

Eilenberg-MacLane Spaces in Homotopy Type Theory

Daniel R. Licata*

Wesleyan University
dlicata@wesleyan.edu

Eric Finster

Inria Paris Rocquencourt
ericfinster@gmail.com

Abstract

Homotopy type theory is an extension of Martin-Löf type theory with principles inspired by category theory and homotopy theory. With these extensions, type theory can be used to construct proofs of homotopy-theoretic theorems, in a way that is very amenable to computer-checked proofs in proof assistants such as Coq and Agda. In this paper, we give a computer-checked construction of *Eilenberg-MacLane spaces*. For an abelian group G , an Eilenberg-MacLane space $K(G, n)$ is a space (type) whose n^{th} homotopy group is G , and whose homotopy groups are trivial otherwise. These spaces are a basic tool in algebraic topology; for example, they can be used to build spaces with specified homotopy groups, and to define the notion of cohomology with coefficients in G . Their construction in type theory is an illustrative example, which ties together many of the constructions and methods that have been used in homotopy type theory so far.

Categories and Subject Descriptors F.3.3 [Logics and meanings of programs]: Studies of program constructs—Type structure; F.4.1 [Mathematical logic and formal languages]: Mathematical logic—Lambda calculus and related systems

General Terms Theory

Keywords type theory, dependent types, homotopy type theory

1. Introduction

Homotopy type theory is an extension of Martin-Löf type theory with principles inspired by category theory and homotopy theory [2, 7–9, 13, 15, 22–24], such as higher inductive types [16, 17, 19] and Voevodsky’s univalence axiom [10, 23]. This paper is part of a line of work on using homotopy type theory to do *synthetic homotopy theory*, where homotopy-theoretic concepts are modeled

*This research was sponsored in part by the National Science Foundation under grant number DMS-1128155 and by the Institute for Advanced Study’s Oswald Veblen fund. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CSL-LICS 2014, July 14–18, 2014, Vienna, Austria.
Copyright © 2014 ACM 978-1-4503-2886-9...\$15.00.
<http://dx.doi.org/10.1145/2603088.2603153>

using the homotopy structure of types, and type theory is used to investigate them—type theory is used as a logic of homotopy theory. In homotopy theory, one studies topological spaces by way of their points, paths (between points), homotopies (paths or continuous deformations between paths), homotopies between homotopies (paths between paths between paths), and so on. In type theory, a space corresponds to a type A . Points of a space correspond to elements $a, b : A$. Paths in a space are modeled by elements of the identity type (propositional equality), which we notate $p : a =_A b$. Homotopies between paths p and q correspond to elements of the iterated identity type $p =_{a=A} q$. The rules for the identity type allow one to define the operations on paths that are considered in homotopy theory. These include identity paths $\text{id}_a : a = a$ (reflexivity of equality), inverse paths $!p : b = a$ when $p : a = b$ (symmetry of equality), and composition of paths $q \circ p : a = c$ when $p : a = b$ and $q : b = c$ (transitivity of equality)¹, as well as homotopies relating these operations (for example, $\text{id} \circ p = p$), and homotopies relating these homotopies, etc.

One of the basic questions in algebraic topology is calculating the homotopy groups of a space. Given a space X with a distinguished point x_0 , the *fundamental group of X at the point x_0* (denoted $\pi_1(X, x_0)$ or just $\pi_1(X)$ when x_0 is clear from context) is the group of loops at x_0 up to homotopy, with composition as the group operation. This fundamental group is the first in a sequence of *homotopy groups*, which provide higher-dimensional information about a space: the homotopy groups $\pi_n(X, x_0)$ “count” the n -dimensional loops in X up to homotopy. *Calculating a homotopy group $\pi_n(X)$* is to construct a group isomorphism between $\pi_n(X)$ and an explicit description of a group, such as \mathbb{Z} or $\mathbb{Z} \bmod k$.

