Synthetic Cohomology Theory in Cubical Agda

***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. We give a new optimized group structure for cohomology with $\mathbb{Z}$ coefficients and verify that it satisfies the Eilenberg-Steenrod axioms. We also characterize the cohomology groups of the spheres, torus, Klein bottle and real projective plane. As all proofs are constructive, we obtain concrete computations which can serve as benchmarks for future implementations.**

I. Introduction

Homotopy Type Theory and Univalent Foundations (HoTT/UF) [1] extends Martin-Löf type theory [2] with Voevodsky's univalence axiom [3] and higher inductive types (HITs). This is based on a close correspondence between types and topological spaces represented as Kan simplicial sets [4]. With this interpretation points in spaces correspond to elements of types, while paths and homotopies correspond to (iterated) identity types between these elements [5]. This enables homotopy theory to be developed *synthetically* using type theory. Many classical results from homotopy theory have been formalized in HoTT/UF using this correspondence: the definition of the Hopf fibration [1], the Blakers-Massey theorem [6], the Seifert-van Kampen theorem [7] and the Serre spectral sequence [8], 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 homology and cohomology have been invented.

Informally, the (co)homology groups of a space X describe its n-dimensional holes. For instance, the n-dimensional hole in the n-sphere $\mathbb{S}^n$ corresponds to its n:th cohomology group being non-trivial. These holes constitute a topological invariant, making (co)homology a powerful technique for establishing which spaces cannot be homotopy equivalent.

The usual formulation of singular (co)homology using (co)chain complexes relies on taking the underlying set of topological spaces when defining the singular (co)chains [9]. This operation is *not* invariant under homotopy equivalence which makes it impossible to directly use when formalizing (co)homology synthetically in HoTT/UF. Luckily, in the case of cohomology, there is a classical homotopy invariant definition using Eilenberg-MacLane spaces which can be formalized in HoTT/UF [10]. This was initially studied by members at the IAS special year on HoTT/UF in 2012–2013 [11] and has since been used in a variety of developments, including working with the Eilenberg-Steenrod axioms [12] and defining cellular cohomology [13]. This paper builds on some of this prior work, but uses `Cubical Agda`—a recent extension of the dependently typed programming language `Agda` [14] with *cubical* features [15].

The `Cubical Agda` system is based on a variation of cubical type theory formulated by Coquand et al. [16]. 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 Martin-Löf identity type [17] 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 [18]. 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 write down a general schema for HITs where the eliminators compute definitionally for higher constructors [19], [20]. This is still an open problem for HoTT/UF and HITs have to be added axiomatically which leads to many bureaucratic transports that complicate proofs.

Mörtberg and Pujet explored practical implications of formalizing synthetic homotopy theory in `Cubical Agda` in [21]. This work led to empirical evidence for the above claims. For instance, the proof of the 3×3 lemma for pushouts was shortened from 3000 lines of code in `HoTT-Agda` [22] 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 for higher constructors [23], [24]. With the additional computation rules of `Cubical Agda`, this proof is now trivial [15, Section 2.4.1].

The present paper is a natural continuation of the prior work on formalizing synthetic homotopy theory in `Cubical Agda`. The two main goals are:

1) to *characterize* $\mathbb{Z}$ cohomology groups of types, and
2) to *compute* using these cohomology groups.

In classical algebraic topology the terms *characterize* and *compute* are often used interchangeably when discussing (co)homology. We are careful to distinguish these two notions. When *characterizing* the cohomology groups of some type we prove that it is isomorphic to some other group. As all of our proofs are constructive, we can then use `Cubical Agda` to actually *compute* with these isomorphisms.

The first of the two goals above is especially interesting because the cubical proofs almost never rely on path induction, but rather use the cubical primitives directly. This leads to new proofs which are sometimes closer to their topological counterparts. The second goal is related to a major open problem in HoTT/UF—the computation of Brunerie's 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 [25] that the absolute value of this number is 2, but even though this is a constructive definition, it has so far proved infeasible to compute using cubical type theory despite considerable efforts.

Having the possibility to do proofs simply by computation is one of the most appealing aspects of developing synthetic homotopy theory in cubical type theory. As this is not always possible in HoTT/UF, one often has to resort to doing long calculations by hand, which leads to complex proofs. If proofs instead can be carried out purely using computational means, many of these long calculations become obsolete. This is a reason why many proofs from synthetic homotopy theory are substantially shorter in `Cubical Agda`. The Brunerie number is currently the main example of an interesting number which has proved infeasible to compute. Developing cohomology theory in `Cubical Agda` allows us to invent many more examples of such numbers. This should help identify where the bottlenecks are and, in the long run, increase the computational capabilities of `Cubical Agda` and similar systems.

Outline The paper is organized as follows:

- Section II details the `Cubical Agda` formalization of the notions from HoTT/UF which the rest of the paper uses.
- Section III defines $\mathbb{Z}$ cohomology and its basic properties.
- Section IV verifies that the definition satisfies the Eilenberg-Steenrod axioms for cohomology theories. These are then used to derive the Mayer-Vietoris sequence and to characterize the cohomology groups of $\mathbb{S}^n$.
- Section V shows how the cohomology groups of various types can be characterized directly without using the abstract machinery of Section IV. Among these are the first synthetic characterizations of the cohomology groups of the Klein bottle and real projective plane.
- Section VI collects benchmarks of computations performed using the cohomology groups from Section V.
- Section VII ends the paper with a comparison of related work and concluding remarks.

All results have been formalized in the `agda/cubical` library available at https://github.com/agda/cubical/. Instructions for how to install `Agda` and the cubical library are available at this URL. 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 and conventions. In order to clarify the connection 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.

