose we have $B : | x | \equiv | y | \to n\text{-Type}$ and want to construct $B\ p$. The elimination principle tells us that it suffices to do so when $p = \mathsf{cong}\ | \_ |\ q$ for $q : x \equiv y$ in A. This is motivated by [38, Theorem 7.3.12].

Truncations allow us to talk about how connected a type is.

▶ **Definition 4** (Connectedness). *A type A is n-connected if $\| A \|_n$ is contractible.*

Connectedness expresses in particular that $| x | \equiv | y |$ holds in $\| A \|_n$ for all $x, y : A$ of an n-connected type A. This enables applications of the induction principle for truncated path spaces discussed above. Most types in this paper are 0-connected. For such types, we can assume that $x \equiv y$ holds for $x, y : A$ whenever we are proving a family of propositions.

Another important class of HITs are pushouts. These correspond to homotopy pushouts in topology. Given $f : A \to B$ and $g : A \to C$, the pushout of the span $B \xleftarrow{f} A \xrightarrow{g} C$ is:

```
data Pushout (f : A → B) (g : A → C) : Type where
  inl : B → Pushout f g
  inr : C → Pushout f g
  push : (a : A) → inl (f a) ≡ inr (g a)
```

Many types that we have seen so far can be defined as pushouts. For instance, $\mathsf{Susp}\ A$ is equivalent to the pushout of the span $\mathbb{1} \leftarrow A \to \mathbb{1}$. Another example is wedge sums:

▶ **Definition 5** (Wedge sums). *Given pointed types A and B, the* wedge sum $A \vee B$ *is the pushout of the span $A \xleftarrow{\lambda\ x\ \to\ *_A} \mathbb{1} \xrightarrow{\lambda\ x\ \to\ *_B} B$. This is pointed by* $\mathsf{inl}\ *_A$.

2.3 Univalence

One of the most important notions in HoTT/UF is Voevodsky's univalence axiom [41]. Informally, this postulates that for all types A and B, there is a term

$$\mathsf{univalence} : (A \simeq B) \simeq (A \equiv B)$$

Here, $A \simeq B$ is the type of functions $e : A \to B$ equipped with a proof that the fiber/preimage of e is contractible at every $x : B$ [38, Chapter 4.4]. This axiom is a provable theorem in `Cubical Agda` using the Glue types of [14, Section 6]. This gives a function $\mathsf{ua} : A \simeq B \to A \equiv B$ which converts equivalences to paths. Transporting along a path constructed using ua applies the function e of the underlying equivalence.

Equivalences $A \simeq B$ are often constructed by exhibiting functions $f : A \to B$ and $g : B \to A$ together with proofs that they cancel. Such a quadruple is referred to as a

[1] For $n = -2$ this construction fails. In this case, simply let $\| A \|_{-2} = \mathbb{1}$ where $\mathbb{1}$ is the unit type.

quasi-equivalence in [38]. It is a corollary of [38, Theorem 4.4.5] that all quasi-equivalences can be promoted to equivalences. This fact is used throughout the formalization and paper.

An important consequence of univalence is that it also applies to *structured* types. A structure on types is simply a function $S : \mathsf{Type} \to \mathsf{Type}$. By taking the dependent sum of this, one obtains types with S-structures as pairs $(A\,,s) : \Sigma_{A:\mathsf{Type}}\,(S\ A)$. One example is the type of groups. This is defined as $(G\,,\mathsf{isGroup}\ G)$, where $\mathsf{isGroup}\ G$ is a structure which consists of proofs that G is a set, is pointed by some $0_G : G$, admits a binary operation $+_G$, and satisfies the usual group laws. In [2], a notion of *univalent* structure and structure preserving isomorphisms $\cong$, for which it is direct to prove that ua induces a function $\mathsf{sip} : A \cong B \to A \equiv B$, are introduced in `Cubical Agda`. This is one way to formalize the informal *Structure Identity Principle* (SIP) from HoTT/UF [38, Section 9.8]. One can show that $\mathsf{isGroup}$ is a univalent structure and that equivalences $e : G \simeq H$ sending $+_G$ to $+_H$ preserve this structure. In other words: sip implies that isomorphic groups are path-equal.

3 $\mathbb{Z}$-cohomology in Cubical Agda

In classical mathematics, the n:th cohomology group with coefficients in an abelian group G of a CW-complex X may be characterized as the group of homotopy classes of functions $X \to K(G,n)$. Here, $K(G,n)$ denotes the n:th *Eilenberg-MacLane space* of G. That is, $K(G,n)$ is the unique space with a single non-trivial homotopy group isomorphic to G, i.e. $\pi_n(K(G,n)) \cong G$ and $\pi_m(K(G,n)) \cong \mathbb{1}$ for $m \neq n$. While this is a theorem in classical mathematics, we take it as our definition of the n:th cohomology group of a type A:

$$H^n(A;G) = \|\, A \to K(G,n) \,\|_0$$

This type inherits the group structure from $K(G,n)$ and the goal of this section is to define this explicitly when $G = \mathbb{Z}$. The group structure which we will define here differs from previous variations in that it is optimized for efficient computations.

3.1 Eilenberg-MacLane spaces

The family of spaces $K(G,n)$ was constructed as a HIT and proved to be an n-truncated and $(n-1)$-connected pointed type by Licata and Finster [26]. In this paper, we focus on the case $G = \mathbb{Z}$ and define this special case following Brunerie [5, Def. 5.1.1]:

▶ **Definition 6.** *The n:th Eilenberg-MacLane space of $\mathbb{Z}$, written K_n, is a pointed type:*

$$\mathsf{K}_n = \begin{cases} (\mathbb{Z}\,,0) & \text{if } n = 0 \\ (\|\,\mathbb{S}^n\,\|_n\,,|\ \ast_{\mathbb{S}^n}\ |) & \text{if } n \geq 1 \end{cases}$$

We write $H^n(A)$ for $H^n(A;\mathbb{Z})$ with K_n for $K(\mathbb{Z},n)$. The type K_n is clearly n-truncated and the fact that it is $(n-1)$-connected follows from the following proposition.

▶ **Proposition 7.** *$\mathbb{S}^n$ is $(n-1)$-connected for $n : \mathbb{N}$.*

Proof. By the definition of $(n-1)$-truncation, the map $|\_| : \mathbb{S}^n \to \|\,\mathbb{S}^n\,\|_{n-1}$ is constant. Hence, $\|\,\mathbb{S}^n\,\|_{n-1}$ has a trivial constructor and must be contractible. ◀

Note that, in particular, K_n is 0-connected for $n > 0$; it is an easy lemma that any m-connected type is also k-connected for $k < m$. Alternatively, one may prove 0-connectedness of K_n directly by truncation elimination and sphere elimination.

The above proof is much more direct than the one in [5, Prop. 2.4.2] which relies on general results about connectedness of pushouts. The reason we prefer this more direct, but less general proof, is that it computes much faster. The problem seems to be that the general theory of connectedness heavily uses univalence. In particular, it relies on repeated use of [38, Thm. 7.3.12] which says that the type of paths $|\, x \,| \equiv |\, y \,|$ over $\|\, A \,\|_{n+1}$ is equivalent to the type of truncated paths $\|\, x \equiv y \,\|_n$.

A more substantial deviation from [5] is in the definition of the group structure on K_n. This is defined in [5, Prop. 5.1.4] using $\mathsf{K}_n \simeq \Omega\, \mathsf{K}_{n+1}$ which itself is proved using the Hopf fibration [38, Section 8.5] when $n = 1$ and the Freudenthal suspension theorem [38, Section 8.6] when $n \geq 2$. This gives rather indirect definitions of addition and negation on K_n by going through $\Omega\, \mathsf{K}_{n+1}$. It turns out that these indirect definitions lead to slow computations [28]. To circumvent this, we give a direct definition of the group structure on K_n which in turn gives a direct proof that $\mathsf{K}_n \simeq \Omega\, \mathsf{K}_{n+1}$ inspired by the proof that $\Omega\, \mathbb{S}^1 \simeq \mathbb{Z}$ of Licata and Shulman [27]. The strategy of first defining the group structure on K_n to then prove that $\Omega\, \mathsf{K}_{n+1} \simeq \mathsf{K}_n$ is similar to the one for proving the corresponding statements for general $K(G, n)$ in [26]. However, we deviate in that we avoid the Freudenthal suspension theorem and theory about connectedness.

The neutral element of K_n is $*_{\mathsf{K}_n}$ and we denote it by 0_k. In order to prove that K_n is a group, we first define addition $+_k : \mathsf{K}_n \to \mathsf{K}_n \to \mathsf{K}_n$. The following lemma is the key for doing this. It is a special case of [38, Lemma 8.6.2], but the proof does not rely on general theory about connected types.

▶ **Lemma 8.** *Let $n, m \geq 1$ and suppose we have a fibration $P : \mathbb{S}^n \times \mathbb{S}^m \to (n+m-2)\text{-}\mathsf{Type}$ together with functions*

$$\mathsf{f}_l : (x : \mathbb{S}^n) \to P\,(x\,,*_{\mathbb{S}^m}) \qquad\qquad \mathsf{f}_r : (y : \mathbb{S}^m) \to P\,(*_{\mathbb{S}^n}\,, y)$$

and a path $p : \mathsf{f}_l\ *_{\mathbb{S}^n} \equiv \mathsf{f}_r\ *_{\mathbb{S}^m}$. There is a function $\mathsf{f} : (z : \mathbb{S}^n \times \mathbb{S}^m) \to P\,z$ with paths

$$\mathsf{left} : (x : \mathbb{S}^n) \to \mathsf{f}_l\ x \equiv \mathsf{f}\,(x\,,*_{\mathbb{S}^m}) \qquad\qquad \mathsf{right} : (y : \mathbb{S}^m) \to \mathsf{f}_r\ y \equiv \mathsf{f}\,(*_{\mathbb{S}^n}\,, y)$$

such that $p \equiv \mathsf{left}\ *_{\mathbb{S}^n} \cdot (\mathsf{right}\ *_{\mathbb{S}^m})^{-1}$. Furthermore, either left or right holds definitionally.

Proof. The proof is by sphere induction on both $\mathbb{S}^n$ and $\mathbb{S}^m$. For details see Appendix A.1. ◀

The general version of Lemma 8 is used for $K(G, n)$ in [26]. The advantage of the above form is the definitional reductions which follow from use of sphere induction in its proof. Consequently, we may define $+_k$ so that e.g. $0_k +_k |\, x \,| \equiv |\, x \,|$ holds definitionally. This allows for statements and proofs which would otherwise not be well-typed.

We define $+_k : \mathsf{K}_n \to \mathsf{K}_n \to \mathsf{K}_n$ and $-_k : \mathsf{K}_n \to \mathsf{K}_n$ by cases on n. When $n = 0$, these are integer addition and negation. Otherwise, we consider the following cases:

- When $n = 1$, we define $+_k$ and $-_k$ by cases:

$$\begin{aligned} |\, x \,| +_k |\, \mathsf{base} \,| &= |\, x \,| \\ |\, \mathsf{base} \,| +_k |\, \mathsf{loop}\ j \,| &= |\, \mathsf{loop}\ j \,| \\ |\, \mathsf{loop}\ i \,| +_k |\, \mathsf{loop}\ j \,| &= |\, \mathsf{Q}\ i\ j \,| \end{aligned} \qquad\qquad \begin{aligned} -_k\, |\, \mathsf{base} \,| &= |\, \mathsf{base} \,| \\ -_k\, |\, \mathsf{loop}\ i \,| &= |\, \mathsf{loop}\,(\sim i) \,| \end{aligned}$$

  where Q is a suitable filler of a square with loop on all sides. The filler Q is easily defined by an hcomp so that $\mathsf{cong}_2\ (\lambda\, x\, y \to |\, x \,| +_k |\, y \,|)\ \mathsf{loop}\ \mathsf{loop} \equiv \mathsf{cong}\ |\_|\ (\mathsf{loop} \cdot \mathsf{loop})$ holds definitionally. We will, from now on, with some abuse of notation, simply write loop for the canonical loop in K_1, i.e. $\mathsf{cong}\ |\_|\ \mathsf{loop}$.

- When $n \geq 2$, we need to construct a map $\mathbb{S}^n \times \mathbb{S}^n \to \mathsf{K}_n$ to define addition. Because K_n is n-truncated, it is also an $(n+n-2)$-Type. By Lemma 8, we are done if we can provide two maps $\mathbb{S}^n \to \mathsf{K}_n$ and prove that they agree on $*_{\mathbb{S}^n}$. In both cases we choose the inclusion map $\lambda\, x \to |\, x\, |$. We then just need to prove that $|\, *_{\mathbb{S}^n}\, | \equiv |\, *_{\mathbb{S}^n}\, |$, which we do by refl.