Homotopy groups can be expressed in type theory by iterating the identity type. To a first approximation, $\pi_2(X, x_0)$ is the group of homotopies between id_{x_0} and itself, $\pi_3(X, x_0)$ is the group of homotopies between $\text{id}_{\text{id}_{x_0}}$ and itself, and so on. However, each homotopy group $\pi_n(X)$ focuses attention on the paths at level exactly n , while the higher identity types preserve information at level n and above—for example, the type $x_0 = x_0$ is a full *space* of loops at x_0 , which may itself have non-trivial higher homotopy groups. Thus, to define the homotopy groups, it is necessary to “kill” the higher structure of a type. This can be accomplished by using the truncation modality [21, Chapter 7]—given a type A , the type $\|A\|_n$ is intuitively the “best approximation of A as an n -type,” where an n -type has trivial homotopy groups above dimension n . For example, taking $n = 0$, the type $\|A\|_0$ is the best approximation to A which behaves like a set, in the sense that any two paths $p, q : x =_A y$ are themselves related by a path. Truncations can be constructed using higher inductive types, a

¹To match the Agda library used for our formalizations, we write path composition in function-composition order; this is different than in the Homotopy Type Theory book [21].

general mechanism that can also be used to define basic spaces such as the spheres. Thus, using tools such as identity types and higher inductives, one can investigate homotopy groups in type theory.

Licata and Shulman [14] present a first example of synthetic homotopy theory, calculating the fundamental group of the circle (i.e. counting the number of different loops on the circle S^1 , considered up to homotopy, or continuous deformation—for example, going around the circle once clockwise and then again counter-clockwise can be deformed to the constant path). The fundamental group of the circle is isomorphic to \mathbb{Z} , the additive group on the integers: up to homotopy, there is the loop that stands still (0), goes around counterclockwise n times ($+n$), or goes around clockwise n times ($-n$). Licata and Brunerie [12] present a second example, calculating $\pi_n(S^n)$, the n^{th} homotopy group of the n -dimensional sphere S^n , which is also \mathbb{Z} . This is proved by induction on n , showing that there are as many n -dimensional loops on the n -sphere as there are $n+1$ -dimensional loops on the $n+1$ -sphere; e.g. there are as many paths on the circle as there are two-dimensional homotopies on the surface of the sphere. Many other examples of synthetic homotopy theory are described in the Homotopy Type Theory book [21]. Moreover, synthetic homotopy theory has proved very amenable to formalization in proof assistants, and many of these theorems have been mechanized in Agda [18] and Coq [6].

1.1 Eilenberg-MacLane spaces

In this paper, we describe a new example of synthetic homotopy theory, a construction of *Eilenberg-MacLane spaces* (*EM-spaces*). For an abelian group G , the EM-space $K(G, n)$ is a space whose n^{th} homotopy group is G , and whose other homotopy groups are the trivial (1-element) group. EM-spaces are a useful tool in algebraic topology because they can be used to build other spaces with specified homotopy groups. For example, given abelian groups G and H , a space with π_1 equal to G and π_2 equal to H can be built by taking the product $K(G, 1) \times K(H, 2)$, because

$$\begin{aligned} & \pi_1(K(G, 1) \times K(H, 2)) \\ = & \pi_1(K(G, 1)) \times \pi_1(K(H, 2)) \quad \text{property of } \times \\ = & G \times 1 \quad \text{definition of EM-space} \\ = & G \end{aligned}$$

and similarly for $\pi_2(K(G, 1) \times K(H, 2)) = H$. Not all homotopy-classes of spaces can be built out of simple products of EM-spaces, but in classical algebraic topology any space can be built as a Postnikov tower, which is a sequence of fibrations whose fibers are EM-spaces. From this point of view, EM-spaces can be seen as basic building blocks of spaces which are dual, in a sense, to the cells found in a cell complex decomposition [25].