II. Homotopy Type Theory in Cubical Agda

The `Agda` system [14] 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 $f(x)$.

The `Agda` system 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 and their formalization in `Cubical Agda` which the rest of the paper relies on. Because of space constraints we omit many technical details and refer curious readers to the paper of Vezzosi et al. [15] for a comprehensive technical treatment of all of the features of `Cubical Agda`.

A. 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 $[0,1] \subset \mathbb{R}$ in homotopy theory. However, in `Cubical Agda` this is a purely formal object. A variable $i : \mathsf{I}$ represents a point varying continuously between the endpoints. The interval is equipped with three operations: *minimum* ($\_\wedge\_ : \mathsf{I} \to \mathsf{I} \to \mathsf{I}$), *maximum* ($\_\vee\_ : \mathsf{I} \to \mathsf{I} \to \mathsf{I}$) and *reversal* ($\sim\_ : \mathsf{I} \to \mathsf{I}$).

A function out of I into one of `Agda`'s universes of types represents a line between two types. By iterating this, we obtain squares, cubes and hypercubes of types making `Agda` inherently *cubical*. Universes in `Agda` are written[1] $\mathsf{Type}\,\ell$ where ℓ is a *universe level*. In order to ease notation, we omit universe levels in the paper. It is often useful to specify the endpoints of a line. This is done via *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 using lambda abstractions:

$$\lambda i \to t : \mathsf{PathP}\ A\ t[\mathsf{i0}\,/\,i]\ t[\mathsf{i1}\,/\,i]$$

provided that $t : A\ i$ for $i : \mathsf{I}$. 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 [1, Sect. 6.2]. Given $A : \mathsf{Type}$ we define the type of non-dependent paths in A using PathP as follows:

$$\begin{aligned} &\_\equiv\_ : A \to A \to \mathsf{Type} \\ &\_\equiv\_\ x\ y = \mathsf{PathP}\ (\lambda\ \_ \to A)\ x\ y \end{aligned}$$

Representing equalities as paths allows us to directly reason about equality. For instance, the constant path represents a proof of reflexivity:

$$\begin{aligned} &\mathsf{refl} : \{x : A\} \to x \equiv x \\ &\mathsf{refl}\ \{x = x\} = \lambda\ i \to x \end{aligned}$$

[1] Readers familiar with `Agda` will note that we rename Set to Type.

Here, the syntax $\{x = x\}$ tells `Cubical Agda` to bind the implicit argument x (first x) to a variable x (second x) that can be used on the right hand side of the definition.

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 : {x y : A} {B : A → Type} (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 in HoTT/UF 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`. For example, function extensionality has a very simple proof:

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

The proof of function extensionality for dependent and n-ary functions is equally direct. Since funExt is a *definable notion* in `Cubical Agda` it has computational content: it simply swaps the arguments to p.

We can also use the operations on I to construct various useful operations. For example, the reversal of a path is defined using ∼_ and represents the fact that _≡_ is symmetric.

```
_⁻¹ : x ≡ y → y ≡ x
p ⁻¹ = λ i → p (∼ i)
```

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

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

The substitution principle is an instance of 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:

$$\lambda\, i \to B\ (p\ i) : B\ x \equiv B\ y$$

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 holding 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 for composing paths and, more generally, for composing higher dimensional cubes. This operation is called *homogeneous composition* and is written hcomp. An important special case is binary composition of paths $\_\cdot\_ : x \equiv y \to y \equiv z \to x \equiv z$. The dependent version for PathP (over _·_) is written _·'_. By composing paths and higher cubes using hcomp, we can reason about equalities/paths in a very direct way, avoiding the use of path induction.

B. *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 (of implicit level ℓ). Given two pointed types A and B, a pointed function is a pair $(f, p) : A \to_* B$, where $f : A \to B$ and $p : f(*_A) \equiv *_B$. Abusing notations, we often leave $*_A$ and p implicit and write $A : \mathsf{Type}_*$ and $f : A \to_* B$.

Most HITs in [1] can be defined directly using the general schema of `Cubical Agda`. For example, the circle can be defined as a type with a base point and a non-trivial loop connecting base with itself:

```
data 𝕊¹ : Type where
  base : 𝕊¹
  loop : base ≡ base
```

Functions out of HITs are written using pattern-matching equations, just like in regular `Agda`. When typechecking the cases for path constructors, `Cubical Agda` checks that the endpoints of what the user writes match up.

The suspension of a type A is defined as follows:

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

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 defined by recursion:*

$$\mathbb{S}^n = \begin{cases} \mathsf{Bool} & \text{if } n = 0 \\ \mathbb{S}^1 & \text{if } n = 1 \\ \mathsf{Susp}\ \mathbb{S}^{n-1} & \text{if } n > 1 \end{cases}$$

These types are pointed by true, base *and* north.

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 $n > 0$ *we define*

$$\Omega^{n+1}\, A = \Omega\, (\Omega^n\, A)$$

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 $\_\cdot\_$ 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).*
- $(n+1)$*-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 $n > -2$, A is an n-type if $\Omega^{n+1}A$ is contractible for any choice of base point $a : A$. We 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 $n \geq -1$ we rely on the "hub and spoke" construction [1, Section 7.3].[2] 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 $B : \| A \|_n \to n\text{-}\mathsf{Type}$, 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 $B : |x| \equiv |y| \to n\text{-}\mathsf{Type}$ and want to construct $B\ p$. The elimination principle tells us that it suffices to do so under the assumption that we have a path $q : x \equiv y$ in A with $p = \mathsf{cong}\ |\_|\ q$. This is motivated by [1, Theorem 7.3.12].

