Cohomology Computations in Cubical Agda

Axel Ljungström

Introduction

- A fair bit of cohomology theory been done in Homotopy Type Theory (HoTT):
  - Licata & Finster (2014)
  - Cavallo (2015)
  - Brunerie (2016)
  - Buchholtz & Favonia (2018)
  - van Doorn (2018)
- We have been working on integer cohomology in Cubical Agda
- The goal is not only to be able to prove the results from the previously mentioned papers, but also to be able to use our theory for computations. This requires some tweaking of previous work.

Definitions

Definition 1 (Suspensions)

Let A be a type. We define the suspension of A, denoted ΣA, as a HIT with the following constructors

- north : ΣA
- south : ΣA
- merid : $A \to \text{north} \equiv \text{south}$

We always take ΣA to be pointed by north.

Definition 2 (Spheres)

We define $\mathbb{S}^1$ with the usual base/loop constructors. For $n \geq 1$, we define $\mathbb{S}^{n+1} = \Sigma \mathbb{S}^n$

Definition 3 (Eilenberg-MacLane Spaces)

For $n \geq 0$, we define K_n, the n-th Eilenberg-MacLane space, by

- $K_0 = \mathbb{Z}$
- $K_n = \|\mathbb{S}^n\|_n$, $n \geq 1$.

We take K_n to be pointed by $0_k : K_n$, defined by

- $0_k = 0$, when $n = 0$
- $0_k = |*_{\mathbb{S}^n}|$, otherwise.

Definition 4

For any type A, we define $H^n(A) = \|A \to K_n\|_0$. This is pointed by $0_h = \lambda x \,.\, 0_k$.

Definitions

Definition 5 (Connectedness)

A type A is said to be n-connected if $\|A\|_n$ is contractible. If a function $f : A \to B$ has n-connected fibres, we say that f is an n-connected function.

Notation: I'll be using $\equiv$ for path equality and cong_f for $\mathrm{ap}_f : x \equiv y \to f(x) \equiv f(y)$, following the notation in the Cubical library. When f is a binary function, we get a binary cong: $\mathrm{cong}_f^2 : (x \equiv y) \times (z \equiv w) \to f(x, z) \equiv f(y, w)$.

Group structure

- We get the group structure on $H^n(A)$ from defining one on K_n.
- In e.g. Brunerie (2016), addition in K_n is given by an isomorphism $K_n \simeq \Omega K_{n+1}$, so that K_n inherits path composition.
- This forces us to use the Freudenthal Suspension Theorem, which does not compute well.
- We still want $K_n \simeq \Omega K_{n+1}$ for most cohomology group characterisations
- Our approach: define the group structure first and get $K_n \simeq \Omega K_{n+1}$ a corollary.
- We want our cohomology theory to satisfy two things:
  - It should not rely on any theory about connected types/functions
  - It should have useful definitional equalities, e.g. $0_k + 0_k \equiv 0_k$.

No Connectedness

- Most theory about connectedness makes (repeated) use of the following lemma.

Theorem 6

For any $n \geq -2$ and $x, y : X$, we have

$$\|x \equiv y\|_n \simeq \left(|x| \equiv_{\|X\|_{n+1}} |y|\right)$$

- The inverse function of this equivalence is given by a combination of truncation elimination and a transport over a path constructed by univalence.
- Wherever this theorem pops up, we seem to run into problems with computations!
- This prevents us from using virtually any theory about connected maps and types.

Group Structure

- The addition on K_n, which we denote by

$$+_k : K_n \times K_n \to K_n$$

  is given by the "Wedge Connectivity Lemma" from the HoTT book (this is similar to what is done in e.g. Licata & Finster (2014)).
- The lemma, in its original form, essentially tells us that maps from wedge sums induce (well-behaved) maps from products under some connectedness assumptions.
- Problem: uses connectedness. We rectify this by only proving the special case of the theorem when the wedge sum

Wedge Connectivity Lemma

Lemma 7

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

$$f_l : (x : \mathbb{S}^n) \to P(x, *_{\mathbb{S}^m})$$
$$f_r : (y : \mathbb{S}^m) \to P(*_{\mathbb{S}^n}, y)$$

together with a path $p : f_l(*_{\mathbb{S}^n}) \equiv f_r(*_{\mathbb{S}^m})$. *Then we there is a total function* $F : (x : \mathbb{S}^n \times \mathbb{S}^m) \to P(x)$ *equipped with homotopies*

$$\mathsf{left} : (x : \mathbb{S}^n) \to f_l(x) \equiv F(x, *_{\mathbb{S}^m})$$
$$\mathsf{right} : (x : \mathbb{S}^m) \to f_r(x) \equiv F(*_{\mathbb{S}^n}, x)$$

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

Wedge Connectivity Lemma

- When stated for spheres, we can also make either the left- or the right homotopy hold by refl.
- Proof idea: Construct F, left and right mutually while inducting on n and m.
- F is constructed by pattern matching on $\mathbb{S}^n$ and $\mathbb{S}^m$. This way, we can force either one of the homotopies to hold definitionally.
- For higher path constructors, we can use the inductive hypothesis to give the required fillers.
- **Remark:** This direct proof should be easy to generalise for Eilenberg-MacLane spaces over arbitrary groups, as presented in Licata & Finster (2014).