  To construct $-_k$, we send $|\,\mathsf{north}\,|$ and $|\,\mathsf{south}\,|$ to 0_k and $|\,\mathsf{merid}\ a\, i\,|$ to $\sigma_n\ a\ ({\sim}\, i)$. Here, σ_n is the map from the Freudenthal equivalence [38, Section 8.6] defined by:

$$\sigma_n : \mathsf{K}_n \to \Omega\, \mathsf{K}_{n+1}$$
$$\sigma_n\, |\, x\, | = \mathsf{cong}\, |\, \_\, |\ (\mathsf{merid}\ x \cdot (\mathsf{merid}\, *_{\mathbb{S}^n})^{-1})$$

The fact that $+_k$ and $-_k$ satisfy the group laws follows from Lemma 8. In fact, all group laws either hold by refl or have proofs that are at least path-equal to refl at 0_k. This in turn simplifies many later proofs and improves the efficiency of computations. We write lUnit_k/rUnit_k for the left/right unit laws and $\mathsf{lCancel}_k$/$\mathsf{rCancel}_k$ for the left/right inverse laws.

The definition of $+_k$ for $n \geq 2$ may seem naive. However, it provably agrees with the definition given in [5, Prop. 5.1.4]. In fact, a simple corollary of Lemma 8 is that there is at most one binary operation on K_n with lUnit_k and rUnit_k such that $\mathsf{lUnit}_k\ 0_k \equiv \mathsf{rUnit}_k\ 0_k$ (i.e. there is at most one *h-structure* [26, Def. 4.1] on K_n). The fact that this is satisfied by $+_k$ holds by refl. The same result was proved for the addition of [5, Prop. 5.1.4] in [28].

The group structure on K_n allows us to extend the usual encode-decode proof that $\mathbb{Z} \simeq \Omega\, \mathbb{S}^1$ (or, equivalently, $\mathsf{K}_0 \simeq \Omega\, \mathsf{K}_1$) to K_n with $n \geq 1$. We should note that a similar proof was used in [26] in order to prove that $G \simeq \pi_1(\mathsf{K}(G, 1))$.

▶ **Theorem 9.** $\mathsf{K}_n \simeq \Omega\, \mathsf{K}_{n+1}$

Proof. The proof is a direct encode-decode proof involving $+_k$ and σ_n. The details can be found in Appendix A.1. ◀

In addition to this, the direct definition of $+_k$ gives a short proof that $\Omega\, \mathsf{K}_n$ is commutative.

▶ **Lemma 10.** *For $n \geq 1$ and $p, q : \Omega\, \mathsf{K}_n$, we have $p \cdot q \equiv \mathsf{cong}_2\ +_k\ p\ q$.*

Proof. First, we remark that the statement is well-typed due to the definitional equality $0_k\ +_k\ 0_k \equiv 0_k$. Recall, $p, q : 0_k \equiv 0_k$ and $\mathsf{cong}_2\ +_k\ p\ q$ is of type $0_k\ +_k\ 0_k \equiv 0_k\ +_k\ 0_k$. Using this definitional equality, we may apply rUnit_k and lUnit_k pointwise to p and q:

$$p \equiv \mathsf{cong}\ (\lambda\, x\ \to\ x +_k 0_k)\ p \qquad\qquad q \equiv \mathsf{cong}\ (\lambda\, y\ \to\ 0_k +_k y)\ q$$

By functoriality of cong_2, we get

$$p \cdot q \equiv \mathsf{cong}\ (\lambda\, x \to x +_k 0_k)\ p \cdot \mathsf{cong}\ (\lambda\, y \to 0_k +_k y)\ q \equiv \mathsf{cong}_2\ +_k\ p\ q$$ ◀

▶ **Lemma 11.** *For $n \geq 1$ and $p, q : \Omega\, \mathsf{K}_n$, we have $\mathsf{cong}_2\ +_k\ p\ q \equiv \mathsf{cong}_2\ +_k\ q\ p$.*

Proof. By a very similar argument as in Lemma 10, but using commutativity of $+_k$. ◀

▶ **Theorem 12.** *$\Omega\, \mathsf{K}_n$ is commutative with respect to path composition.*

Proof. As $\mathbb{Z}$ is a set, this is trivial for $n = 0$. For $n \geq 1$ it follows from Lemmas 10 and 11. ◀

An alternative proof of Theorem 12 can be found in [5, Prop. 5.1.4]. In that proof, one first translates $\Omega\, \mathsf{K}_n$ into $\Omega^2\, \mathsf{K}_{n-1}$, applies the Eckmann-Hilton argument and then translates back. This translation back-and-forth is problematic from a computational point of view, and the proof of Theorem 12 is more computationally efficient.

3.2 Group structure on $H^n(A)$

We now return to $H^n(A)$ and define $0_h = |\,\lambda\, x \to 0_k\,|$ together with the group operations:

$$|\,f\,| +_h |\,g\,| = |\,\lambda\, x \to f\ x +_k g\ x\,| \qquad\qquad -_h |\,f\,| = |\,\lambda\, x \to -_k\ f\ x\,|$$

The fact that $(H^n(A), 0_h, +_h, -_h)$ forms an abelian group follows immediately from the group laws for K_n and funExt. We have also defined a *reduced* version of our cohomology theory and proved that it satisfies the Eilenberg-Steenrod axioms [16]. We refer the interested reader to Appendix B for the statement and verification of these axioms. This allows us to use abstract machinery to characterize cohomology groups of many spaces. However, in order to obtain definitions with good computational properties, we often prefer giving direct characterizations not relying on abstract results.

4 The Cup Product and Cohomology Ring

We will now equip the cohomology groups studied in the previous section with a multiplicative structure $\smile\, : H^n(A) \to H^m(A) \to H^{n+m}(A)$. This operation is called the *cup product* and it turns the $H^n(A)$ into a graded commutative ring $H^*(A)$ called the *cohomology ring* of A.

4.1 Defining the cup product in `Cubical Agda`

The cup product $\smile$ for $\mathbb{Z}$-cohomology in HoTT/UF was introduced by Brunerie [5, Section 5.1]. The definition is induced from a pointed map $\mathsf{K}_n \wedge \mathsf{K}_m \to_* \mathsf{K}_{n+m}$, where $\wedge$ is the *smash product* HIT. This HIT has proved to be surprisingly complex to reason about formally [6] and we therefore consider an alternative definition of $\smile$. The key observation in this reformulation is the pointed equivalence of $A \wedge B \to_* C$ and $A \to_* B \to_* C$ proved in HoTT/UF by van Doorn [39, Thm 4.3.8]. We hence construct $\smile$ by first defining a pointed map $x \smile_k y : \mathsf{K}_n \to \mathsf{K}_m \to_* \mathsf{K}_{n+m}$ by induction on n, thereby avoiding the smash product. When $n = 0$, this map just adds y to itself x times and similarly when $m = 0$. When $n, m \geq 1$, the key lemma is:

▶ **Lemma 13.** *The type $\mathsf{K}_m \to_* \mathsf{K}_{n+m}$ is an n-type.*

Proof. This is a special case of [9, Corollary 4.3]. We give a direct proof of this special case in Appendix A.2 which does not rely on any explicit connectedness arguments. ◀

Truncation elimination can hence be applied and we only need to define $|\,a\,| \smile_k y$ for $a : \mathbb{S}^n$.

$$\begin{array}{cc}
\mathbf{n = 1:} & \mathbf{n \geq 2:} \\
|\,\mathsf{base}\,| \smile_k y = 0_k & |\,\mathsf{north}\,| \smile_k y = 0_k \\
|\,\mathsf{loop}\ i\,| \smile_k y = \sigma_m\ y\ i & |\,\mathsf{south}\,| \smile_k y = 0_k \\
 & |\,\mathsf{merid}\ a\ i\,| \smile_k y = \sigma_{(n-1)+m}\ (|\,a\,| \smile_k y)\ i
\end{array}$$

The fact that $\lambda\, y \to x \smile_k y$ is pointed for $x : \mathsf{K}_n$ follows easily. In addition, we get pointedness in x immediately by construction. With this simple definition, we can now define the cup product $\smile\, : H^n(A) \to H^m(A) \to H^{n+m}(A)$ analogously to $+_h$ by:

$$|\,f\,| \smile |\,g\,| = |\,\lambda\, x \to f\, x \smile_k g\, x\,|$$

4.2 The cohomology ring

We will now prove that $\smile$ turns $H^n(A)$ into a graded ring. First of all, as $\smile_k$ is pointed in both arguments, we get that $x \smile 0_h \equiv 0_h \equiv 0_h \smile y$. Furthermore, it is easy to see that $1_h = |\,\lambda\, x \to 1\,|$ in $H^0(A)$ is a unit for $\smile$. The key lemma for proving properties of $\smile_k$ is:

▶ **Lemma 14.** *Given a pointed type A and two pointed functions $(f,p),(g,q) : A \to_* \mathsf{K}_n$, we have that if $f \equiv g$ then $(f,p) \equiv (g,q)$.*

Proof. This is proved using a notion of *homogeneous* types in Appendix A.2. ◀

In order to increase readability, we omit transports in Propositions 15, 17, and 18. We first verify that $\smile_k$ distributes over $+_k$.

▶ **Proposition 15.** *For $z : \mathsf{K}_n$ and $x, y : \mathsf{K}_m$, we have $z \smile_k (x +_k y) \equiv z \smile_k x +_k z \smile_k y$ and $(x +_k y) \smile_k z \equiv x \smile_k z +_k y \smile_k z$.*

Proof. We sketch the proof for left distributivity and focus on the case when $n, m \geq 1$. We want to show that $\lambda\, z \to z \smile_k (x +_k y)$ and $\lambda\, z \to z \smile_k x +_k z \smile_k y$ are equal *as pointed functions*. This allows for truncation elimination on x and y by Lemma 13. Thus we want to show that $z \smile_k (|\,a\,| +_k |\,b\,|) \equiv z \smile_k |\,a\,| +_k z \smile_k |\,b\,|$ for $a, b : \mathbb{S}^m$. We are proving an $(m-1)$-type and Lemma 8 applies. Hence we need to construct

$$f_l : (a : \mathbb{S}^n) \to z \smile_k (|\,a\,| +_k 0_k) \equiv z \smile_k |\,a\,| +_k z \smile_k 0_k$$

$$f_r : (b : \mathbb{S}^m) \to z \smile_k (0_k +_k |\,b\,|) \equiv z \smile_k 0_k +_k z \smile_k |\,b\,|$$

such that $f_l(*_{\mathbb{S}^n}) \equiv f_r(*_{\mathbb{S}^m})$. By Lemma 14, we only need to construct f_l and f_r for the underlying functions. We get f_l and f_r by applications of $\mathsf{lUnit}_k/\mathsf{rUnit}_k$ and the law of right multiplication by 0_k. Due to definitional equalities at 0_k, $f_l(*_{\mathbb{S}^n}) \equiv f_r(*_{\mathbb{S}^m})$ holds by refl. ◀

In order to prove that $\smile_k$ is associative, we need the following lemma:

▶ **Lemma 16.** *Let $n, m \geq 1$. For $x : \mathsf{K}_n$ and $y : \mathsf{K}_m$, $\sigma_{n+m}(x \smile_k y) \equiv \mathsf{cong}\,(\smile_k y)\ (\sigma_n\ x)$.*

Proof. This is proved in Appendix A.2. ◀

Lemma 16 occurs in [5, Prop. 6.1.1], albeit for a different definition of $\smile$. Interestingly, Brunerie does not use it to prove associativity of $\smile_k$, but to construct the *Gysin sequence*.

▶ **Proposition 17.** *For $x : \mathsf{K}_n$, $y : \mathsf{K}_m$ and $z : \mathsf{K}_\ell$, we have $x \smile_k (y \smile_k z) \equiv (x \smile_k y) \smile_k z$.*

Proof. The proof is easy when one of n, m or ℓ is 0. When $n, m, \ell \geq 1$, we want to show that $\lambda\, z\, y \to x \smile_k (y \smile_k z)$ and $\lambda\, z\, y \to (x \smile_k y) \smile_k z$ are equal as doubly pointed functions, i.e. as terms of $\mathsf{K}_m \to_* \mathsf{K}_\ell \to_* \mathsf{K}_{n+m+\ell}$. This is an n-type by repeated use of Lemma 13 and we may let $x = |\,a\,|$ for $a : \mathsf{K}_n$. We again only need to prove the underlying functions equal. We do this by induction on n. For $n = 1$, the case $a = \mathsf{base}$ holds by refl. In the case $a = \mathsf{loop}\, i$, we need to prove that $\sigma_{m+\ell}(y \smile_k z) \equiv \mathsf{cong}\,(\smile_k z)\ (\sigma_m\ y)$ which is Lemma 16. The $n \geq 2$ case follows by an analogous argument using the inductive hypothesis. ◀

Finally, we can verify that $\smile_k$ is graded commutative.