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

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 algebraic topology and let us define many useful spaces. Given functions $f : A \to B$, $g : A \to C$, the pushout of the span $B \xleftarrow{f} A \xrightarrow{g} C$ is the HIT:

```
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 $\mathsf{inl}\ *_A$.

C. Univalence

One of the most important notions in HoTT/UF is Voevodsky's univalence axiom [3]. 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 $x : B$ [1, Chapter 4.4]. This axiom is a provable theorem in `Cubical Agda` using the Glue types of [16, Section 6]. This implies that there is an underlying function

$$\mathsf{ua} : A \simeq B \to A \equiv B$$

which allows equivalences to be converted to paths without losing computational content. Transporting along a path constructed using ua applies the function e of the 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 *quasi-equivalence* in [1]. It is a corollary of [1, Theorem 4.4.5] that

all quasi-equivalences can be promoted to equivalences. This very useful 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 the pair $(G, \mathsf{isGroup}\ G)$ 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 [26], a notion of *univalent* structure and structure preserving isomorphisms $\cong$ for which it is direct to prove that ua induces a function $\mathsf{sip} : A \cong B \to A \equiv B$ is introduced in `Cubical Agda`. This is one way to formalize the informal *Structure Identity Principle* (SIP) from HoTT/UF [1, 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: the SIP implies that isomorphic groups are path-equal.

We will sometimes take the liberty of referring to types as groups despite their not being sets. This means that the type admits a group structure but may have non-trivial higher homotopies.

III. $\mathbb{Z}$ 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* over G. That is, $K(G,n)$ is the unique space with a single non-trivial homotopy group isomorphic to G, i.e. $\pi_n(K(G,n)) \cong G$. 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. The group structure which we will define here differs from previous variations in that it is optimized for efficient computations.

A. 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 [10]. In this paper, we focus on the case $G = \mathbb{Z}$ and define this special case following Brunerie [25, Def. 5.1.1]:

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

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

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

Proof. When $n = 0$, the statement is trivial as $\mathbb{S}^0$ is pointed. When $n \geq 1$, we want to show that $\| \mathbb{S}^n \|_{n-1}$ is contractible. 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 thus 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 proof of Proposition 1 is much more direct than the one in [25, 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 reason for this speedup seems to be that the general theory about connectedness relies on rather complex proofs involving univalence. In particular, it relies on repeated use of [1, Theorem 7.3.12] which says that the type of paths $|x| \equiv |y|$ over $\| A \|_{n+1}$ is equivalent to $\| x \equiv y \|_n$.

A more substantial deviation from [25] is in the definition of the group structure on K_n. This is defined in [25, Prop. 5.1.4] using $\mathsf{K}_n \simeq \Omega \mathsf{K}_{n+1}$ which itself is proved using the Hopf fibration [1, Section 8.5] when $n = 1$ and the Freudenthal suspension theorem [1, 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 [27] due to the application of the Freudenthal suspension theorem. 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 [28]. 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 [10]. 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 [1, Lemma 8.6.2], but the proof does not rely on general theory about connected types.

Lemma 1. *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*

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

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

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

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

Proof (sketch). 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. The left path is just refl and the right path is easy to construct by sphere 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 induction 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 induction hypothesis. We omit the construction—it is not difficult, but rather technical. □

The general version of Lemma 1 is used for Eilenberg-MacLane spaces over an arbitrary group G in [10]. The advantage of the above form is the definitional reductions that we get from the fact that the lemma is proved by sphere induction. Consequently, we may define $+_k$ so that e.g. $0_k +_k 0_k \equiv 0_k$ holds definitionally. This allows us to make statements and carry out proofs which would otherwise not be well-typed. We define $+_k : \mathsf{K}_n \to \mathsf{K}_n \to \mathsf{K}_n$ as follows.

- When $n = 0$, $+_k$ is simply integer addition.
- When $n = 1$, we define $+_k$ by cases:

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

  where Q is a suitable filler of a square with loop on all sides. This is easily defined by a single hcomp so that $\mathsf{cong}_2\ +_k\ \mathsf{loop}'\ \mathsf{loop}' \equiv \mathsf{loop}' \cdot \mathsf{loop}'$ holds definitionally for the canonical loop $\mathsf{loop}' = \mathsf{cong}\ |\_|\ \mathsf{loop}$ in K_1.
- When $n \geq 2$ we need to construct a map $\mathbb{S}^n \times \mathbb{S}^n \to \mathsf{K}_n$. Because K_n is n-truncated it is also an $(n+n-2)$-Type. By Lemma 1, 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.

We define negation $-_k : \mathsf{K}_n \to \mathsf{K}_n$ as follows:

- When $n = 0$, $-_k$ is simply integer negation.
- When $n \geq 1$, we define $-_k\ x$ for $x : \mathsf{K}_n$ by applying the inverse of $\lambda\, y \to x +_k y$ to 0_k. This requires us to prove this map is an equivalence. Since this is a proposition, it suffices to prove it when x is 0_k. After applying truncation elimination to x, this map is just the identity and we are done.