Addition

- We may now use this to define our addition $+_k : K_n \times K_n \to K_n$
- For $n = 0$, we choose regular integer addition.

Addition

- We may now use this to define our addition $+_k : K_n \times K_n \to K_n$
- For $n = 0$, we choose regular integer addition.
- For $n \geq 1$, it suffices to give a map $\mathbb{S}^n \times \mathbb{S}^n \to \underbrace{K_n}_{\|\mathbb{S}^n\|_n}$

Addition

- We may now use this to define our addition $+_k : K_n \times K_n \to K_n$
- For $n = 0$, we choose regular integer addition.
- For $n \geq 1$, it suffices to give a map $\mathbb{S}^n \times \mathbb{S}^n \to \underbrace{K_n}_{\|\mathbb{S}^n\|_n}$
- For $n = 1$, the addition is defined by

$$(\mathsf{base}, x) \mapsto |\, x \,|$$
$$(\mathsf{loop}\, i, \mathsf{base}) \mapsto |\, \mathsf{loop}\, i \,|$$

The final square for $(\mathsf{loop}\, i, \mathsf{loop}\, j)$ is essentially $\mathsf{cong}_{\lambda x.\, |x|}(\mathsf{loop} \cdot \mathsf{loop})$.

Addition

- For $n \geq 2$, wedge connectivity will apply. So we only need two maps

$$f_l, f_r : \mathbb{S}^n \to K_n$$

and a proof $p : f_l(\text{north}) \equiv f_r(\text{north})$. We choose the inclusion $\lambda\, x\,.\ |x|$ for both maps and let $p = \text{refl}$.

Monoid laws

- The monoid laws and commutativity for $+_k$ are now very easy to prove using wedge connectivity.
- All of them will reduce to refl or something very similar at 0_k (this is good – they often occur instantiated at 0_k in computations).
- In particular, $\mathsf{lUnit}_k(0_k) \equiv \mathsf{rUnit}_k(0_k)$ holds by refl.
- Hence, $+_k$ is an h-structure on K_n. These are unique (this also follows by wedge connectivity), so we can be convinced that our definition of $+_k$ is correct.

Inversion

- Inversion is easy to define: for any $x : K_n$, the map $f_x = \lambda y \,.\, x +_k y$ is an equivalence. Since we are proving a proposition, we only need to do it for $x = 0_k$, in which case it reduces to the identity function.
- However, there is a more direct definition. I'll give it for $n \geq 2$ (it's similar for $n = 1$). $-_k \,|x|$ is defined by induction on x

$$
\begin{aligned}
-_k \,|\mathsf{north}| &= |\mathsf{north}| \\
-_k \,|\mathsf{south}| &= |\mathsf{north}| \\
-_k \,|\mathsf{merid}\, a\, i| &= |((\mathsf{merid}\, \mathsf{north}) \cdot (\mathsf{merid}\, a)^{-1})\, i|
\end{aligned}
$$

Properties

- The following lemma gives us short proofs of the cancellation laws for $-_k$ and the commutativity of path composition in ΩK_n.

Lemma 8

For any $p, q : \Omega K_n$, we have $cong^2_{+_k}(p, q) \equiv p \cdot q$.

- Note that this would not be well-typed if $0_k +_k 0_k$ did not reduce definitionally to 0_k.
- This will reduce the cancellation laws of $-_k$ to the cancellation laws of path composition...
- ... and the commutativity of path composition in ΩK_n to the commutativity of $+_k$.

$K_n \simeq \Omega K_{n+1}$

- Now that the set up is done, it is easy to prove that $K_n \simeq \Omega K_{n+1}$ by a very standard encode decode proof.
- We omit the proof, but here's the definition of the code fibration

$$\begin{aligned} \mathsf{Code} : K_{n+1} &\to n\text{-Type} \\ \mathsf{Code}(|\,\mathsf{north}\,|) &= K_n \\ \mathsf{Code}(|\,\mathsf{south}\,|) &= K_n \\ \mathsf{Code}(|\,\mathsf{merid}\,a\,i\,|) &= \mathsf{ua}(\lambda\, x\,.\ |a| +_k x)\, i \end{aligned}$$

- The proof is very similar to that of $\Omega\mathbb{S}^1 \simeq \mathbb{Z}$ after this.

Cohomology groups

- $H^n(A) = \|A \to K_n\|_0$ inherits all its structure from K_n. We have given characterised all cohomology groups of...
  - $\mathbb{S}^n$
  - the torus
  - $\mathbb{R}P^2$
  - the Klein bottle.
  - Arbitrary wedge sums, i.e. $H^n(A \vee B) \cong H^n(A) \times H^n(B)$ for $n \geq 1$.
- By "characterise", I mean that we have constructed group ismorphisms $H^n(A) \cong G$ where G is some well-known group (e.g. $\mathbb{Z}$).