▶ **Proposition 18.** *For $x : \mathsf{K}_n$ and $y : \mathsf{K}_m$, we have $x \smile_k y \equiv \text{-}_k{}^{m \cdot n}\,(y \smile_k x)$.*

Proof. This is very non-trivial to check and a proof sketch can be found in Appendix A.2. ◀

The cup product $\smile$ inherits the properties of $\smile_k$ and we can hence organize $H^n(A)$ into a graded commutative ring $H^*(A)$.

5 Characterizing $\mathbb{Z}$-cohomology Groups

We will now characterize $H^n(A)$ for A being the spheres, torus, Klein bottle, and real/complex projective planes. It is an easy lemma that $H^0(A) \simeq \mathbb{Z}$ if A is 0-connected, which is the case for all types considered here. The cases when $H^n(A) \simeq \mathbb{1}$ for $n \geq 1$ are also easy using connectedness arguments. For a detailed proof of this for $\mathbb{S}^n$, see Appendix A.3. The main focus in this section will hence be on the non-trivial $H^n(A)$ with $n \geq 1$. Furthermore, we only focus on the equivalence parts of the characterizations, but we emphasize that all cases, including homomorphism proofs, have been formalized.

5.1 Spheres

The key to characterizing the cohomology groups of $\mathbb{S}^n$ is the **Suspension** axiom for cohomology. This axiom says that $H^n(A) \simeq H^{n+1}(\mathsf{Susp}\ A)$ and a proof can be found in Appendix B. Recall that $\mathbb{S}^{m+1} = \mathsf{Susp}\,\mathbb{S}^m$ for $m \geq 1$ and thus we have that $H^{n+1}(\mathbb{S}^{m+1}) \simeq H^n(\mathbb{S}^m)$.

▶ **Proposition 19.** *$H^n(\mathbb{S}^n) \simeq \mathbb{Z}$ for $n \geq 1$.*

Proof. By **Suspension** and induction, it suffices to consider the $n = 1$ case. We inspect the underlying function space of $H^1(\mathbb{S}^1)$, i.e. $\mathbb{S}^1 \to \mathsf{K}_1$. A map $f : \mathbb{S}^1 \to \mathsf{K}_1$ is uniquely determined by $f\ \mathsf{base} : \mathsf{K}_1$ and $\mathsf{cong}\ f\ \mathsf{loop} : f\ \mathsf{base} \equiv f\ \mathsf{base}$. Thus, we have $H^1(\mathbb{S}^1) \simeq \|\sum_{x:\mathsf{K}_1} x \equiv x\|_0$. By a base change we get $(x \equiv x) \simeq (0_k \equiv 0_k)$ for any $x : \mathsf{K}_1$. Hence

$$H^1(\mathbb{S}^1) \simeq \|\,\mathsf{K}_1 \times \Omega\,\mathsf{K}_1\,\|_0 \simeq \|\,\mathsf{K}_1\,\|_0 \times \|\,\Omega\,\mathsf{K}_1\,\|_0 \simeq \|\,\Omega\,\mathsf{K}_1\,\|_0 \simeq \|\,\Omega\,\mathbb{S}^1\,\|_0 \simeq \mathbb{Z}$$ ◀

5.2 The torus

The torus HIT, $\mathbb{T}^2$, is defined as follows:

```
data 𝕋² : Type where
  pt : 𝕋²
  ℓ₁ ℓ₂ : pt ≡ pt
  □ : PathP (λ i → ℓ₂ i ≡ ℓ₂ i) ℓ₁ ℓ₁
```

The constructor □ corresponds to the usual gluing diagram for constructing the torus in classical topology as it identifies ℓ_1 with itself over an identification of ℓ_2 with itself. As discussed in the introduction, proving $\mathbb{T}^2 \simeq \mathbb{S}^1 \times \mathbb{S}^1$ is easy in `Cubical Agda`. This allows us to curry $\mathbb{T}^2 \to \mathsf{K}_n$, which is the key step to prove Propositions 20 and 21.

▶ **Proposition 20.** *$H^1(\mathbb{T}^2) \simeq \mathbb{Z} \times \mathbb{Z}$*

Proof. We inspect the underlying function space $\mathbb{T}^2 \to \mathsf{K}_1$, which is equivalent to $\mathbb{S}^1 \to (\mathbb{S}^1 \to \mathsf{K}_1)$. From Proposition 19, we know that $(\mathbb{S}^1 \to \mathsf{K}_1) \simeq \mathsf{K}_1 \times \Omega\,\mathsf{K}_1 \simeq \mathsf{K}_1 \times \mathbb{Z}$. Hence

$$H^1(\mathbb{T}^2) \simeq \|\,\mathbb{S}^1 \to \mathsf{K}_1 \times \mathbb{Z}\,\|_0 \simeq \|\,\mathbb{S}^1 \to \mathsf{K}_1\,\|_0 \times \|\,\mathbb{S}^1 \to \mathbb{Z}\,\|_0 \stackrel{\text{def}}{\equiv} H^1(\mathbb{S}^1) \times H^0(\mathbb{S}^1) \simeq \mathbb{Z} \times \mathbb{Z}$$ ◀

▶ **Proposition 21.** *$H^2(\mathbb{T}^2) \simeq \mathbb{Z}$*

Proof. The underlying function space, post currying, is $\mathbb{S}^1 \to (\mathbb{S}^1 \to \mathsf{K}_2)$. Like above, this is $(\mathbb{S}^1 \to \mathsf{K}_2 \times \Omega\,\mathsf{K}_2) \simeq (\mathbb{S}^1 \to \mathsf{K}_2 \times \mathsf{K}_1) \simeq (\mathbb{S}^1 \to \mathsf{K}_2) \times (\mathbb{S}^1 \to \mathsf{K}_1)$. Hence

$$H^2(\mathbb{T}^2) \simeq \|\,(\mathbb{S}^1 \to \mathsf{K}_2) \times (\mathbb{S}^1 \to \mathsf{K}_1)\,\|_0 \simeq H^2(\mathbb{S}^1) \times H^1(\mathbb{S}^1) \simeq \mathbb{Z}$$ ◀

5.3 The Klein bottle and real projective plane

The Klein bottle and real projective plane are also HITs, but with twists in $\square$ just like in the classical gluing diagrams:

```
data 𝕂² : Type where            data ℝP² : Type where
  pt : 𝕂²                         pt : ℝP²
  ℓ₁ ℓ₂ : pt ≡ pt                 ℓ : pt ≡ pt
  □ : PathP (λ i → ℓ₂ (∼ i) ≡ ℓ₂ i) ℓ₁ ℓ₁     □ : ℓ ≡ ℓ ⁻¹
```

Note that $\square$ for $\mathbb{K}^2$ equivalently may be interpreted as the path $\ell_2 \cdot \ell_1 \cdot \ell_2 \equiv \ell_1$. To characterize the cohomology groups of $\mathbb{K}^2$, we need to understand their underlying function spaces. It is easy to see that

$$(\mathbb{K}^2 \to \mathsf{K}_n) \simeq \sum_{x:\mathsf{K}_n} \sum_{p,q:x \,\equiv\, x} (p \cdot q \cdot p \equiv q)$$

By Theorem 12, $\_\cdot\_$ in $\Omega\,\mathsf{K}_n$ is commutative, so $(p \cdot q \cdot p \equiv q) \simeq (p \cdot p \equiv \mathsf{refl})$. Hence

$$(\mathbb{K}^2 \to \mathsf{K}_n) \simeq \sum_{x:\mathsf{K}_n} \left((x \equiv x) \times \sum_{p:x \,\equiv\, x} (p \cdot p \equiv \mathsf{refl}) \right) \tag{1}$$

▶ **Proposition 22.** $H^1(\mathbb{K}^2) \simeq \mathbb{Z}$

Proof. Note that for $x : \mathsf{K}_1$, we have that $\sum_{p:x\,\equiv\,x}(p \cdot p \equiv \mathsf{refl}) \simeq \sum_{a:\mathbb{Z}}(a + a \equiv 0) \simeq \mathbb{1}$. This allows us to simplify (1) and get

$$H^1(\mathbb{K}^2) \simeq \|\, \mathbb{K}^2 \to \mathsf{K}_1 \,\|_0 \simeq \|\, \sum_{x:\mathsf{K}_1} (x \equiv x) \,\|_0 \simeq H^1(\mathbb{S}^1) \simeq \mathbb{Z}$$ ◀

▶ **Lemma 23.** *For $n : \mathbb{Z}$, define $p : \|\, \sum_{x:\mathsf{K}_1} (x +_k x \equiv 0_k) \,\|_0$* by $p = |\, (0_k\,, \mathsf{loop}^n) \,|$. We have $p \equiv |\, (0_k\,, \mathsf{refl}) \,|$ if n is even and $p \equiv |\, (0_k\,, \mathsf{loop}) \,|$ if n is odd.

Proof. This is proved in Appendix A.3. ◀

▶ **Proposition 24.** $H^2(\mathbb{K}^2) \simeq \mathbb{Z}/2\mathbb{Z}$

Proof. Using 0-connectedness of K_2 and $(x \equiv x)$ for $x : \mathsf{K}_2$, it is easy to see that, by truncating both sides of (1), we get

$$H^2(\mathbb{K}^2) \simeq \|\, \sum_{p:\Omega\,\mathsf{K}_2} (p \cdot p \equiv \mathsf{refl}) \,\|_0$$

Using the equivalence $\Omega\,\mathsf{K}_2 \simeq \mathsf{K}_1$ and the fact that it takes path composition to addition, this can be further simplified to $\|\, \sum_{x:\mathsf{K}_1} (x +_k x \equiv 0_k) \,\|_0$. It is easy to see that for any $p : \|\, \sum_{x:\mathsf{K}_1} (x +_k x \equiv 0_k) \,\|_0$, we have that $p \equiv |\, (0_k\,, \mathsf{loop}^n) \,|$ for some $n : \mathbb{N}$. We map p into $\mathbb{Z}/2\mathbb{Z}$ by sending it to 0 if n is even and 1 if n is odd. As an immediate consequence of Lemma 23, this map must be an equivalence, and thus we are done. ◀

The attentive reader will have noticed that something reminiscent of the real projective plane, $\mathbb{R}P^2$, appears in both proofs in this section. We characterize $H^n(\mathbb{R}P^2)$ for $n \geq 1$ by

$$\|\, \mathbb{R}P^2 \to \mathsf{K}_n \,\|_0 \simeq \|\, \sum_{x:\mathsf{K}_n} \sum_{p:x\,\equiv\,x} (p \equiv p^{-1}) \,\|_0 \simeq \|\, \sum_{x:\mathsf{K}_n} \sum_{p:x\,\equiv\,x} (p \cdot p \equiv \mathsf{refl}) \,\|_0 \simeq \|\, \sum_{p:\Omega\,\mathsf{K}_n} (p \cdot p \equiv \mathsf{refl}) \,\|_0$$

When n is 1 or 2, this is precisely one of the types appearing in the proofs of Propositions 22 and 24 respectively, so $H^1(\mathbb{R}P^2) \simeq \mathbb{1}$ and $H^2(\mathbb{R}P^2) \simeq \mathbb{Z}/2\mathbb{Z}$.

5.4 The complex projective plane

We define the complex projective plane, $\mathbb{C}P^2$, as the pushout of the span $\mathbb{S}^2 \xleftarrow{h} \mathbb{S}^3 \to \mathbb{1}$ where h is part of the Hopf fibration [38, Section 8.5]. The function space $\mathbb{C}P^2 \to \mathsf{K}_n$ is quite hard to work with directly, so we settle for an indirect characterization of $H^n(\mathbb{C}P^2)$ via the Mayer-Vietoris sequence (see Appendix B.3). For $n \geq 2$, this gives us an exact sequence:

$$H^{n-1}(\mathbb{S}^2) \to H^{n-1}(\mathbb{S}^3) \to H^n(\mathbb{C}P^2) \to H^n(\mathbb{S}^2) \to H^n(\mathbb{S}^3)$$

For $n \in \{3, 5, 6, \dots\}$, we have that $H^n(\mathbb{C}P^2) \simeq \mathbb{1}$, as other groups in the sequence become trivial. When $n = 2$, all groups but $H^2(\mathbb{S}^2)$ are trivial, and hence $H^2(\mathbb{C}P^2) \simeq H^2(\mathbb{S}^2) \simeq \mathbb{Z}$. When $n = 4$, the only non trivial group is $H^3(\mathbb{S}^3)$, and hence we get $H^4(\mathbb{C}P^2) \simeq H^3(\mathbb{S}^3) \simeq \mathbb{Z}$. A simple connectedness argument finally gives us that $H^1(\mathbb{C}P^2) \simeq \mathbb{1}$.