EM-spaces also enable a definition of *cohomology*. While most of the results in homotopy type theory so far have concerned homotopy groups, it is well known in algebraic topology that the computation of these groups quickly becomes extremely difficult, even for the most simple spaces: for example, there is no known complete description of the homotopy groups of the 2-sphere. As an alternative, modern algebraic topology focuses much of its attention on *cohomological type invariants*. These invariants satisfy additional properties rendering them more computable, yet still manage to capture important invariants of spaces. The construction of EM-spaces given in this paper provides the first appearance of this class of invariants in type theory. Given a type A , we define the n^{th} *reduced integral cohomology group of A* , denoted $\tilde{H}^n(A)$, by

$$\tilde{H}^n(A) = \|\!| A \rightarrow K(\mathbb{Z}, n) \|\!|_0$$

It follows from the calculations in this paper that we have

$$\tilde{H}^n(S^k) = \begin{cases} 0 & k \neq n \\ \mathbb{Z} & k = n \end{cases}$$

which shows that, unlike the homotopy groups, the cohomology groups of all spheres can be calculated explicitly.

More generally, we can use EM-spaces $K(G, n)$ to define *cohomology with coefficients in G* , and this definition fits with a general pattern found in the literature on *stable homotopy theory*. First, notice that the collection of Eilenberg-MacLane spaces $\{K(G, n)\}_{n \geq 0}$ has an interesting property: one has an equivalence

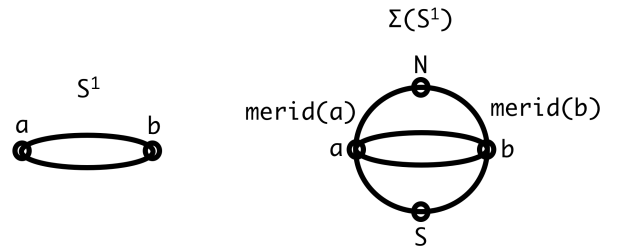
$$K(G, n) \simeq \Omega K(G, n+1)$$

where $\Omega(A)$, the loop space of A , is defined to be paths $a_0 =_A a_0$ for a specified base point $a_0 : A$. By iterating this observation, we find that for any k , there exists a space Y such that $K(G, n) \simeq \Omega^k Y$ (namely, we can take $Y = K(G, n+k)$). A space X with this property is known as an *infinite loop space*, and the hypothesized space Y is called a *k -fold delooping of X* . An infinite loop space together with a chosen collection of deloopings, one for each k , is known as a *spectrum* and it turns out that *every* such object gives rise to its own cohomological invariant of spaces [1]. Hence, the spaces defined in this paper could be used to construct a spectrum, namely that of ordinary cohomology with coefficients in G .

1.2 Overview of Results

In this paper, we give a definition of EM-spaces $K(G, n)$ and prove that they have the correct homotopy groups; we have formalized this proof in Agda. For intuition, consider the special case $K(\mathbb{Z}, n)$: we are looking to construct a space whose n^{th} homotopy group is \mathbb{Z} and whose homotopy groups are trivial otherwise. The n -sphere is a good starting point, because as mentioned above, $\pi_n(S^n) = \mathbb{Z}$. Moreover, it turns out that $\pi_k(S^n) = 1$ for $k < n$ (e.g., any 1-dimensional path on the sphere can be contracted to a point, because its inside is filled in). As hinted at above, however, it is not the case that $\pi_k(S^n) = 1$ for $k > n$ —indeed, the spheres have highly non-trivial higher homotopy groups. But by using the truncation modality, we can “kill” these higher homotopy groups, which leads us to define $K(\mathbb{Z}, n)$ by $\|\!| S^n \|\!|_n$.