Cohomology

- All isomorphisms have been constructed by giving explicitly constructed functions in each direction and proving that they cancel out, instead of using more general constructions (e.g. the Mayer-Vietoris sequence or similar).
- For instance, the isomorphism $H^1(\mathbb{S}^1) \cong \mathbb{Z}$ is given by

$$H^1(\mathbb{S}^1) = \left\|\mathbb{S}^1 \to K_1\right\|_0 \simeq \left\|\sum_{x:K_1} x \equiv x\right\|_0 \simeq \Omega K_1 \simeq \mathbb{Z}$$

- The first step looks the same for all cohomology groups we have characterised. After that, it is usually relatively clear what the rest of the function should be.

Computations

- Given a characterisation $\varphi : H^n(A) \cong G$, we have run two tests in Cubical Agda:
- Test 1: check whether $\varphi(\varphi^{-1}(x))$ reduces to x (assuming G is a closed type)
- Test 2: check whether $\varphi(\varphi^{-1}(x) +_h \varphi^{-1}(y))$ reduces to $x +_G y$

Computations

Type A	Cohomology	Group G	Test 1	Test 2
$\mathbb{S}^1$	H^1	$\mathbb{Z}$	✓	✓
$\mathbb{S}^2$	H^2	$\mathbb{Z}$	✓	✗[1]
$\mathbb{S}^3$	H^3	$\mathbb{Z}$	✓	✗
$\mathbb{T}^2$	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	$\mathbb{Z}$	✓	✓
$\mathbb{K}^2$	H^1	$\mathbb{Z}$	✓	✓
	H^2	$\mathbb{Z}/2\mathbb{Z}$	✗	✗
$\mathbb{R}P^2$	H^2	$\mathbb{Z}/2\mathbb{Z}$	✗	✗

[1] Some very trivial tests like $\varphi(\varphi^{-1}(0) + \varphi^{-1}(1)) = 1$ work ok

Computations

- Some of these results seem contradictory: it is, for instance, very surprising that $H^2(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1)$ passes Test 2 but $H^2(\mathbb{S}^2)$ does not.
- The isomorphism $H^2(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1) \cong \mathbb{Z}$ is essentially the same (but more complicated) than $H^2(\mathbb{S}^2) \cong \mathbb{Z}$.
- The isomorphism is given (roughly) by the projection

$$H^2(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1) \to H^2(\mathbb{S}^2)$$

Computations

- Another funny thing happens with $H^2(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1)$. Consider the term $\varphi(\varphi^{-1}(1) +_h \varphi^{-1}(1))$. This reduces to 2.
- We can expand $\varphi^{-1}(1)$ – unsurprisingly, it is a pretty complicated term. However, it is path equal to a much simpler term, namely $|\,g\,|_0$, where $g : \mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1 \to K_2$ is given by

$$g \restriction_{\mathbb{S}^2} (x) = |\,x\,|$$
$$g \restriction_{\mathbb{S}^1 \vee \mathbb{S}^1} (x) = 0_k$$

- But, unlike the more complicated case with $\varphi^{-1}(1)$, Agda is unable to evaluate $\varphi(|\,g\,|_0 +_h |\,g\,|_0)$!

The solution

- By inspecting the construction of these functions, one notices that the main difference between the constructions is a useless 0_h that gets appended by the map from $H^2(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1)$
- We can define a new addition, $+'_h$ by

$$x +'_h y = (x +_h 0_h) +_h (y +_h 0_h)$$

The solution

- By inspecting the construction of these functions, one notices that the main difference between the constructions is a useless 0_h that gets appended by the map from $H^2(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1)$
- We can define a new addition, $+'_h$ by

$$x +'_h y = (x +_h 0_h) +_h (y +_h 0_h)$$

- With this definition of addition, the previous examples compute!

The solution

- By inspecting the construction of these functions, one notices that the main difference between the constructions is a useless 0_h that gets appended by the map from $H^2(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1)$
- We can define a new addition, $+'_h$ by

$$x +'_h y = (x +_h 0_h) +_h (y +_h 0_h)$$

- With this definition of addition, the previous examples compute!
- Why???

The solution

- By inspecting the construction of these functions, one notices that the main difference between the constructions is a useless 0_h that gets appended by the map from $H^2(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1)$
- We can define a new addition, $+'_h$ by

$$x +'_h y = (x +_h 0_h) +_h (y +_h 0_h)$$

- With this definition of addition, the previous examples compute!
- Why???
- Could a similar stupid trick be used for the Brunerie number?

Conclusions

- We have defined a cohomology theory which is computationally efficient enough to let us carry out several non-trivial computations.
- This makes us able to produce several examples of numbers similar to but simpler than the Brunerie number.
- Many of these numbers seem to push the limits of Agda without being completely infeasible.
- Hopefully, these numbers can help shed some light on what it is that makes these types of computations so complex.