6 Proving by computations in `Cubical Agda`

One of the appealing aspects of developing cohomology theory in `Cubical Agda` is that we can prove properties purely by computation. This can discharge proof goals involving complex path algebra as soon as the types are fully instantiated. For example, in Proposition 18 when $m = n = 1$, the main subgoal involves compositions paths in $\Omega^2\,\mathsf{K}_2$ which can be reduced to a computation purely involving $\mathbb{Z}$, using the equivalence $\Omega^2\,\mathsf{K}_2 \simeq \mathbb{Z}$. As we have been careful about proving things as directly as possible with efficient computations in mind, this works quite well, but there are some cases which are surprisingly slow in `Cubical Agda`, and we have collected some benchmarks in Appendix C.

Furthermore, we can use the fact that the isomorphisms compute to establish that some types cannot be equivalent. This is the case for all spaces in the previous section, as they have different cohomology groups. However, there are some spaces where it is not enough to only look at the cohomology groups. We have proved in Appendix B that our cohomology theory satisfies the **Binary Additivity** axiom which says that $H^n(A \vee B) \simeq H^n(A) \times H^n(B)$. So we can easily prove that $\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1$ has the same cohomology groups as $\mathbb{T}^2$. However, these two types are not equivalent and the standard way to prove this is to use the cup product. We can do this traditional proof computationally in `Cubical Agda` by defining a predicate $P : \mathsf{Type} \to \mathsf{Type}$ by $P(A) = (x\ y : H^1(A)) \to x \smile y \equiv 0_h$ and show that $P(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1)$ holds while $P(\mathbb{T}^2)$ does not. In `Cubical Agda`, we have defined isomorphisms:

$$f_1 : H^1(\mathbb{T}^2) \cong \mathbb{Z} \times \mathbb{Z} \qquad g_1 : H^1(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1) \cong \mathbb{Z} \times \mathbb{Z}$$
$$f_2 : H^2(\mathbb{T}^2) \cong \mathbb{Z} \qquad g_2 : H^2(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1) \cong \mathbb{Z}$$

To disprove $P(\mathbb{T}^2)$ we need $x, y : H^1(\mathbb{T}^2)$ such that $x \smile y \not\equiv 0_h$. Let $x = f_1^{-1}(0, 1)$ and $y = f_1^{-1}(1, 0)$. In `Cubical Agda`, $f_2(x \smile y) \equiv 1$ holds by refl and thus $x \smile y \not\equiv\ 0_k$. It remains to prove $P(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1)$. Let $x, y : H^1(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1)$. In `Cubical Agda`, we have that $g_2(g_1^{-1}(g_1\ x) \smile g_1^{-1}(g_1\ y)) \equiv 0$, again by refl, and thus $g_1^{-1}(g_1\ x) \smile g_1^{-1}(g_1\ y) \equiv x \smile y \equiv 0_h$.

For a more ambitious example, consider Chapter 6 of Brunerie's PhD thesis [5]. This chapter is devoted to proving, using sophisticated techniques like the Gysin sequence, that the generator $e : H^2(\mathbb{C}P^2)$ when multiplied with itself yields a generator of $H^4(\mathbb{C}P^2)$. Let $g : \mathbb{Z} \to \mathbb{Z}$ be the map described by

$$\mathbb{Z} \xrightarrow{\cong} H^2(\mathbb{C}P^2) \xrightarrow{\lambda\, x \to x\, \smile\, x} H^4(\mathbb{C}P^2) \xrightarrow{\cong} \mathbb{Z}$$

The number $g(1)$ should reduce to ± 1 for $e \smile e$ to generate $H^4(\mathbb{C}P^2)$ and by evaluating it in `Cubical Agda` we should be able to reduce the whole chapter to a single computation.

However, Cubical Agda is currently stuck on computing $g(1)$. This number can hence be seen as another "Brunerie number"—a mathematically interesting number which is currently infeasible to compute using an implementation of cubical type theory. This computation should be more feasible than the original Brunerie number. As our definition of $\smile$ produces very simple terms, most of the work has to occur in the two isomorphisms, and we are optimistic that future optimizations will allow us to perform this computation.

7 Conclusions

We have developed multiple classical results from cohomology theory synthetically in Cubical Agda. This has led to new and more direct constructive proofs than what already exists in the HoTT/UF literature. Furthermore, Section 4 contains the first fully formalized verification of the graded commutative ring axioms for $\mathbb{Z}$-cohomology. The key to this is the new definition of $\smile$ which avoids the smash product. The synthetic characterizations of the cohomology groups of $\mathbb{K}^2$ and $\mathbb{R}P^2$ are also novel. The proofs have been constructed with computational efficiency in mind, allowing us to make explicit computations involving several non-trivial cohomology groups. In particular, the number $g(1)$ is another "Brunerie number" which should be more feasible to compute, and its computation would allow us to reduce the complex proofs of [5, Chapter 6] to a single computation. This is hence a new challenge for future improvements of Cubical Agda and related systems like cooltt [37].

7.1 Related and future work

In addition to the related work already mentioned in the paper, there is some related prior work in Cubical Agda. Qian [32] formalized $K(G, 1)$ as a HIT, following [26], and proved that it satisfies $\pi_1(K(G, 1)) \equiv G$. Alfieri [1] and Harington [18] formalized $K(G, 1)$ as the classifying space BG using G-torsors. Using this, $H^1(\mathbb{S}^1; \mathbb{Z}) \equiv \mathbb{Z}$ was proved—however, computing using the maps in this definition proved to be infeasible. It is not clear where the bottlenecks are, but we emphasize that with the definitions in this paper, there are no problems computing with this cohomology group.

Certified computations of homology groups using proof assistants have been considered before the invention of HoTT/UF. For instance, the Coq system [36] has been used to compute homology [21] and persistent homology [20] with coefficients in a field. This was later extended to homology with $\mathbb{Z}$-coefficients in [10]. The approach in these papers was entirely algebraic and spaces were represented as simplicial complexes. However, a synthetic approach to homology in HoTT/UF was developed informally by Graham [17] using stable homotopy groups. This was later extended with a proof of Hurewicz theorem by Christensen and Scoccola [13]. It would be interesting to see if this could be made formal in Cubical Agda so that we can also characterize and compute with homology groups.

The definition of $H^*(A)$ in HoTT/UF is due to Brunerie [5, Chapter 5.1]. Here, however, $\smile$ relies on the smash product which has proved very complex to reason about formally [6]. Despite this, Baumann generalized this to $H^n(X; G)$ and managed to formalize graded commutativity in HoTT-Agda [4]. Baumann's formal proof of this property is $\sim$ 5000 LOC while our formalization is just $\sim$ 900 LOC. This indicates that it would be infeasible to formalize other algebraic properties of $H^*(A)$ with this definition. Associativity seems particularly infeasible, but with our definition the formal proof is only $\sim$ 200 LOC. However, this comparison should be taken with a grain of salt as Baumann proves the result for $H^n(X; G)$. Nevertheless, we conjecture that our constructions should be relatively easy to generalize to cohomology with coefficients in an arbitrary group.

References

1 Victor Alfieri. Formalisation de notions de théorie des groupes en théorie cubique des types, 2019. Internship report, supervised by Thierry Coquand.

2 Carlo Angiuli, Evan Cavallo, Anders Mörtberg, and Max Zeuner. Internalizing representation independence with univalence. *Proc. ACM Program. Lang.*, 5(POPL), January 2021. doi:10.1145/3434293.

3 Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 146(1):45–55, January 2009. doi:10.1017/S0305004108001783.

4 Tim Baumann. The cup product on cohomology groups in homotopy type theory. Master's thesis, University of Augsburg, 2018.

5 Guillaume Brunerie. *On the homotopy groups of spheres in homotopy type theory.* PhD thesis, Université Nice Sophia Antipolis, 2016. URL: http://arxiv.org/abs/1606.05916.

6 Guillaume Brunerie. Computer-generated proofs for the monoidal structure of the smash product. *Homotopy Type Theory Electronic Seminar Talks*, November 2018. URL: https://www.uwo.ca/math/faculty/kapulkin/seminars/hottest.html.

7 Guillaume Brunerie, Kuen-Bang Hou (Favonia), Evan Cavallo, Tim Baumann, Eric Finster, Jesper Cockx, Christian Sattler, Chris Jeris, Michael Shulman, et al. Homotopy Type Theory in Agda, 2018. URL: https://github.com/HoTT/HoTT-Agda.

8 Ulrik Buchholtz and Kuen-Bang Hou Favonia. Cellular Cohomology in Homotopy Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 521–529, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3209108.3209188.

9 Ulrik Buchholtz, Floris van Doorn, and Egbert Rijke. Higher Groups in Homotopy Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 205–214, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3209108.3209150.

10 Guillaume Cano, Cyril Cohen, Maxime Dénès, Anders Mörtberg, and Vincent Siles. Formalized Linear Algebra over Elementary Divisor Rings in Coq. *Logical Methods in Computer Science*, 12(2), 2016. URL: http://dx.doi.org/10.2168/LMCS-12(2:7)2016, doi:10.2168/LMCS-12(2:7)2016.

11 Evan Cavallo. Synthetic Cohomology in Homotopy Type Theory. Master's thesis, Carnegie Mellon University, 2015.

12 Evan Cavallo and Robert Harper. Higher Inductive Types in Cubical Computational Type Theory. *Proceedings of the ACM on Programming Languages*, 3(POPL):1:1–1:27, January 2019. doi:10.1145/3290314.

13 J. Daniel Christensen and Luis Scoccola. The Hurewicz theorem in Homotopy Type Theory, 2020. Preprint. URL: https://arxiv.org/abs/2007.05833, arXiv:2007.05833.

14 Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In Tarmo Uustalu, editor, *21st International Conference on Types for Proofs and Programs (TYPES 2015)*, volume 69 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:34, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TYPES.2015.5.

15 Thierry Coquand, Simon Huber, and Anders Mörtberg. On Higher Inductive Types in Cubical Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS 2018, pages 255–264, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209197.

16 Samuel Eilenberg and Norman Steenrod. *Foundations of Algebraic Topology.* Foundations of Algebraic Topology. Princeton University Press, 1952.

17 Robert Graham. Synthetic Homology in Homotopy Type Theory, 2018. Preprint. URL: https://arxiv.org/abs/1706.01540, arXiv:1706.01540.

18 Elies Harington. Groupes de cohomologie en théorie des types univalente, 2020. Internship report, supervised by Thierry Coquand.

19 Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002. URL: `https://pi.math.cornell.edu/~hatcher/AT/AT.pdf`.

20 Jónathan Heras, Thierry Coquand, Anders Mörtberg, and Vincent Siles. Computing Persistent Homology Within Coq/SSReflect. *ACM Transactions on Computational Logic*, 14(4):1–26, 2013. URL: `http://doi.acm.org/10.1145/2528929`, `doi:10.1145/2528929`.

21 Jónathan Heras, Maxime Dénès, Gadea Mata, Anders Mörtberg, María Poza, and Vincent Siles. Towards a Certified Computation of Homology Groups for Digital Images. In *Proceedings of the 4th International Conference on Computational Topology in Image Context*, CTIC'12, pages 49–57, Berlin, Heidelberg, 2012. Springer-Verlag. `doi:10.1007/978-3-642-30238-1_6`.

22 Kuen-Bang Hou (Favonia), Eric Finster, Daniel R. Licata, and Peter LeFanu Lumsdaine. A Mechanization of the Blakers-Massey Connectivity Theorem in Homotopy Type Theory. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '16, pages 565–574, New York, NY, USA, 2016. ACM. `doi:10.1145/2933575.2934545`.

23 Kuen-Bang Hou (Favonia) and Michael Shulman. The Seifert-van Kampen Theorem in Homotopy Type Theory. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, volume 62 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.CSL.2016.22`.

24 Krzysztof Kapulkin and Peter LeFanu Lumsdaine. The Simplicial Model of Univalent Foundations (after Voevodsky), June 2016. Preprint. URL: `https://arxiv.org/abs/1211.2851`, `arXiv:1211.2851`.

25 Daniel R. Licata and Guillaume Brunerie. A Cubical Approach to Synthetic Homotopy Theory. In *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '15, pages 92–103, Washington, DC, USA, 2015. IEEE Computer Society. `doi:10.1109/LICS.2015.19`.

26 Daniel R. Licata and Eric Finster. Eilenberg-MacLane Spaces in Homotopy Type Theory. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, CSL-LICS '14, New York, NY, USA, 2014. Association for Computing Machinery. `doi:10.1145/2603088.2603153`.

27 Daniel R. Licata and Michael Shulman. Calculating the Fundamental Group of the Circle in Homotopy Type Theory. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '13, pages 223–232, Washington, DC, USA, 2013. IEEE Computer Society. `doi:10.1109/LICS.2013.28`.