To see how this generalizes, it is necessary to inspect the definition of S^n . Given a type A , the *suspension of A* , written ΣA , is a higher inductive type that generated by two points that are connected by one path for each point in A .² For example, the suspension of the circle (S^1) is the sphere (S^2):



Here a and b are points on the circle. N and S are the two new points in the suspension; they are connected by a path $\text{merid}(x)$ through each point on the circle; the figure illustrates the paths through a and b . The n -sphere S^n can be defined by iterating the suspension of the circle

$$S^n = \Sigma^{n-1}(S^1)$$

(e.g. $S^3 = \Sigma S^2 = \Sigma \Sigma S^1$), so the above definition of $K(\mathbb{Z}, n)$ is $\|\!| \Sigma^{n-1}(S^1) \|\!|_n$.

To define $K(G, n)$ in general, we can do an analogous construction: first, we define a space $K(G, 1)$ for any G directly as a higher

²The notation ΣX is standard for suspensions; it is unrelated to the usual type-theoretic notation $\Sigma x:A. B$ for dependent pair types.

inductive type. Then, we define

$$K(G, n) = \|\Sigma^{n-1}(K(G, 1))\|_n$$

The main theorem is that this definition of $K(G, n)$ has the correct homotopy groups: $\pi_n(K(G, n)) = G$ and $\pi_{k \neq n}(K(G, n)) = 1$.

This proof is interesting for several reasons: Like many synthetic proofs, our proof applies not only to traditional models of homotopy theory such as simplicial sets or topological spaces, but to any model of homotopy type theory. Moreover, the proof is not a transcription of a proof in classical algebraic topology, but makes use of some novel type-theoretic methods and tools. It is not yet clear how these type-theoretic proofs relate to proofs in other settings for homotopy theory; we hope that a deeper understanding of this issue will eventually be obtained. Another interesting aspect of this proof is that it builds on existing constructions (such as truncations and suspensions), results (such as Lumsdaine’s proof of the Freudenthal suspension theorem [21]), and methods (such as the encode-decode method [14, 21]). This suggests that these tools may scale to larger proofs and formalizations. Moreover, work on the computational interpretation of homotopy type theory [3, 4, 13, 20] suggests that it has computational content, such as an algorithm for reducing n -paths in $K(G, n)$ to elements of G .

The main content of this paper is an “informalization” of Agda proofs available at www.github.com/dlicata335/hott-agda, in the style of the Homotopy Type Theory book [21]. We have chosen this presentation in order to add to the growing body of evidence that homotopy type theory allows for proofs suitable both for people and for proof assistants, and to give a basis for comparing informal and formal proofs of the same theorem. The construction is conducted in homotopy type theory, by which we mean Martin-Löf type theory extended with the univalence axiom and higher inductive types; see [21] for details. Homotopy type theory can be simulated in Agda by adding univalence as an axiom and postulating individual higher inductive types (see [11]).

As a prerequisite, we encourage the reader to read either [14], or Part I and Chapter 8 of [21]. These introduce the basic constructions of homotopy type theory (such as transport, or path lifting, and ap , a function’s action on paths), as well as the synthetic approach to homotopy theory that is developed further in this paper. The level of detail of the informal proofs is intended to be such that a reader familiar with [21, Chapter 8] could easily fill in the missing details.

In Section 2, we review some preliminary definitions and results. In Section 3, we describe a construction of $K(G, 1)$ and a proof that it has the correct homotopy groups. In Section 4, we prove a lemma that will be used to show that $K(G, 2)$ has the correct homotopy groups. In Section 5, we use these two preliminary results to construct $K(G, n)$ for general n . We discuss the relationship between the proof presented here and the Agda formalization in Section 6.

2. Preliminaries

In this section, we review some basic definitions; these are discussed in [21] and are part of the Agda library that we use as a basis for our formalization.