The fact that $+_k$ and $-_k$ satisfy the group laws follows from Lemma 1. 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 somewhat naive. However, it provably agrees with the definition given by Brunerie in [25, Prop. 5.1.4]. In fact, a simple corollary of Lemma 1 is that there is at most one binary operation on K_n with lUnit_k and rUnit_k such that $\mathsf{lUnit}_k\ 0_k \equiv \mathsf{rUnit}_k\ 0_k$ (that is, there is at most one *h-structure* [10, 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 [25, Prop. 5.1.4] in [27].

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 [10] in order to prove that $G \simeq \pi_1(\mathsf{K}(G,1))$.

Theorem 1. $K_n \simeq \Omega\,K_{n+1}$

Proof. As observed above, the case $n = 0$ is just $\mathbb{Z} \simeq \Omega\,\mathbb{S}^1$, so we focus on the case when $n \geq 1$. We first define

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

This is just the usual map from the Freudenthal equivalence defined in [1, Section 8.6]. It follows easily from Lemma 1 that σ_n is a morphism in the sense that $\sigma_n\ (x +_k y) \equiv \sigma_n\ x \cdot \sigma_n\ y$. We proceed by the encode-decode method and define a fibration $\mathsf{Code} : \mathsf{K}_{n+1} \to n\text{-}\mathsf{Type}$. Since n-Type is $(n+1)$-truncated [1, 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

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

The inverse is defined by

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

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 transport laws for functions and ua we can deduce that this function applied to $|\,x\,|$ yields the following (up to a path):

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

As σ_n is a morphism and $+_k$ 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 transport law for ua, the other direction amounts to showing $(|\,y\,| +_k 0_k)\ \text{-}_k\ 0_k \equiv |\,y\,|$, which clearly holds. □

In addition to Theorem 1, the direct definition of $+_k$ gives a short proof that $\Omega\,\mathsf{K}_n$ is commutative.

Lemma 2. *For $n > 0$ 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 e.g. $\mathsf{cong}_2\ +_k\ p\ q$ is of type $0_k +_k 0_k \equiv 0_k +_k 0_k$. Using this definitional equality, we may apply the rUnit_k and lUnit_k laws pointwise to p and q which gives us:

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

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 3. *For $n > 0$ 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 2, but using using commutativity of $+_k$. □

Theorem 2. *$\Omega\,\mathsf{K}_n$ is commutative with respect to path composition.*

Proof. As $\mathbb{Z}$ is a set, the statement is trivial for $n = 0$. When $n \geq 1$ the result directly follows from Lemmas 2 and 3. □

An alternative proof of Theorem 2 can be found in [25, 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 2 is more computationally efficient.

B. Group structure on $H^n(A)$

We now return to $H^n(A)$ and define the group operations:

$$0_h = |\,\lambda\, x \to 0_k\,|$$
$$|\,f\,| +_h |\,g\,| = |\,\lambda\, x \to f\ x +_k g\ x\,|$$
$$\text{-}_h\,|\,f\,| = |\,\lambda\, x \to \text{-}_k\ f\ x\,|$$

The fact that $(H^n(A), 0_h, +_h, \text{-}_h)$ forms an abelian group follows immediately from the group laws for K_n and funExt.

Analogously to the above cohomology groups, we can also define a *reduced* version which we denote by $\widetilde{H}^n(A)$. This is often preferred in classical algebraic topology as it avoids some exceptional cases which simplify statements [9]. 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 $n \geq 1$.

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

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 [25, Prop. 5.1.4] it was difficult to get `Cubical Agda` to typecheck in reasonable time without using the abstract keyword (which erases computational content).

IV. The Eilenberg-Steenrod axioms

A common approach in classical mathematics is to work abstractly with cohomology using the Eilenberg-Steenrod axioms [29]. The goal of this section is to verify that our definition of cohomology satisfies a variation of these axioms, which ensures that it is a well-behaved cohomology theory.

A. The axioms in HoTT/UF

The Eilenberg-Steenrod axioms have been studied previously in HoTT/UF by [12], [13] and [8]. In order to state the **Exactness** axiom, we need to introduce (homotopy) cofibers (also known as *mapping cones*).

Definition 7 (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 axioms.

Definition 8 (Eilenberg-Steenrod axioms)**.** *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 $f : A \to_* B$, the below sequence is exact.*

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

Here f^* and cfcod^* are short for $E^n\, f$ and $E^n\, \mathsf{cfcod}$.

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. The sequence in the **Exactness** axiom is *exact* if the kernel of f^* (the elements of $E^n\, B$ that gets mapped to 0 in $E^n\, A$) is equal to the image of cfcod^*. As $E^n\, A$ is a set, the property of "b being 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 family of types A_i indexed by I the following groups are isomorphic:

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

Proving this typically requires that the index set I satisfies the set theoretic axiom of choice [13]. 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. 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 $\widetilde{H}^n(A)$ for $n \geq 0$, 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 2. *$\widehat{H}^n$ is an ordinary reduced cohomology theory.*

Proof (sketch). We verify the axioms, omitting trivial cases and leaving technical 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

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

where $p = \mathsf{rCancel}_k\ (f\ 0_k)$. By pointwise application of Theorem 1, this gives us a map $\varphi : H^{n+1}(\mathsf{Susp}\ A) \to H^n(A)$, sending $|\, f\, |$ to $|\, \lambda\, x \to \sigma_n^{-1}\, (f'\ x)\, |$. 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 $f\ \mathsf{north}$ 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 a path $p' : f^*\ |\, g\, | \equiv 0_h$. 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:

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

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 3. *$\widehat{H}^n$ 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 $\phi : \widetilde{H}^0(A \vee B) \to \widetilde{H}^0(A) \times \widetilde{H}^0(B)$. That ϕ is an isomorphism follows easily as $\mathbb{Z}$ 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