28 Axel Ljungström. Computing Cohomology in `Cubical Agda`. Master's thesis, Stockholm University, 2020.

29 Per Martin-Löf. An Intuitionistic Theory of Types: Predicative Part. In H. E. Rose and J. C. Shepherdson, editors, *Logic Colloquium '73*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 73–118. North-Holland, 1975. `doi:10.1016/S0049-237X(08)71945-1`.

30 Per Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in Proof Theory*. Bibliopolis, 1984.

31 Anders Mörtberg and Loïc Pujet. Cubical Synthetic Homotopy Theory. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, pages 158–171, New York, NY, USA, 2020. Association for Computing Machinery. `doi:10.1145/3372885.3373825`.

32 Zesen Qian. Towards Eilenberg-MacLane Spaces in Cubical Type Theory. Master's thesis, Carnegie Mellon University, 2019.

33 Michael Shulman. Cohomology, 2013. post on the Homotopy Type Theory blog: `http://homotopytypetheory.org/2013/07/24/`.

34 Kristina Sojakova. The Equivalence of the Torus and the Product of Two Circles in Homotopy Type Theory. *ACM Transactions on Computational Logic*, 17(4):29:1–29:19, November 2016. doi:10.1145/2992783.

35 The Agda Development Team. The Agda Programming Language, 2021. URL: http://wiki.portal.chalmers.se/agda/pmwiki.php.

36 The Coq Development Team. The Coq Proof Assistant, 2021. URL: https://www.coq.inria.fr.

37 The RedPRL Development Team. The cooltt proof assistant, 2021. URL: https://github.com/RedPRL/cooltt/.

38 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Self-published, 2013. URL: https://homotopytypetheory.org/book/.

39 Floris van Doorn. *On the Formalization of Higher Inductive Types and Synthetic Homotopy Theory*. PhD thesis, University of Nottingham, May 2018. URL: https://arxiv.org/abs/1808.10690.

40 Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types. *Proceedings of the ACM on Programming Languages*, 3(ICFP):87:1–87:29, August 2019. doi:10.1145/3341691.

41 Vladimir Voevodsky. The equivalence axiom and univalent models of type theory, February 2010. Notes from a talk at Carnegie Mellon University. URL: http://www.math.ias.edu/vladimir/files/CMU_talk.pdf.

42 Vladimir Voevodsky. An experimental library of formalized mathematics based on the univalent foundations. *Mathematical Structures in Computer Science*, 25(5):1278–1294, 2015. doi:10.1017/S0960129514000577.

A Proofs

This appendix contains proofs for the results in the paper. Everything has been formalized in Cubical Agda, and we refer the interested reader to the formalized code for technical details.

A.1 Proofs for Section 3

Proof of Lemma 8. The proof proceeds by induction: first on n and then on m for the case $n = 1$. For $n = m = 1$, we define the map

$$\begin{aligned}
&\mathsf{f} : (z : \mathbb{S}^1 \times \mathbb{S}^1) \to P\ z\\
&\mathsf{f}\ (x\ , \mathsf{base}) = \mathsf{f}_l\ x\\
&\mathsf{f}\ (\mathsf{base}\ , \mathsf{loop}\ i) = (p \cdot' \mathsf{cong}\ f_r\ \mathsf{loop} \cdot' p^{-1})\ i\\
&\mathsf{f}\ (\mathsf{loop}\ i\ , \mathsf{loop}\ j) = \mathsf{Q}\ i\ j
\end{aligned}$$

where Q is given by the fact that P is a set. Here $\_\cdot'\_$ is the dependent version of $\_\cdot\_$ for PathP (over $\_\cdot\_$). The left path is just refl and the right path is easy to construct by circle induction. In particular, we let $\mathsf{right}\ *_{\mathbb{S}^1} = p^{-1}$. Thus $p \equiv \mathsf{left}\ *_{\mathbb{S}^1} \cdot (\mathsf{right}\ *_{\mathbb{S}^1})^{-1}$ is immediate by construction.

For the inductive step, we focus on $\mathbb{S}^{n+1} \times \mathbb{S}^m$ and omit the proof for $\mathbb{S}^1 \times \mathbb{S}^{m+1}$ since it is close to identical. We begin by defining f for north and south.

$$\begin{aligned}
&\mathsf{f}\ (\mathsf{north}\ , y) = \mathsf{f}_r\ y\\
&\mathsf{f}\ (\mathsf{south}\ , y) = \mathsf{transport}\ (\lambda\, i \to P\ (\mathsf{merid}\ *_{\mathbb{S}^m}\ i\ , y))\ (\mathsf{f}_r\ y)
\end{aligned}$$

Note that already here, we have the right path; it holds by refl. The left path is constructed in parallel with f. Thus far, we can only define it for north and south. This is easily done so that $p \equiv \mathsf{left}\ *_{\mathbb{S}^{n+1}} \cdot (\mathsf{right}\ *_{\mathbb{S}^m})^{-1}$ is satisfied.

We now need to define $\mathsf{f}\,(\mathsf{merid}\ x\,i\,,y)$. That is, we need to provide a dependent path from $\mathsf{f}_r\ y$ to

$$\mathsf{transport}\ (\lambda\, i \to P\,(\mathsf{merid}\ *_{\mathbb{S}^m}\ i\,,y))\ (\mathsf{f}_r\ y)$$

over $P\,(\mathsf{merid}\ x\ i\,,y)$ for $(x\,,y) : \mathbb{S}^n \times \mathbb{S}^m$. The type of such paths is an $(n+m-2)$-type and we may apply the inductive hypothesis. This means that we only need to construct it for $(*_{\mathbb{S}^n}\,,y)$ and $(x\,,*_{\mathbb{S}^m})$ and prove that these two constructions agree on $(*_{\mathbb{S}^n}\,,*_{\mathbb{S}^m})$. Furthermore, since it remains to construct $\mathsf{left}\ (\mathsf{merid}\ \ x\ i)$, this construction has to respect the definition of $\mathsf{left}\ \mathsf{north}$ and $\mathsf{left}\ \mathsf{south}$. This follows in a straightforward manner from the left and right paths given by the inductive hypothesis. We omit the construction—it is not difficult, but rather technical. ◀

Proof of Theorem 9. The case $n = 0$ is just $\mathbb{Z} \simeq \Omega\,\mathbb{S}^1$, so we focus on the case when $n \geq 1$. We proceed by the encode-decode method and define a fibration $\mathsf{Code} : \mathsf{K}_{n+1} \to n\text{-}\mathsf{Type}$. Since $n\text{-}\mathsf{Type}$ is $(n+1)$-truncated [38, Theorem 7.1.11], we may define it by truncation elimination.

$$\begin{aligned}
\mathsf{Code}\ |\,\mathsf{north}\,| &= \mathsf{K}_n\\
\mathsf{Code}\ |\,\mathsf{south}\,| &= \mathsf{K}_n\\
\mathsf{Code}\ |\,\mathsf{merid}\ x\ i\,| &= \mathsf{ua}\ (\lambda\, y\ \to\ |\,x\,| +_k y)\ i
\end{aligned}$$

The last case uses the fact that for any $x : \mathsf{K}_n$, the map $\lambda\, y \to x +_k y$ is an equivalence. As usual, we define

$$\begin{aligned}
&\mathsf{encode} : (x : \mathsf{K}_{n+1}) \to 0_k \equiv x \to \mathsf{Code}\ x\\
&\mathsf{encode}\ x\ p = \mathsf{subst}\ \mathsf{Code}\ p\ 0_k
\end{aligned}$$

The inverse is defined by

$$\begin{aligned}
&\mathsf{decode} : (x : \mathsf{K}_{n+1}) \to \mathsf{Code}\ x \to 0_k \equiv x\\
&\mathsf{decode}\ |\,\mathsf{north}\,| = \sigma_n\\
&\mathsf{decode}\ |\,\mathsf{south}\,| = \lambda\,|\,x\,| \to \mathsf{cong}\ |\,_\,|\ (\mathsf{merid}\ x)\\
&\mathsf{decode}\ |\,\mathsf{merid}\ \ y\,i\,| = \ldots
\end{aligned}$$

For the missing case we need to prove that the function

$$\mathsf{transport}\ (\lambda\, i \to \mathsf{Code}\ |\,\mathsf{merid}\ y\,i\,| \to 0_k \equiv |\,\mathsf{merid}\ y\ i\,|)\ \sigma_n$$

takes $|\,x\,|$ to $\mathsf{cong}\ |\,\_\,|\ (\mathsf{merid}\ x)$. By the $\mathsf{transport}$ laws for functions and ua, we can deduce that this function applied to $|\,x\,|$ yields the following (up to a path):

$$\sigma_n\ (-_k\ |\,y\,| +_k |\,x\,|) \cdot \mathsf{cong}\ |\,\_\,|\ (\mathsf{merid}\ y)$$

It follows easily from Lemma 8 that σ_n is a homomorphism in the sense that $\sigma_n\ (x +_k y) \equiv \sigma_n\ x \cdot \sigma_n\ y$. Furthermore, as $+_k$ is commutative, we obtain:

$$\sigma_n\ |\,x\,| \cdot (\sigma_n\ |\,y\,|)^{-1} \cdot \mathsf{cong}\ |\,\_\,|\ (\mathsf{merid}\ y)$$

Unfolding $\sigma_n\ \ |\,x\,|$ and $\sigma_n\ \ |\,y\,|$ then yields a composition of paths which simplifies to $\mathsf{cong}\ |\,\_\,|\ (\mathsf{merid}\ x)$ as desired.

Proving that $\mathsf{encode}\,|\,\mathsf{north}\,|$ and $\mathsf{decode}\,|\,\mathsf{north}\,|$ are mutually inverse is very direct. By generalizing to any $x : \mathsf{K}_{n+1}$, $\mathsf{decode}\ x\ (\mathsf{encode}\ x\ p) \equiv p$ follows by path induction. By the $\mathsf{transport}$ law for ua, the other direction amounts to showing $(|\,y\,| +_k 0_k) -_k 0_k \equiv |\,y\,|$, which clearly holds. ◀

A.2 Proofs for Section 4

To prove the results in Section 4, we introduce a notion of *homogeneous* types.

▶ **Definition 25.** *We say that two pointed types* $(A\,,a_0)$ *and* $(B\,,b_0)$ *are equivalent if there is an equivalence* $f : A \simeq B$ *such that* $f\ a_0 \equiv b_0$*. We denote this by* $(A\,,a_0) \simeq_* (B\,,b_0)$*.*

▶ **Definition 26.** *A type* A *is* homogeneous *if for any* $x, y : A$*, we have that* $(A\,,x) \simeq_* (A\,,y)$ *or, equivalently,* $(A\,,x) \equiv (A\,,y)$*.*

The following very useful lemma is due to Evan Cavallo.

▶ **Lemma 27.** *Let* A *and* B *be pointed types with* B *homogeneous and let* $(f\,,p), (g\,,q) : A \to_* B$ *be pointed functions. If* $f \equiv g$*, then* $(f\,,p) \equiv (g\,,q)$*.*

Proof. Let A and B be pointed by a_0 and b_0 respectively. By assumption, we have a homotopy $h : (x : A) \to f\ x \equiv g\ x$. We construct a path $r : b_0 \equiv b_0$ by $r = p^{-1} \cdot h\ a_0 \cdot q$. Define $P : (B\,,b_0) \equiv_{\mathsf{Type}_*} (B\,,b_0)$ by $P = \lambda\ i \to (B\,,r\ i)$. We get $P \equiv \mathsf{refl}$ as an easy consequence of the homogeneity of B. Hence, instead of proving that $(f\,,p) \equiv (g\,,q)$, it is enough to prove that $\mathsf{transport}\ (\lambda\, i \to A \to_* P\, i)\ (f\,,p) \equiv (g\,,q)$. The transport only acts on p and q, so $\mathsf{fst}\ (\mathsf{transport}\ (\lambda\, i \to A \to_* P\, i)\ (f\,,p)) \equiv g$ holds by h. For the second components, we are reduced to proving that $p \cdot r \equiv h\ a_0 \cdot q$. This is true immediately by construction of r. ◀

The following lemma is proved in a similar manner.

▶ **Lemma 28.** *Let* A *and* B *be two pointed types with* B *homogeneous. The type* $A \to_* B$ *is homogeneous.*

▶ **Lemma 29.** K_n is homogeneous.

Proof. Let $x : \mathsf{K}_n$. We show that $(\mathsf{K}_n\,,0_k) \simeq_* (\mathsf{K}_n\,,x)$. The function $\lambda\, y \to x +_k y$ is an equivalence. Clearly, $f\ 0_k \equiv x$, and thus we are done. ◀

Proof of Lemma 13. When $m = 0$, the result is obvious, so assume $m \geq 1$. We want to show that $\mathsf{K}_m \to_* \mathsf{K}_{n+m}$ is an n-type. This is equivalent to proving that $\Omega^{n+1}\,(\mathsf{K}_m \to_* \mathsf{K}_{n+m})$ is contractible. We get

$$\Omega^{n+1}\,(\mathsf{K}_m \to_* \mathsf{K}_{n+m}) \simeq (\mathsf{K}_m \to_* \Omega^{n+1}\,\mathsf{K}_{n+m}) \simeq (\mathsf{K}_m \to_* \mathsf{K}_{m-1})$$

Hence, we only need to verify that $\mathsf{K}_m \to_* \mathsf{K}_{m-1}$ is contractible.