2.1 n -types and Truncations

One of the new ideas in homotopy type theory is that types can be stratified into levels, depending on what level (if any) their points, paths, paths-between-paths, etc. become trivial. For example, a proposition is a type A for which you can prove $\prod x, y : A. x = y$ (points are trivial); a set, or 0-type is a type A with a proof of $\prod x, y : A. \prod p, q : x = y. p = q$ (paths are trivial). In general, we say that a type A is a -2 -type (or is contractible) if $\Sigma x : A. \prod y : A. x = y$, and that A is an $n + 1$ -type if $\prod x, y : A. n\text{-type}(x = y)$. See [21, Chapter 3] for more details.

Given any type A , there is a type $\|A\|_n$, the n -truncation of A , which is “the best approximation of A as an n -type. This type has a constructor $|-| : A \rightarrow \|A\|_n$, and the recursion principle rule says that to define a function $f : \|A\|_n \rightarrow C$ where C is an n -type, it suffices to give a function $b : A \rightarrow C$ —and then $f(|x|) = b(x)$. More generally, the induction principle says that to define a function $f : \prod x : \|A\|_n. C(x)$ where $C : \|A\|_n \rightarrow n$ -type, it suffices to give $b : \prod x : A. C(|x|)$ —with the same computation rule. Truncations can be implemented by certain higher inductive types [21, Chapter 6.9], so they need not be new a primitive ingredient, but it is conceptually helpful to think of them as a new type constructor.

A dual notion to truncatedness is connectedness: an n -connected type is trivial at and below dimension n . Connectedness can be defined using truncation:

DEFINITION 2.1. [21, Definition 7.5.1] *A type A is n -connected iff $\|A\|_n$ is contractible.*

The n -truncation kills everything above n , so if this is contractible, then the original type was trivial at and below n . For example, a type A is 0-connected when the set-truncation of A is contractible—i.e. the set of points of the type has exactly one element.

2.2 Groups

DEFINITION 2.2: GROUP. A group G is a tuple (dependent pair type) consisting of an underlying set $|G|$ of elements, together with multiplication ($x \odot y$) and inverse (x^{-1}) operations and a unit element e , which satisfy associativity, inverse, and unit laws expressed in terms of the identity type (e.g. $\prod x : |G|. x \odot e = |G|.x$). G is abelian iff $x \odot y = y \odot x$ for all x, y .

The important point is that a group has a *set*, rather than a general *type*, of elements; because of this, the proofs of the group laws are elements of a proposition, and therefore proof-irrelevant, and these groups behave like groups in traditional foundations. To obtain a useful generalization to a less truncated type of elements, it is usually necessary to extend the definition of a group with additional coherences on the proofs of the group laws, such as the pentagon equation for different ways of reassociating $x \odot y \odot z \odot w$.

DEFINITION 2.3: GROUP HOMOMORPHISM. Given groups G and H , a group homomorphism consists of a function $f : |G| \rightarrow |H|$ together with proofs of $f(e_G) = e_H$ and $f(x \odot_G y) = f(x) \odot_H f(y)$.

By convention, we often write G for the underlying set $|G|$.

2.3 Homotopy groups

DEFINITION 2.4. [21, Definition 2.1.7] *A pointed type is a pair (A, a_0) where $a_0 : A$ (i.e. the type of pointed types is $\Sigma A : \text{type}. A$).*

We often refer to a pointed type (A, a_0) by its underlying type A .

DEFINITION 2.5. [21, Definition 2.1.8] *Given a pointed type (A, a_0) , the loop space of A , written $\Omega(A, a_0)$, is a pointed type defined by $\Omega(A, a_0) := (a_0 = a_0, \text{id})$. For positive n , the iterated loop space $\Omega^n(A, a_0)$ is defined by*

$$\begin{aligned} \Omega^1(A, a_0) &::= \Omega(A, a_0) \\ \Omega^{n+1}(A, a_0) &::= \Omega^n(\Omega(A, a_0)) \end{aligned}$$

Thus, the n^{th} loop space is the type of loops at $\text{id}_{\dots \text{id}_{a_0}}$, and gives information about the n -dimensional structure in A . The loop spaces preserve information about levels above n , while the homotopy groups space focus on one dimension at a time:

DEFINITION 2.6. [21, Definition 8.0.1] *Given a pointed type (A, a_0) , the n^{th} homotopy group $\pi_n(A, a_0)$ is defined to be the set $\|\Omega^n(A, a_0)\|_0$; path composition, inverses, and identity induce a group structure.*

2.4 Suspensions

In the introduction, we discussed the notion of suspension informally; formally, it is defined as follows:

DEFINITION 2.7. [21, Section 6.5] *Given a type A , the suspension of A , written ΣA , is a higher inductive type with constructors $N : \Sigma A$ and $S : \Sigma A$ and $\text{merid} : A \rightarrow N =_{\Sigma A} S$*

The corresponding recursion principle says that one can define a function $f : \Sigma A \rightarrow C$ by giving $c_1 : C$ and $c_2 : C$ and $m : A \rightarrow c_1 = c_2$. The computation rules for f are that $f(N) \equiv c_1$ and $f(S) \equiv c_2$ and $\text{ap}_f(\text{merid}(x)) = m(x)$. The induction principle says that one can define a dependent function $f : \Pi z : \Sigma A. C(z)$ by giving $c_1 : C(N)$ and $c_2 : C(S)$ and $m : \Pi x : A. \text{transport}_C(\text{merid}(x), c_1) = c_2$ with analogous computation rules. When we regard ΣA as a pointed type, the implicit base point is N .

3. $K(G, 1)$

Let G be a group. In this section, we define a type $K(G, 1)$ such that $\pi_1(K(G, 1))$ is G and $\pi_k(K(G, 1))$ is trivial otherwise. G does not need to be abelian for $K(G, 1)$, though it is necessary to assume that G is abelian for $K(G, 2)$ and above.

3.1 Definition of $K(G, 1)$

DEFINITION 3.1: $K(G, 1)$. To define $K(G, 1)$, we use a *higher inductive 1-type* with the following constructors:

$K(G, 1)$: 1-type
base	: $K(G, 1)$
loop	: $G \rightarrow (\text{base} =_{K(G, 1)} \text{base})$
loop–ident	: $\text{loop}(e) = \text{id}$
loop–comp	: $\Pi x, y : G. \text{loop}(x \odot y) = \text{loop}(y) \circ \text{loop}(x)$

base is a point (element) of $K(G, 1)$. loop is a function that constructs a path from base to base for each element of G . loop–ident says that the path constructed from the unit element is the identity path, and loop–comp that the path constructed from a multiplication of elements is the composition of the corresponding paths.

By declaring $K(G, 1)$ to be a 1-type, we assert that, while it may have non-trivial paths between points (like $\text{loop}(x)$ for any x), any two paths between paths are identified. By definition, a 1-type has a set of paths, and therefore path equality is a proposition, and therefore any two proofs of path equality are themselves equal. Thus, the statement “ $K(G, 1)$ is a 1-type” is equivalent to

$$\Pi x, y : K(G, 1). \Pi p, q : x = y. \Pi r, s : p = q. r = s$$

For example, there are two proofs

$$r, s : \text{loop}(x \odot y \odot z) = \text{loop}(z) \circ \text{loop}(y) \circ \text{loop}(x)$$

where r is given by using loop–comp twice, and q is given by using associativity of \odot and then using loop–comp twice. Because $K(G, 1)$ is a 1-type, there is a path $r = s$. This is necessary for $K(G, 1)$ to have the correct higher homotopy groups. [5] gives a general reduction of n -truncated higher inductive types to ordinary higher-inductive types. For $K(G, 1)$, this general construction is equivalent to simply adding another higher inductive constructor trunc with the above type expressing “ $K(G, 1)$ is a 1-type”.