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

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

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

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

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

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

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

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 $\mathsf{P}\ 0_k\ \mathsf{refl}\ \mathsf{refl} \equiv \mathsf{Q}\ 0_k\ \mathsf{refl} \equiv \mathsf{refl}$; in this case we only need to fill a square with refl on all sides, which is done by $\mathsf{refl}\ \{x = \mathsf{refl}\}$. 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 III, 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 (1). □

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.

C. Characterizing $\mathbb{Z}$ cohomology groups using the axioms

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

Theorem 3 (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*

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

There are many variants of Theorem 3. Cavallo constructed the sequence for general reduced cohomology directly from the Eilenberg-Steenrod axioms in [12], whereas Brunerie constructed a version with alternating reduced and unreduced groups for a cohomology theory similar to ours in [25, 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 3. It is easy to prove that $\widetilde{H}^0(\mathbb{S}^0) \cong \mathbb{Z}$ and get a stable sequence

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

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 quite 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 in the `agda/cubical` library, but have so far been able to avoid it and instead give direct characterizations of all cohomology groups that we considered.

V. Characterizing cohomology groups directly

In this section, we characterize the unreduced cohomology groups $H^n(A)$ for the spheres, torus, Klein bottle and real projective plane. These will be used for computations in Section VI and the goal is to characterize the groups directly in order to have isomorphisms that compute as fast as possible.

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 $H^n(A)$ is trivial for $n > 0$ are also easy to characterize for

all A that we consider using connectedness arguments. We give one such characterization in Proposition 5 and the rest are omitted for space reasons. The main focus in this section will hence be on the non-trivial $H^n(A)$ with $n > 0$, but we emphasize that all cases, as well as homomorphism proofs, have been formalized and the curious reader is encouraged to consult the formalization.

A. Spheres

The key to characterizing the cohomology groups of spheres is the **Suspension** axiom. Recall that $\mathbb{S}^{n+1} = \mathsf{Susp}\,\mathbb{S}^n$ for $n \geq 1$ and thus we have that $H^{n+1}(\mathbb{S}^{m+1}) \simeq H^n(\mathbb{S}^m)$. This covers the inductive step in the proof of the following proposition.

Proposition 4. *$H^n(\mathbb{S}^n) \simeq \mathbb{Z}$ for $n \geq 1$.*

Proof. By **Suspension** and induction, it suffices to consider the case when $n = 1$. 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 that $(x \equiv x) \simeq (\mathsf{0}_k \equiv \mathsf{0}_k)$ for any $x : \mathsf{K}_1$. Hence

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

□

The trivial cohomology groups are easily handled in a similar manner.

Proposition 5. *$H^n(\mathbb{S}^m) \simeq \mathbb{1}$ for $n, m \geq 1$ with $n \neq m$.*

Proof. By **Suspension** 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 $|f| \equiv \mathsf{0}_h$. This is a proposition, so by 0-connectedness of K_1, we may assume that $f\ \mathsf{base} \equiv |\mathsf{0}_k|$. But by the definition of $(m-1)$-truncations, any map $f : \mathbb{S}^m \to \| \mathsf{K}_1 \|_{m-1}$ is constant. Hence $|f| \equiv \mathsf{0}_h$ 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 4 that

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

This type is hence clearly contractible since K_n is 0-connected for all $n > 0$. □

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.

B. The torus

The cohomology groups of the torus are also easy to characterize directly. 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 square constructor $\square$ corresponds to the usual gluing diagram for constructing the torus in classical topology as it identifies the line ℓ_1 with itself over an identification of ℓ_2 with itself. As discussed in the introduction, it is easy to prove in `Cubical Agda` that $\mathbb{T}^2 \simeq \mathbb{S}^1 \times \mathbb{S}^1$ (see [15, Section 2.4.1]). This allows us to apply currying to $\mathbb{T}^2 \to \mathsf{K}_n$, which is the key step in the proofs of the following two propositions.

Proposition 6. *$H^1(\mathbb{T}^2) \simeq \mathbb{Z} \times \mathbb{Z}$*

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 the proof of Proposition 4, 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, we have that

$$\begin{aligned} H^1(\mathbb{T}^2) &\simeq \| \mathbb{S}^1 \to \mathsf{K}_1 \times \mathbb{Z} \|_0 \\ &\simeq \| (\mathbb{S}^1 \to \mathsf{K}_1) \times (\mathbb{S}^1 \to \mathbb{Z}) \|_0 \\ &\simeq \| \mathbb{S}^1 \to \mathsf{K}_1 \|_0 \times \| \mathbb{S}^1 \to \mathbb{Z} \|_0 \\ &\overset{\text{def}}{\equiv} H^1(\mathbb{S}^1) \times H^0(\mathbb{S}^1) \\ &\simeq \mathbb{Z} \times \mathbb{Z} \end{aligned}$$

□

Proposition 7. *$H^2(\mathbb{T}^2) \simeq \mathbb{Z}$*

Proof. Again, we consider the underlying function space, post currying, $\mathbb{S}^1 \to (\mathbb{S}^1 \to \mathsf{K}_2)$. Like above, this is just

$$\begin{aligned} (\mathbb{S}^1 \to \mathsf{K}_2 \times \Omega\,\mathsf{K}_2) &\simeq (\mathbb{S}^1 \to \mathsf{K}_2 \times \mathsf{K}_1) \\ &\simeq (\mathbb{S}^1 \to \mathsf{K}_2) \times (\mathbb{S}^1 \to \mathsf{K}_1) \end{aligned}$$

and hence we have

$$\begin{aligned} H^2(\mathbb{T}^2) &\simeq \| (\mathbb{S}^1 \to \mathsf{K}_2) \times (\mathbb{S}^1 \to \mathsf{K}_1) \|_0 \\ &\simeq \| \mathbb{S}^1 \to \mathsf{K}_2 \|_0 \times \| \mathbb{S}^1 \to \mathsf{K}_1 \|_0 \\ &\overset{\text{def}}{\equiv} H^2(\mathbb{S}^1) \times H^1(\mathbb{S}^1) \\ &\simeq \mathbb{Z} \end{aligned}$$

□

C. The Klein Bottle and the Real Projective Plane

The Klein bottle, $\mathbb{K}^2$, is also defined as a HIT, but with a twist in $\square$ just like in the classical gluing diagram:

```
data 𝕂² : Type where
  pt : 𝕂²
  ℓ₁ ℓ₂ : pt ≡ pt
  □ : PathP (λ i → ℓ₂ (~ i) ≡ ℓ₂ i) ℓ₁ ℓ₁
```

Note that $\square$ equivalently may be interpreted as the more convenient 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

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

Noting that, by Theorem 2, path composition in K_n is commutative, we get

$$(p \cdot q \cdot p \equiv q) \simeq (p \cdot p \cdot q \equiv q) \simeq (p \cdot p \equiv \mathsf{refl})$$

Hence, we get

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

This the key to characterizing the cohomology groups of $\mathbb{K}^2$.

Proposition 8. $H^1(\mathbb{K}^2) \simeq \mathbb{Z}$

Proof. Note that for $x : \mathsf{K}_1$, we have that

$$\sum_{p:x \equiv x} (p \cdot p \equiv \mathsf{refl}) \simeq \sum_{a:\mathbb{Z}} (a + a \equiv 0)$$

and hence this type is clearly contractible. This allows us to simplify (2) and get

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

□

Proposition 9. $H^2(\mathbb{K}^2) \cong \mathbb{Z}/2\mathbb{Z}$

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 (2), 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:

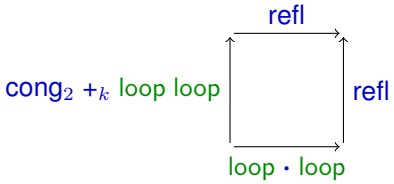
$$\| \sum_{x:\mathsf{K}_1} (x +_k x \equiv 0_k) \|_0$$

With some abuse of notation, let us simply write loop for the canonical loop in K_1, i.e. cong $|\_|$ loop. We are done if we can prove that, for any element of $\| \sum_{x:\mathsf{K}_1} (x +_k x \equiv 0_k) \|_0$, it is either is path-equal to $|\,(0_k\,,\mathsf{refl})\,|$ or to $|\,(0_k\,,\mathsf{loop})\,|$. Note that this is well-typed as $0_k +_k 0_k$ is definitionally 0_k.

We begin by characterizing terms on the form $|\,(0_k\,,p)\,|$ by noting that $p \equiv \mathsf{loop}^k$ for some $k : \mathbb{Z}$, due to the equivalence $\Omega\,\mathsf{K}_1 \simeq \mathbb{Z}$. The claim is that $|\,(0_k\,,\mathsf{loop}^k)\,| \equiv |\,(0_k\,,\mathsf{refl})\,|$ if k is even and $|\,(0_k\,,\mathsf{loop}^k)\,| \equiv |\,(0_k\,,\mathsf{loop})\,|$ if k is odd. We do this by inducting on k (assuming $k \geq 0$ as the case $k < 0$ is completely symmetric). When k is 0 or 1, the statement is trivial, since $\mathsf{loop}^0 \equiv \mathsf{refl}$ by definition. The crucial case is when $k = 2$ in which 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 now need to fill the square:



However, this is trivial since $\mathsf{cong}_2\ +_k\ \mathsf{loop}\ \mathsf{loop}$ definitionally reduces to loop · loop.

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

$$|\,(0_k\,,p)\,| \diamond |\,(0_k\,,q)\,| \equiv |\,(0_k\,,p \cdot q)\,| \quad (3)$$

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 1 in order to define $\diamond$. We define

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

To complete the definition of $\diamond$, Lemma 1 requires us to prove that $|\,(0_k\,,p \cdot q)\,| \equiv |\,(0_k\,,q \cdot p)\,|$. This follows immediately by commutativity of $\Omega\,\mathsf{K}_1$, and thereby $\diamond$ is defined. The fact that it satisfies (3) follows from the left path in Lemma 1.

The inductive step is now easy to complete. We have

$$\begin{aligned} |\,(0_k\,,\mathsf{loop}^{k+2})\,| &\equiv |\,(0_k\,,\mathsf{loop}^k)\,| \diamond |\,(0_k\,,\mathsf{loop}^2)\,| \\ &\equiv |\,(0_k\,,\mathsf{loop}^k)\,| \diamond |\,(0_k\,,\mathsf{refl})\,| \\ &\equiv |\,(0_k\,,\mathsf{loop}^k)\,| \end{aligned}$$

Hence, we have shown that there are precisely two elements (up to path-equality) in $\| \sum_{x:\mathsf{K}_1} (x +_k x \equiv 0_k) \|_0$, and thus

$$H^2(\mathbb{K}^2) \simeq \| \sum_{x:\mathsf{K}_1} (x +_k x \equiv 0_k) \|_0 \simeq \mathbb{Z}/2\mathbb{Z}$$ □

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 define $\mathbb{R}P^2$ as a HIT as follows.