To prove this, we choose $c = ((\lambda\, x \to 0_k)\,,\mathsf{refl})$ as the center of contraction. Let $(f\,,p)$ be another pointed function. We want to show that $c = (f\,,p)$. Since K_{m-1} is homogeneous, it is enough, by Lemma 27 and function extensionality, to show that $f\ |\,x\,| \equiv 0_k$ for $x : \mathbb{S}^m$. This is an $(m-2)$-type, so by sphere elimination, it is enough to show that $f\ 0_k \equiv 0_k$. Since f is assumed to be pointed, we are done. ◀

Proof of Lemma 14. Immediate corollary of Lemmas 27–29. ◀

Proof of Lemma 16. We show that $\lambda\, y \to \sigma_{n+m}(x \smile_k y)$ and $\lambda\, y \to \mathsf{cong}\,(\smile_k y)\ (\sigma_n\ x)$ are equal as pointed functions of type $\mathsf{K}_m \to_* \Omega\,\mathsf{K}_{n+m+1}$. Since $\Omega\,\mathsf{K}_{n+m+1} \simeq \mathsf{K}_{n+m}$, this is an n-type by Lemma 13, and we may assume that x is on the form $|\,a\,|$ for $a : \mathbb{S}^n$. Applying

Lemma 14, we only need to show that $\sigma_{n+m}(|a| \smile_k y) \equiv \mathsf{cong}\,(\smile_k y)\,(\sigma_n\,|a|)$ for $y : \mathsf{K}_m$. We have

$$\begin{aligned}
&\mathsf{cong}\,(\smile_k y)\,(\sigma_n\,|a|) \\
&\equiv \mathsf{cong}\,(\lambda x \to |x| \smile_k y)\,(\mathsf{merid}\ a) \cdot \mathsf{cong}\,(\lambda x \to |x| \smile_k y)\,(\mathsf{merid}\ \ast_{\mathbb{S}^n})^{-1} \\
&\stackrel{\text{def}}{\equiv} \sigma_{n+m}(|a| \smile_k y) \cdot \sigma_{n+m}(0_k \smile_k y)^{-1} \\
&\equiv \sigma_{n+m}(|a| \smile_k y)
\end{aligned}$$ ◀

For graded commutativity, we need the following two lemmas.

▶ **Lemma 30.** *Let A be a pointed type and $p : \Omega^2\,A$. We have that*
- $(\lambda\, i\, j \to p\, j\, i) \equiv p^{-1}$
- $(\lambda\, i\, j \to p\,(\sim i)\,(\sim j)) \equiv p$
- $(\lambda\, i\, j \to p\, i\,(\sim j)) \equiv p^{-1}$

Proof. We begin by proving that $(\lambda\, i\, j \to p\, j\, i) \equiv p^{-1}$. We generalize the lemma. Let $q : x \equiv x$ and $p : \mathsf{refl} \equiv q$ for some $x : A$. The key to the proof is to define a suitable path $B : \mathsf{I} \to \mathsf{Type}$ such that $A = \mathsf{PathP}\ (\lambda\, i \to B\ i)\ (\lambda\, i\, j \to p\, j\, i)\ (p^{-1})$ is well-typed. We define B by

$$B\ i = \mathsf{PathP}\ (\lambda\, j \to x \equiv q\,(i \vee j))\ (\lambda\, k \to q\,(i \wedge k))\ \mathsf{refl}$$

By path induction on p, A holds by refl. Fixing $q = \mathsf{refl}$, A reduces to $(\lambda\, i\, j \to p\, j\, i) \equiv p^{-1}$, and we are done.

Verifying that $(\lambda\, i\, j \to p\,(\sim i)\,(\sim j)) \equiv p$ is done in the exact same way, but this time with $B : \mathsf{I} \to \mathsf{Type}$ defined by $B\ i = (\lambda\, j \to q\,(i \vee (\sim j))) \equiv (\lambda\, j \to q\,(i \wedge j))$.

Finally, $(\lambda\, i\, j \to p\, i\,(\sim j)) \equiv p^{-1}$ is given by instantiating the previous equality with $p = p^{-1}$. ◀

▶ **Lemma 31.** *Let $p : \Omega\,\mathsf{K}_n$.* We have $\mathsf{cong}\ \text{-}_k\ p \equiv p^{-1}$

Proof. We prove the lemma for $n \geq 2$. When $n = 0$, it is trivial and when $n = 1$, it follows in an analogous manner. The lemma is easily proved using the encode-decode method. We define a function $f : (x : \mathsf{K}_n) \to 0_k \equiv x \to x \equiv 0_k$ by truncation elimination and sphere induction on x. We let

$$\begin{aligned}
f\ |\mathsf{north}| &= \mathsf{cong}\ \text{-}_k \\
f\ |\mathsf{south}| &= \lambda\, p \to (\mathsf{cong}\ |_|(\mathsf{merid}\ \ast_{\mathbb{S}^{n-1}})^{-1} \cdot \mathsf{cong}\ \text{-}_k\ p) \\
f\ |\mathsf{merid}\ a\ i| &= \ldots
\end{aligned}$$

The $\mathsf{merid}\ a$ case boils down to proving that for any $p : 0_k \equiv |\mathsf{south}|$, we have that

$$\begin{aligned}
&\mathsf{transport}\ (\lambda\, i \to |\mathsf{merid}\ a\ i| \equiv 0_k) \\
&\qquad (f\ |\mathsf{north}|\ (\mathsf{transport}(\lambda\, i \to 0_k \equiv |\mathsf{merid}\ a\ (\sim i)|)\ p)
\end{aligned}$$

or, equivalently,

$$\mathsf{cong}\ |\_|\ (\mathsf{merid}\ a)^{-1} \cdot \mathsf{cong}\ \text{-}_k\ p \cdot (\mathsf{cong}\ |\_|(\mathsf{merid}\ a \cdot (\mathsf{merid}\ \ast_{\mathbb{S}^{n-1}})^{-1})) \tag{2}$$

is equal to $f\ |\mathsf{south}|\ p$. Swapping the last two components in (2) using the fact that path composition in $\Omega\,\mathsf{K}_n$ is commutative, we may cancel out $\mathsf{merid}\ a$, and we are done. Thus, f is defined.

We now have that that $f\ x\ p \equiv p^{-1}$ for any $x : \mathsf{K}_n$ and $p : 0_k \equiv x$ by path induction on p. In particular, $f\ |\mathsf{north}|\ p \stackrel{\text{def}}{\equiv} \mathsf{cong}\ \text{-}_k\ p \equiv p^{-1}$ for $p : \Omega\,\mathsf{K}_n$. ◀

Proof of Proposition 18. When n or m is equal to 0, the proof is easy. We sketch the argument for $n, m \geq 1$. Let $y : \mathsf{K}_m$. The goal is to prove that $\lambda\, x \to x \smile_k y$ and $\lambda\, x \to -_k{}^{m \cdot n}(y \smile_k x)$ are equal as pointed functions. We apply Lemmas 13 and 14 and we are reduced to showing that $|\,a\,| \smile_k y \equiv -_k{}^{m \cdot n}(y \smile_k |\,a\,|)$, ignoring pointedness, for $a : \mathbb{S}^n$. We temporarily fix a and now abstract over y instead. We generalize the problem to that of proving that for all $y : \mathsf{K}_m$, we have that $\lambda\, a \to |\,a\,| \smile_k y$ and $\lambda\, a \to -_k{}^{m \cdot n}(y \smile_k |\,a\,|)$ are equal, now seen as pointed functions of type $\mathbb{S}^n \to_* \mathsf{K}_{n+m}$. Since this is equivalent to $\Omega^n\, \mathsf{K}_{n+m} \simeq \mathsf{K}_m$, this is an m-type, and we may thus let $y = |\,b\,|$ for some $b : \mathbb{S}^m$. We may again ignore pointedness at this stage, by Lemma 14, and we are thus reduced to proving that $|\,a\,| \smile_k |\,b\,| \equiv -_k{}^{m \cdot n}(|\,b\,| \smile_k |\,a\,|)$ for $a : \mathbb{S}^n$, $b : \mathbb{S}^m$.

The case when $n = m = 1$ boils down to proving that

$$\lambda\, i\, j \to |\,\mathsf{loop}\ i\,| \smile_k |\,\mathsf{loop}\ j\,| \equiv \lambda\, i\, j \to -_k\ (|\,\mathsf{loop}\ j\,| \smile_k |\,\mathsf{loop}\ i\,|)$$

viewed as elements of $\Omega^2\, \mathsf{K}_2$ (here, we are ignoring transports and coherence paths). This is immediate by Lemmas 30 and 31. In `Cubical Agda`, we can also verify this computationally by noting that the equivalence $\Omega^2\, \mathsf{K}_2 \simeq \mathbb{Z}$ sends both paths to 1.

We now do the same thing for the case $n, m \geq 2$ (the case $n = 1$ and $m \geq 1$ is close to identical). We may assume as our inductive hypothesis that the statement holds for all $n', m' : \mathbb{N}$ such that $n' + m' < n + m$. This time, the proof boils down to showing that

$$\lambda\, i\, j \to |\,\mathsf{merid}\ a\ i\,| \smile_k |\,\mathsf{merid}\ b\ j\,| \equiv \lambda\, i\, j \to -_k{}^{m \cdot n}\ (|\,\mathsf{merid}\ b\ j\,| \smile_k |\,\mathsf{merid}\ a\ i\,|)$$

again ignoring coherence paths and transports. Here, $a : \mathbb{S}^{n-1}$ and $b : \mathbb{S}^{m-1}$. We fix i and j and give a rough outline of the argument. We have

$$\begin{aligned}
|\,\mathsf{merid}\ a\ i\,| \smile_k |\,\mathsf{merid}\ b\ j\,| &\equiv \sigma_{n+m-1}(|\,a\,| \smile_k |\,\mathsf{merid}\ b\ j\,|)\ i \\
&\equiv -_k{}^{m \cdot (n-1)}\ (\sigma_{n+m-1}(|\,\mathsf{merid}\ b\ j\,| \smile_k |\,a\,|)\ i) \\
&\equiv -_k{}^{m \cdot (n-1)}\ (\sigma_{n+m-1}(\sigma_{n+m-2}(|\,b\,| \smile_k |\,a\,|)\ j)\ i) \\
&\equiv -_k{}^{m \cdot (n-1)} -_k{}^{(n-1) \cdot (m-1)}\ (\sigma_{n+m-1}(\sigma_{n+m-2}(|\,a\,| \smile_k |\,b\,|)\ j)\ i) \\
&\equiv -_k{}^{n+1}\ (\sigma_{n+m-1}(\sigma_{n+m-2}(|\,a\,| \smile_k |\,b\,|)\ j)\ i) \\
&\equiv -_k{}^{n+1}\ (\sigma_{n+m-1}(|\,\mathsf{merid}\ a\ j\,| \smile_k |\,b\,|)\ i) \\
&\equiv -_k{}^{n+1} -_k{}^{(m-1) \cdot n}\ (\sigma_{n+m-1}(|\,b\,| \smile_k |\,\mathsf{merid}\ a\ j\,|)\ i) \\
&\equiv -_k{}^{n+1} -_k{}^{(m-1) \cdot n}\ (|\,\mathsf{merid}\ b\ i\,| \smile_k |\,\mathsf{merid}\ a\ j\,|) \\
&\equiv -_k{}^{m \cdot n+1}\ (|\,\mathsf{merid}\ b\ i\,| \smile_k |\,\mathsf{merid}\ a\ j\,|) \\
&\equiv -_k{}^{m \cdot n}\ (|\,\mathsf{merid}\ b\ j\,| \smile_k |\,\mathsf{merid}\ a\ i\,|)
\end{aligned}$$

The last equality comes from Lemmas 31 and 30. The rest of the steps are just unfoldings of the definition of $\smile_k$, applications of the the inductive hypothesis and implicit uses of Lemma 30 and the fact that $\sigma_n(-_k\ x) \equiv \mathsf{cong}\ -_k\ (\sigma_n\ x)$.

We note that although this informal argument is easy, the formal version is much more technical since we also have to verify that the proof sketched above respects the boundary constraints (i.e. our choices of paths for the point constructors). As we also need to express many of these equalities using PathP or transport (over paths in $\mathbb{N}$), things become even more complicated. ◀

A.3 Proofs for Section 5

The trivial cohomology groups of spheres are easily handled in a similar manner to the non-trivial ones.