```
data ℝP² : Type where
  pt : ℝP²
  ℓ : pt ≡ pt
  □ : ℓ ≡ ℓ ⁻¹
```

We characterize $H^n(\mathbb{R}P^2)$ for $n \geq 1$ by

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

When instantiated with $n = 1$ and $n = 2$ this is precisely one of the types appearing in the proofs of Proposition 8 and

Proposition 9 respectively. We have hence already proved that $H^1(\mathbb{R}P^2) \simeq \mathbb{1}$ and $H^2(\mathbb{R}P^2) \simeq \mathbb{Z}/2\mathbb{Z}$.

VI. Computing with the cohomology groups

For every equivalence $\phi : H^n(A) \simeq G$ in Section V, 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 15ms to terminate, which we record in a comment.

```
test : φ (φ⁻¹ 1 +ₕ φ⁻¹ 2) ≡ 3        -- 15ms
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	$\mathbb{Z}$	✓	✗[1]
$\mathbb{T}^2$	H^1	$\mathbb{Z} \times \mathbb{Z}$	✓	✓
	H^2	$\mathbb{Z}$	✓	✗[1]
$\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1$	H^1	$\mathbb{Z} \times \mathbb{Z}$	✓	✓
	H^2	$\mathbb{Z}$	✓	✗[1]
$\mathbb{K}^2$	H^1	$\mathbb{Z}$	✓	✓
	H^2	$\mathbb{Z}/2\mathbb{Z}$	✗	✗
$\mathbb{R}P^2$	H^2	$\mathbb{Z}/2\mathbb{Z}$	✗	✗

[1] Some simple cases terminate (e.g. $\phi\,(\phi^{-1}\ 0\ +_h\ \phi^{-1}\ 0) \equiv 0$), but take between 0.2s and 4s.

For most spaces considered here **Test 1** terminates in less than 0.2s. This is a considerable improvement to prior attempts in [27] 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 [25] and the Mayer-Vietoris sequence). However, **Test 1** fails to terminate for $H^2(\mathbb{K}^2)$ and $H^2(\mathbb{R}P^2)$. 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 morphism. For $\mathbb{K}^2$, this construction also relies on the proof of Theorem 2. Higher cohomology groups of spheres also appear to suffer from the same problems. For $\phi : H^3(\mathbb{S}^3) \simeq \mathbb{Z}$, **Test 1** fails even for 0. This is more surprising as it is a very simple example of a non-trivial cohomology group with $n > 2$. We conjecture that one of the major culprits is `Cubical Agda`'s handling of truncations and are exploring ways to optimize it.

VII. Conclusions

We have developed many classical results in cohomology theory synthetically using `Cubical Agda`. This has led to new and more direct proofs than what already exists in the HoTT/UF literature. Furthermore, the synthetic characterizations of the cohomology groups for $\mathbb{K}^2$ and $\mathbb{R}P^2$ are novel. The proofs have been constructed with computational efficiency in mind, allowing us to make explicit computations involving several non-trivial cohomology groups. These tests can easily be modified and used to give benchmarks for future improvements of `Cubical Agda` and related systems like `redtt` [30].

A. 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 [31] formalized $K(G,1)$ as a HIT, following [10], and proved that it satisfies $\pi_1(K(G,1)) \equiv G$. Alfieri [32] and Harington [33] 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 [34] has been used to compute homology [35] and persistent homology [36] with coefficients in a field. This was later extended to homology with $\mathbb{Z}$ coefficients in [37]. 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 [38] using stable homotopy groups. 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.

Another natural continuation of this work would be to consider the cohomology ring $H^*(A)$ of a type A. The invariants characterized in this paper cannot distinguish $\mathbb{T}^2$ and $\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1$, but this should be possible by means of a single computation using the cup product $\smile$ in their respective cohomology rings. A more ambitious application would be to reduce all of [25, Chapter 6] to a single computation using $\smile$ for $\mathbb{C}P^2$. The definition of $H^*(A)$ in HoTT/UF was given by Brunerie [25, Chapter 5.1]. However, properties of $\smile$ relies on the smash product HIT which has proved to be surprisingly complex to reason about formally [39]. Luckily, Baumann [40] has managed to define and reason about $\smile$ in `HoTT-Agda` using ideas of van Doorn [8]. It should be possible to port these ideas to `Cubical Agda` and hopefully significantly simplify the involved proofs.

References

[1] The Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*. Self-published, 2013. [Online]. Available: https://homotopytypetheory.org/book/

[2] P. Martin-Löf, *Intuitionistic type theory*, ser. Studies in Proof Theory. Bibliopolis, 1984, vol. 1.

[3] V. Voevodsky, "The equivalence axiom and univalent models of type theory," February 2010, notes from a talk at Carnegie Mellon University. [Online]. Available: http://www.math.ias.edu/vladimir/files/CMU_talk.pdf

[4] K. Kapulkin and P. L. Lumsdaine. (2016, June) The Simplicial Model of Univalent Foundations (after Voevodsky). Preprint. [Online]. Available: https://arxiv.org/abs/1211.2851

[5] S. Awodey and M. A. Warren, "Homotopy theoretic models of identity types," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 146, no. 1, pp. 45–55, January 2009.

[6] K.-B. Hou (Favonia), E. Finster, D. R. Licata, and P. L. 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*, ser. LICS '16. New York, NY, USA: ACM, 2016, pp. 565–574.

[7] K.-B. Hou (Favonia) and M. Shulman, "The Seifert-van Kampen Theorem in Homotopy Type Theory," in *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), J.-M. Talbot and L. Regnier, Eds., vol. 62. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, pp. 22:1–22:16.

[8] F. van Doorn, "On the Formalization of Higher Inductive Types and Synthetic Homotopy Theory," Ph.D. dissertation, University of Nottingham, May 2018. [Online]. Available: https://arxiv.org/abs/1808.10690

[9] A. Hatcher, *Algebraic Topology*. Cambridge University Press, 2002. [Online]. Available: https://pi.math.cornell.edu/~hatcher/AT/AT.pdf

[10] D. R. Licata and E. 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)*, ser. CSL-LICS '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2603088.2603153

[11] M. Shulman, "Cohomology," 2013, post on the Homotopy Type Theory blog: http://homotopytypetheory.org/2013/07/24/.

[12] E. Cavallo, "Synthetic Cohomology in Homotopy Type Theory," Master's thesis, Carnegie Mellon University, 2015.

[13] U. Buchholtz and K.-B. Hou Favonia, "Cellular Cohomology in Homotopy Type Theory," in *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, ser. LICS '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 521–529. [Online]. Available: https://doi.org/10.1145/3209108.3209188

[14] The `Agda` Development Team, "The `Agda` Programming Language," 2021. [Online]. Available: http://wiki.portal.chalmers.se/agda/pmwiki.php

[15] A. Vezzosi, A. Mörtberg, and A. Abel, "Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types," *Proceedings of the ACM on Programming Languages*, vol. 3, no. ICFP, pp. 87:1–87:29, August 2019.

[16] C. Cohen, T. Coquand, S. Huber, and A. Mörtberg, "Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom," in *21st International Conference on Types for Proofs and Programs (TYPES 2015)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), T. Uustalu, Ed., vol. 69. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 5:1–5:34.

[17] P. Martin-Löf, "An Intuitionistic Theory of Types: Predicative Part," in *Logic Colloquium '73*, ser. Studies in Logic and the Foundations of Mathematics, H. E. Rose and J. C. Shepherdson, Eds. North-Holland, 1975, vol. 80, pp. 73–118.

[18] V. Voevodsky, "An experimental library of formalized mathematics based on the univalent foundations," *Mathematical Structures in Computer Science*, vol. 25, no. 5, pp. 1278–1294, 2015.

[19] T. Coquand, S. Huber, and A. Mörtberg, "On Higher Inductive Types in Cubical Type Theory," in *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, ser. LICS 2018. New York, NY, USA: ACM, 2018, pp. 255–264.

[20] E. Cavallo and R. Harper, "Higher Inductive Types in Cubical Computational Type Theory," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1:1–1:27, January 2019.

[21] A. Mörtberg and L. Pujet, "Cubical Synthetic Homotopy Theory," in *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, ser. CPP 2020. New York, NY, USA: Association for Computing Machinery, 2020, pp. 158–171. [Online]. Available: https://doi.org/10.1145/3372885.3373825

[22] G. Brunerie, K.-B. Hou (Favonia), E. Cavallo, T. Baumann, E. Finster, J. Cockx, C. Sattler, C. Jeris, M. Shulman *et al.*, "Homotopy Type Theory in `Agda`," 2018. [Online]. Available: https://github.com/HoTT/HoTT-Agda

[23] K. Sojakova, "The Equivalence of the Torus and the Product of Two Circles in Homotopy Type Theory," *ACM Transactions on Computational Logic*, vol. 17, no. 4, pp. 29:1–29:19, November 2016.

[24] D. R. Licata and G. Brunerie, "A Cubical Approach to Synthetic Homotopy Theory," in *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, ser. LICS '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 92–103.

[25] G. Brunerie, "On the homotopy groups of spheres in homotopy type theory," Ph.D. dissertation, Université Nice Sophia Antipolis, 2016. [Online]. Available: http://arxiv.org/abs/1606.05916

[26] C. Angiuli, E. Cavallo, A. Mörtberg, and M. Zeuner, "Internalizing representation independence with univalence," *Proc. ACM Program. Lang.*, vol. 5, no. POPL, Jan. 2021. [Online]. Available: https://doi.org/10.1145/3434293

[27] A. Ljungström, "Computing Cohomology in `Cubical Agda`," Master's thesis, Stockholm University, 2020.

[28] D. R. Licata and M. 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*, ser. LICS '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 223–232.

[29] S. Eilenberg and N. Steenrod, *Foundations of Algebraic Topology*, ser. Foundations of Algebraic Topology. Princeton University Press, 1952.

[30] The RedPRL Development Team, "The `redtt` proof assistant," 2018. [Online]. Available: https://github.com/RedPRL/redtt/

[31] Z. Qian, "Towards Eilenberg-MacLane Spaces in Cubical Type Theory," Master's thesis, Carnegie Mellon University, 2019.

[32] V. Alfieri, "Formalisation de notions de théorie des groupes en théorie cubique des types," 2019, internship report, supervised by Thierry Coquand.

[33] E. Harington, "Groupes de cohomologie en théorie des types univalente," 2020, internship report, supervised by Thierry Coquand.

[34] The `Coq` Development Team, "The `Coq` Proof Assistant," 2021. [Online]. Available: https://www.coq.inria.fr

[35] J. Heras, M. Dénès, G. Mata, A. Mörtberg, M. Poza, and V. Siles, "Towards a Certified Computation of Homology Groups for Digital Images," in *Proceedings of the 4th International Conference on Computational Topology in Image Context*, ser. CTIC'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 49–57. [Online]. Available: https://doi.org/10.1007/978-3-642-30238-1_6

[36] J. Heras, T. Coquand, A. Mörtberg, and V. Siles, "Computing Persistent Homology Within `Coq`/SSReflect," *ACM Transactions on Computational Logic*, vol. 14, no. 4, pp. 1–26, 2013. [Online]. Available: http://doi.acm.org/10.1145/2528929

[37] G. Cano, C. Cohen, M. Dénès, A. Mörtberg, and V. Siles, "Formalized Linear Algebra over Elementary Divisor Rings in `Coq`," *Logical Methods in Computer Science*, vol. 12, no. 2, 2016. [Online]. Available: http://dx.doi.org/10.2168/LMCS-12(2:7)2016

[38] R. Graham, "Synthetic Homology in Homotopy Type Theory," 2018, preprint. [Online]. Available: https://arxiv.org/abs/1706.01540

[39] G. Brunerie, "Computer-generated proofs for the monoidal structure of the smash product," Nov. 2018, *Homotopy Type Theory Electronic Seminar Talks*. [Online]. Available: https://www.uwo.ca/math/faculty/kapulkin/seminars/hottest.html

[40] T. Baumann, "The cup product on cohomology groups in homotopy type theory," Master's thesis, University of Augsburg, 2018.