▶ **Proposition 32.** *$H^n(\mathbb{S}^m) \simeq \mathbb{1}$ for $n, m \geq 1$ with $n \neq m$.*

Proof. By **Suspension** (see Appendix B) and induction on n and m, it suffices to prove the statement for the cases (a) $n = 1$, $m \geq 2$ and (b) $m = 1$, $n \geq 2$.

For case (a), we note that $\mathsf{K}_1 \simeq \| \mathsf{K}_1 \|_{m-1}$ since K_1 is a 1-type and $m \geq 2$. Let $| f | : \| \mathbb{S}^m \to \| \mathsf{K}_1 \|_{m-1} \|_0$. We prove that $| f | \equiv 0_h$. This is a proposition, so by 0-connectedness of K_1, we may assume that f $\mathsf{base} \equiv | 0_k |$. But by the definition of $(m-1)$-truncations, any map $f : \mathbb{S}^m \to \| \mathsf{K}_1 \|_{m-1}$ is constant. Hence $| f | \equiv 0_h$ and consequently $H^1(\mathbb{S}^m) \simeq \| \mathbb{S}^m \to \| \mathsf{K}_1 \|_{m-1} \|_0$ is contractible.

For case (b), we get, in the same way as in the proof of Proposition 19, that

$$\| \mathbb{S}^1 \to \mathsf{K}_n \|_0 \simeq \| \mathsf{K}_n \times \Omega\, \mathsf{K}_n \|_0 \simeq \| \mathsf{K}_n \|_0 \times \| \mathsf{K}_{n-1} \|_0$$

This type is contractible since K_n is 0-connected for $n > 0$. ◀

We note that part (a) of the proof above can be generalized for any $n, m \geq 1$ such that $n < m$. This gives a short and computationally efficient proof of this special case.

Proof of Lemma 23. We induct on n (assuming $n \geq 0$ as the case $n < 0$ is completely symmetric). When n is 0 or 1, the statement is trivial. The crucial case is when $n = 2$. We need to show that $| (0_k , \mathsf{loop} \cdot \mathsf{loop}) | \equiv | (0_k , \mathsf{refl}) |$. Naturally, their first components agree. However, we do not prove this by refl. Instead we prove that $0_k \equiv 0_k$ by loop. By the characterization of paths over dependent sums, we need to show that $\mathsf{cong}_2 +_k \mathsf{loop}\ \mathsf{loop} \equiv \mathsf{loop} \cdot \mathsf{loop}$, which is immediate by construction.

It is not a priori obvious how to define the inductive step. The goal is to define an operation $\diamond$ on $\| \sum_{x:\mathsf{K}_1} (x +_k x \equiv 0_k) \|_0$ such that for $p, q : \Omega\, \mathsf{K}_1$, we have

$$| (0_k , p) | \diamond | (0_k , q) | \equiv | (0_k , p \cdot q) | \tag{3}$$

Suppose we have two terms $| (| a | , p) |$ and $| (| b | , q) |$ of type $\| \sum_{x:\mathsf{K}_1} (x +_k x \equiv 0_k) \|_0$. Since this type is a set, we may apply Lemma 8 in order to define $\diamond$. We define

$$| (| a | , p) | \diamond_l | (0_k , q) | = | (| a | , p \cdot q) | \qquad | (0_k , p) | \diamond_r | (| b | , q) | = | (| b | , q \cdot p) |$$

To complete the definition of $\diamond$, Lemma 8 requires us to prove that $| (0_k , p \cdot q) | \equiv | (0_k , q \cdot p) |$. This follows immediately by commutativity of $\Omega\, \mathsf{K}_1$. The fact that $\diamond$ satisfies (3) follows from the left path in Lemma 8. The inductive step is now easy to complete. We have

$$| (0_k , \mathsf{loop}^{n+2}) | \equiv | (0_k , \mathsf{loop}^n) | \diamond | (0_k , \mathsf{loop}^2) | \equiv | (0_k , \mathsf{loop}^n) | \diamond | (0_k , \mathsf{refl}) | \equiv | (0_k , \mathsf{loop}^n) |$$

and we are done by the inductive hypothesis. ◀

B The Eilenberg-Steenrod axioms for cohomology

A common approach in classical mathematics is to work abstractly with cohomology using the Eilenberg-Steenrod axioms [16]. The goal of this appendix is to verify that our definition of cohomology is a well-behaved cohomology theory and satisfies a variation of these axioms.

To this end we can also define a *reduced* version which we denote by $\widetilde{H}^n(A)$. This cohomology theory is often preferred in classical algebraic topology as it avoids some exceptional cases, which simplify statements [19]. Given a pointed type A let

$$\widetilde{H}^n(A) = \| A \to_* \mathsf{K}_n \|_0$$

It is easy to prove that the following map is an equivalence for $n \geq 1$.

$$\varphi : H^n(A) \to \widetilde{H}^n(A)$$
$$\varphi \mid f \mid = \mid \lambda\, x \to (f\ x -_k f\ *_A\ ,\ \mathsf{rCancel}_k\ (f\ *_A)) \mid$$

Using this equivalence, the group structure on $\widetilde{H}^n(A)$ can be induced from the group structure on $H^n(A)$ using the SIP. One may also define it directly. This is more subtle, as the group laws also have to respect the pointedness proofs, but it turns out to be straightforward with our definition of the group structure on K_n. In the formalization, this is how the group structure on $\widetilde{H}^n(A)$ is defined. Interestingly, in a previous attempt to give a direct definition of this group structure using the definition of $+_k$ from [5, Prop. 5.1.4], it was difficult to get `Cubical Agda` to typecheck in reasonable time without using the abstract keyword.

B.1 The axioms in HoTT/UF

The Eilenberg-Steenrod axioms have been studied previously in HoTT/UF by [11, 8, 39]. To state the **Exactness** axiom, we need to introduce cofibers (also known as *mapping cones*).

▶ **Definition 33** (Cofiber). *Given $f : A \to B$, we define the* cofiber *of f, denoted $\mathsf{coFib}\ f$, as the pushout of the span $\mathbb{1} \xleftarrow{\lambda\, x \to *_{\mathbb{1}}} A \xrightarrow{f} B$. We write cfcod for the right inclusion $\mathsf{inr} : B \to \mathsf{coFib}\ f$.*

With this, we can state the Eilenberg-Steenrod axioms:

▶ **Definition 34.** *A family of contravariant functors $E^n : \mathsf{Type}_* \to \mathsf{AbGrp}$ indexed by $n : \mathbb{Z}$ is an ordinary (reduced) cohomology theory if the following axioms are satisfied.*

Suspension: *For $A : \mathsf{Type}_*$, there is a group isomorphism $E^n\ A \cong E^{n+1}\ (\mathsf{Susp}\ A)$. Furthermore, this isomorphism is natural with respect to Susp.*

Exactness: *For $f : A \to_* B$ there is an exact sequence:*

$$E^n\ (\mathsf{coFib}\ f) \xrightarrow{E^n\ \mathsf{cfcod}} E^n\ B \xrightarrow{E^n\ f} E^n\ A$$

Dimension: *For $n : \mathbb{Z}$ with $n \neq 0$, $E^n\ \mathbb{S}^0$ is trivial.*

Here, "ordinary" refers to the fact that E^n satisfies the **Dimension** axiom. Let $f^* = E^n\ f$ and $\mathsf{cfcod}^* = E^n\ \mathsf{cfcod}$. The sequence in the **Exactness** axiom is *exact* if the kernel of f^* (the elements of $E^n\ B$ that get mapped to $0 : E^n\ A$) is equal to the image of cfcod^*. As $E^n\ A$ is a set, the statement "b is in the kernel of f^*" is a proposition. Univalence then implies that **Exactness** follows if all $b : E^n\ B$ are in the kernel of f^* iff they are in the image of cfcod^*. One often also consider a further axiom:

Additivity: For $I : \mathsf{Type}$ and a family of types A_i with $i : I$, we have an isomorphism:

$$E^n\ \left(\bigvee_{i:I} A_i\right) \cong ((i : I) \to E^n\ A_i)$$

Proving this typically requires that the index set I satisfies the set theoretic axiom of choice [8]. As we are interested in computations, we do not rely on this general form. Instead, the following version is sufficient for all examples we consider:

Binary Additivity: For $n : \mathbb{Z}$ and $A, B : \mathsf{Type}_*$ the following groups are isomorphic:

$$E^n\ (A \vee B) \cong (E^n\ A \times E^n\ B)$$

B.2 Verifying the axioms

It is possible to directly show that $\widetilde{H}^n$ satisfies the axioms. However, it turns out that when working formally, unreduced cohomology H^n is often easier to work with, as it avoids pointed types. The only caveat is that **Exactness** fails for H^0. We therefore show that the axioms hold for the following equivalent cohomology theory:

$$\widehat{H}^n(A) = \begin{cases} \mathbb{1} & \text{if } n < 0 \\ \widetilde{H}^0(A) & \text{if } n = 0 \\ H^n(A) & \text{if } n > 0 \end{cases}$$

As $\widehat{H}^n(A)$ is isomorphic to $\widetilde{H}^n(A)$, the SIP implies that it suffices to show that the axioms hold for $\widehat{H}^n(A)$ in order to show that $\widetilde{H}^n(A)$ also satisfies them.

▶ **Proposition 35.** *$\widehat{H}^n$ is an ordinary reduced cohomology theory.*

Proof (sketch). We verify the axioms, omitting trivial cases and details to the formalization.

Suspension: The proof is almost identical for $n = 0$ and $n > 0$, so we focus on the latter. Given $f : \mathsf{Susp}\, A \to \mathsf{K}_{n+1}$ we get $f' : A \to \Omega\, \mathsf{K}_{n+1}$ sending $a : A$ to

$$p^{\,\text{-}1} \cdot \mathsf{cong}\ (\lambda\, x \ \to\ f\ x \,\text{-}_k\, f\ 0_k)\ (\mathsf{merid}\ a \cdot (\mathsf{merid}\ *_A)^{\,\text{-}1}) \cdot p$$

where $p = \mathsf{rCancel}_k\,(f\ 0_k)$. By pointwise application of Theorem 9, this gives us a map $\varphi : H^{n+1}(\mathsf{Susp}\ A) \to H^n(A)$, sending $|\,f\,|$ to $|\,\lambda\, x \ \to\ \sigma_n^{\,\text{-}1}\,(f'\ x)\,|$. The inverse is defined analogously. The fact that this is an isomorphism is technical but straightforward using that σ_n is an equivalence.

When $n = -1$, we need to prove that $\widetilde{H}^0\,(\mathsf{Susp}\ A)$ is contractible for pointed types A. This is immediate: any function $f : \mathsf{Susp}\, A \to \mathbb{Z}$ is uniquely determined by f north because $f\,\mathsf{south} \equiv f\,\mathsf{north}$ must hold by $\mathsf{merid}\ *_A$, and $\mathsf{cong}\ f\ (\mathsf{merid}\ x) \equiv \mathsf{cong}\ f\ (\mathsf{merid}\ y)$ holds for any $x, y : A$ since $\mathbb{Z}$ is a set.

Naturality of these isomorphisms follows immediately by construction. It even holds definitionally modulo induction on n, truncation elimination and pattern matching on $\mathsf{Susp}\ A$.

Exactness: This proof is also almost identical for $n = 0$ and $n > 0$, so we focus on the latter again. It suffices to check that all $|\,g\,| : H^n(B)$ are in the kernel of f^* iff they are in the image of cfcod^*. For the left to right direction, assume that we have $p' : f^*\ |\,g\,| \equiv 0_h$. We are proving a proposition, and we may thus apply the induction principle for (set) truncated paths to p'. This gives a path $p : g \circ f \equiv \lambda\, x \to 0_k$ and we define $h : \mathsf{coFib}\ f \to \mathsf{K}_n$ by:

$$\begin{aligned} &h\ (\mathsf{inl}\ *_{\mathbb{1}}) = 0_k \\ &h\ (\mathsf{cfcod}\ x) = g\ x \\ &h\ (\mathsf{push}\ a\ i) = p\ ({\sim}\, i)\ a \end{aligned}$$

This satisfies $\mathsf{cfcod}^*\ |\,h\,| \equiv |\,g\,|$ definitionally and hence $|\,g\,|$ is in the image of cfcod^*. The other direction is proved similarly.

Dimension: The only non-trivial case is $n > 0$. It suffices to prove that for $|\,f\,| : H^n(\mathbb{S}^0)$, we have $|\,f\,| \equiv 0_h$. Since K_n is 0-connected in this case and $|\,f\,| \equiv 0_h$ is a proposition, we may assume that $f\ \mathsf{true} \equiv f\ \mathsf{false} \equiv 0_k$ and thereby we are done by function extensionality. ◀

The binary additivity axiom also holds.

▶ **Proposition 36.** *$\widehat{H}^n$ satisfies **Binary Additivity**.*

Proof. For $n = 0$, the intuition is that $\widetilde{H}^0(A \vee B)$ consists of pairs of functions $f : A \to \mathbb{Z}$ and $g : B \to \mathbb{Z}$ with a path $p : f \ *_A \ \equiv g *_A$ and a proof of pointedness $q : f \ *_A \ \equiv 0$. The path q tells us that f is pointed, and by composition with p we may also deduce that g is pointed. Hence, we get a homomorphism $\phi : \widetilde{H}^0(A \vee B) \to \widetilde{H}^0(A) \times \widetilde{H}^0(B)$. We can easily deduce that ϕ is an isomorphism from the fact that $\mathbb{Z}$ is a set, and thus ϕ preserves p and q trivially.

When $n \geq 1$ we can define a homomorphism by:

$$\phi : H^n(A \vee B) \to H^n(A) \times H^n(B)$$
$$\phi \mid f \mid = (\mid f \circ \mathsf{inl} \mid , \mid f \circ \mathsf{inr} \mid)$$

This map simply forgets that $f\,(\mathsf{inl} *_A) \equiv f\,(\mathsf{inr} *_B)$ holds. The topological intuition here is that this path always can be contracted by continuously varying the choice of points $f(\mathsf{inl} *_A)$ and $f(\mathsf{inr} *_B)$. We define the inverse by

$$\psi : H^n(A) \times H^n(B) \to H^n(A \vee B)$$
$$\psi\ (\mid f \mid , \mid g \mid) = \mid f \vee g \mid$$

where $f \vee g : A \vee B \to \mathsf{K}_n$ is defined by

$$(f \vee g)\ (\mathsf{inl}\, x) = f\ x +_k g *_B$$
$$(f \vee g)\ (\mathsf{inr}\, x) = f\ \ *_A\ +_k g\ x$$
$$(f \vee g)\ (\mathsf{push}\ *_{\mathbb{1}}\ \ i) = f\ \ *_A\ +_k g *_B$$

The fact that $\phi\,(\psi\, x) \equiv x$ holds is easy—since the statement is a proposition, we may assume, for any pair of functions $f : A \to \mathsf{K}_n$ and $g : B \to \mathsf{K}_n$, that $f\ \ *_A\ \equiv g\ *_B\ \equiv 0_k$, using the fact that K_n is 0-connected.

For the other direction, again due to 0-connectedness, we may assume that we have a path $\ell : 0_k \equiv f\,(\mathsf{inl} *_A)$. Under this assumption, we prove that $f\ c \equiv ((f \circ \mathsf{inl}) \vee (f \circ \mathsf{inr}))\ c$ by induction on $c : A \vee B$. For $c = \mathsf{inl}\, a$, we need to prove that $f\,(\mathsf{inl}\, a) \equiv f\,(\mathsf{inl}\, a) +_k f\,(\mathsf{inr} *_B)$. We use the following construction

$$\mathsf{P} : (x : \mathsf{K}_n)\ \{y\ z : \mathsf{K}_n\} \to 0_k \equiv y \to y \equiv z \to x \equiv x +_k z$$
$$\mathsf{P}\ x\ p\ q = (\mathsf{rUnit}_k\ x)^{\text{-1}} \cdot (\lambda\, i \to x +_k p\ i) \cdot (\lambda\, i \to x +_k q\ i)$$

and are done by $\mathsf{P}\ (f\ (\mathsf{inl}\, a))\ \ell\ (\mathsf{cong}\ f\ (\mathsf{push}\ *_{\mathbb{1}}))$.

For $c = \mathsf{inr}\, b$, the goal is $f\,(\mathsf{inr}\, b) \equiv f\,(\mathsf{inl} *_A) +_k f\,(\mathsf{inr}\, b)$. We define

$$\mathsf{Q} : (x : \mathsf{K}_n)\ \{y\ : \mathsf{K}_n\} \to 0_k \equiv y \to x \equiv y +_k x$$
$$\mathsf{Q}\ x\ p = (\mathsf{lUnit}_k\ x)^{\text{-1}} \cdot (\lambda\, i \to p\ i +_k x)$$

and are done by $\mathsf{Q}\ (f\ (\mathsf{inr}\, b))\ \ell$.

For $c = \mathsf{push}\ *_{\mathbb{1}}\ \ i$, we need to construct a filler of type

$$\mathsf{PathP}\ (\lambda\, i \to \mathsf{P'}\ i \equiv \mathsf{Q'}\ i)\ (\mathsf{cong}\ f\ (\mathsf{push}\ *_{\mathbb{1}}))\ \mathsf{refl} \tag{4}$$

where $\mathsf{P'} = \mathsf{P}\ (f\ (\mathsf{inl} *_A))\ \ell\ (\mathsf{cong}\ f\ (\mathsf{push}\ *_{\mathbb{1}}))$ and $\mathsf{Q'} = \mathsf{Q}\ (f\ (\mathsf{inr} *_B))\ \ell$. In order to do this, we generalize and ask that for arbitrary $x,\ y : \mathsf{K}_n$ and paths $p : 0_k \equiv x$ and $q : x \equiv y$, there is a filler of the square

$$\square_{p,q} : \mathsf{PathP}\ (\lambda\, i \to \mathsf{P}\ x\ p\ q\ i \equiv \mathsf{Q}\ y\ p\ i)\ q\ \mathsf{refl}$$

By path induction on p and q, we are done if we can show that $\mathsf{P}\ 0_k\ \mathsf{refl}\ \mathsf{refl} \equiv \mathsf{Q}\ 0_k\ \mathsf{refl} \equiv \mathsf{refl}$. Since $p \equiv q \equiv \mathsf{refl}$, the only non-trivial components of $\mathsf{P}\ 0_k\ \mathsf{refl}\ \mathsf{refl}$ and $\mathsf{Q}\ 0_k\ \mathsf{refl}$ are $(\mathsf{rUnit}_k\ 0_k)^{-1}$ and $(\mathsf{lUnit}_k\ 0_k)^{-1}$ respectively. As remarked in Section 3, these are both (definitionally) equal to refl, and we are done as $\square_{\ell,\mathsf{cong}\ f\ (\mathsf{push}\ *_{\mathbb{1}})}$ is the filler we needed for (4). ◀

The axioms are hence satisfied by H^n for $n > 0$, $\widetilde{H}^n$ for $n \geq 0$, and $\widehat{H}^n$ for all $n : \mathbb{Z}$. This means that they are all well-behaved cohomology theories and we can now do some concrete characterizations using the axioms.

B.3 The Mayer-Vietoris sequence

The Eilenberg-Steenrod axioms are enough for many fundamental constructions in cohomology theory. One important example is the Mayer-Vietoris sequence.

▶ **Theorem 37** (Mayer-Vietoris sequence). *Let E^n be a cohomology theory and D be the pushout of the span $A \xleftarrow{f} C \xrightarrow{g} B$. There is an exact sequence*

$$\cdots \to E^{n-1}\,C \to E^n\,D \to E^n\,A \times E^n\,B \to E^n\,C \to \dots$$

There are many variants of Theorem 37. Cavallo constructed the sequence for general reduced cohomology directly from the Eilenberg-Steenrod axioms in [11], whereas Brunerie constructed a version with alternating reduced and unreduced groups for a cohomology theory similar to ours in [5, Prop. 5.2.2].

Many elementary results about cohomology groups can be deduced from this sequence. For instance, by viewing $\mathbb{S}^{n+1}$ as the pushout of the span $\mathbb{1} \leftarrow \mathbb{S}^n \to \mathbb{1}$ and noting that $\widetilde{H}^n(\mathbb{1}) \cong \mathbb{1}$, we get exact sequences

$$\mathbb{1} \longrightarrow \widetilde{H}^n(\mathbb{S}^n) \xrightarrow{d_{n+1}} \widetilde{H}^{n+1}(\mathbb{S}^{n+1}) \to \mathbb{1}$$

where d_{n+1} is the map from $E^n\,C$ to $E^{n+1}\,D$ in Theorem 37. It is easy to prove that $\widetilde{H}^0(\mathbb{S}^0) \cong \mathbb{Z}$ and get a stable sequence

$$\mathbb{Z} \cong \widetilde{H}^1(\mathbb{S}^1) \cong H^1(\mathbb{S}^1) \cong \widetilde{H}^2(\mathbb{S}^2) \cong H^2(\mathbb{S}^2) \cong \dots$$

With computations in `Cubical Agda` in mind, we prefer not to use proofs such as this one. The problem with proofs from exact sequences is that many constructions become indirect. For instance, the inverse of d_n is induced by the proofs of the exactness properties of the Mayer-Vietoris sequence instead of being constructed directly. We have formalized an unreduced version of the sequence, but have mostly been able to avoid it and instead give direct characterizations of most cohomology groups that we consider.

C Benchmarking computations with the cohomology groups

For every equivalence $\phi : H^n(A) \simeq G$ in Section 5, two benchmarks have been run in `Cubical Agda`. **Test 1** concerns the behavior of ϕ and ϕ^{-1}. The aim was to check whether $\phi\,(\phi^{-1}\,g) \equiv g$ is proved by refl for different values of $g : G$. **Test 2** concerns the behavior of $+_h$ and the aim was to check whether $\phi\,(\phi^{-1}\,g_1 +_h \phi^{-1}\,g_2) \equiv g_1 +_G g_2$ for $g_1, g_2 : G$.

For an example of how the tests were performed, let $\phi : H^1(\mathbb{K}^2) \simeq \mathbb{Z}$. We then measure how long it takes to typecheck that **Test 2** is proved by refl when instantiated with concrete numbers. In the example below we use 1 and 2, and the test took 22ms to terminate, which we record in a comment.

```
test : ϕ (ϕ⁻¹ 1 +ₕ ϕ⁻¹ 2) ≡ 3        -- 22ms
test = refl
```

As we expect similar goals to appear in future formalizations, the tests were run on a regular laptop with 1.60GHz Intel processor and 16GB RAM. The group elements in the tests were made up from integers between -5 and 5. Results of these tests are summarized in the table below. The failed computations, marked with ✗, were manually terminated after 10min. Details and exact timings can be found at `https://github.com/agda/cubical/blob/master/Cubical/Experiments/ZCohomology/Benchmarks.agda`.

Type A	**Cohomology**	**Group** G	**Test 1**	**Test 2**
$\mathbb{S}^1$	H^1	$\mathbb{Z}$	✓	✓
$\mathbb{S}^2$	H^2	$\mathbb{Z}$	✓	✓
$\mathbb{S}^3$	H^3	$\mathbb{Z}$	✓	✗
$\mathbb{S}^4$	H^4	$\mathbb{Z}$	✗	✗
$\mathbb{T}^2$	H^1	$\mathbb{Z} \times \mathbb{Z}$	✓	✓
	H^2	$\mathbb{Z}$	✓	✓
$\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1$	H^1	$\mathbb{Z} \times \mathbb{Z}$	✓	✓
	H^2	$\mathbb{Z}$	✓	✓
$\mathbb{K}^2$	H^1	$\mathbb{Z}$	✓	✓
	H^2	$\mathbb{Z}/2\mathbb{Z}$	✗	✗
$\mathbb{R}P^2$	H^2	$\mathbb{Z}/2\mathbb{Z}$	✗	✗
$\mathbb{C}P^2$	H^2	$\mathbb{Z}$	✓	✓
	H^4	$\mathbb{Z}$	✗	✗

For most spaces considered here, **Test 1** terminates in less than 0.2s. This is a considerable improvement to prior attempts in [28] where the same calculations failed to terminate for both $H^2(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1)$ and $H^2(\mathbb{T}^2)$ (that formalization used $+_h$ from [5] and, for most characterizations, the Mayer-Vietoris sequence). However, **Test 1** fails to terminate for $H^2(\mathbb{K}^2)$, $H^2(\mathbb{R}P^2)$ and $H^4(\mathbb{C}P^2)$. After many optimizations, even $\phi\ 0_h \equiv 0$ can only be verified computationally in `Cubical Agda` for $\mathbb{R}P^2$ (the same test fails for $\mathbb{K}^2$). This is not as surprising as it may seem. For both spaces, ϕ attempts to compute the winding number of a loop in $\Omega \mathsf{K}_1$ which is constructed in terms of the complex proof that $\sigma_2^{-1} : \Omega\, \mathsf{K}_2 \to \mathsf{K}_1$ is a homomorphism. For $\mathbb{K}^2$, this construction also relies on the proof of Theorem 12. Higher cohomology groups of spheres also appear to suffer from the same problems. For $\phi : H^3(\mathbb{S}^3) \simeq \mathbb{Z}$, **Test 2** fails even for $\phi^{-1}\ 0 +_h \phi^{-1}\ 0$.