Recursion principle In general, the elimination rule for a higher inductive type allows for defining a function by giving a point for each point constructor, a path for each path constructor, etc. In this case, the associated recursion principle says that, to define a function $f : K(G, 1) \rightarrow C$ for some other type C , it suffices to give

- a point $c : C$
- a family of loops $l : G \rightarrow (c =_C c)$

- a path $l(e) = \text{id}$
- a path $l(x \odot y) = l(y) \circ l(x)$
- a proof that C is a 1-type

Then f satisfies the equations

$$\begin{aligned} f(\text{base}) &\equiv c \\ \text{ap}_f(\text{loop}(x)) &= l(x) \end{aligned}$$

One could also give computation rules expressing how ap_{ap_f} acts on loop–comp and loop–ident. However, the computation rules for a constructor are an equality at one dimension higher than the level of the constructor (e.g., the computation rule for the path constructor $\text{loop}(x)$ is a path between paths). loop–comp and loop–ident are proofs of equality of paths, so these computation rules would be paths between proofs of equality of paths, and therefore follow from the fact that C is a 1-type. Thus, we do not even need to state them.

The fact that the recursion principle requires C to be a 1-type is an instance of a general pattern, in which the elimination rule for an n -truncated higher-inductive type allows one to define functions only into other n -types. But considering our explicit definition of $K(G, 1)$ being a 1-type by the extra constructor trunc above, in this specific case it is easy to see that C being a 1-type is exactly what it means for the trunc constructor to be preserved, because translating the type of trunc to C is exactly the statement that C is a 1-type.

It is also helpful to note that the function l and the paths expressing how it acts on unit and multiplication can be packaged together by observing that, for any 1-type C with point $c : C$, there is a group $\Omega(C, c)$ whose underlying set is $c = c$, the type of loops at c , with path identity, composition, and inverses as the group operations. Then, to specify a function f , it suffices to give

- a proof that C is a 1-type
- a point $c : C$
- a group homomorphism from G to $\Omega(C, c)$

Induction principle. There is also a corresponding dependent elimination or induction principle, which says that to prove a family $C : K(G, 1) \rightarrow 1\text{-type}$, it suffices to give

- a point $c : C(\text{base})$
- a family of paths $p : \Pi x : G. \text{transport}_C(\text{loop}(x), c) = c$

such that p preserves identity and composition (see the formalization for details).

3.2 Fundamental Group

Next, we prove

THEOREM 3.2. *The group $\pi_1(K(G, 1))$ is isomorphic to G .*

Proof. To define a group isomorphism, it suffices to give an equivalence on the underlying types that preserves composition. First, we prove that the type $\pi_1(K(G, 1))$ is equivalent to the type G . $\pi_1(K(G, 1))$ is defined to be $\|\Omega(K(G, 1))\|_0$ where $\Omega(K(G, 1))$ is the type $\text{base} =_{K(G, 1)} \text{base}$. Because $K(G, 1)$ is a 1-type, $\Omega(K(G, 1))$ is a 0-type by definition. Therefore $\|\Omega(K(G, 1))\|_0$ is equivalent to $\Omega(K(G, 1))$ because 0-truncating a type that is already 0-truncated has no effect [21, Corollary 7.3.7]. Thus, it suffices to show that $\Omega(K(G, 1)) \simeq G$.

This is proved using the encode-decode method [14, 21]. We have a constructor

$$\text{loop} : G \rightarrow \Omega(K(G, 1))$$

so it suffices to define a function

$$G \rightarrow \Omega(K(G, 1))$$

and show that these two maps are mutually inverse. To do so, we first define a dependent type

$$\text{Codes} : K(G, 1) \rightarrow \text{type}$$

such that the fiber over base , $\text{Codes}(\text{base})$, is equal to G . Then we can define

$$\begin{aligned} \text{encode} &: \prod z:K(G, 1). \text{base} = z \rightarrow \text{Codes}(z) \\ \text{encode}_z(p) &\equiv \text{transport}_{\text{Codes}}(p, e) \end{aligned}$$

The instance $\text{encode}_{\text{base}}$ has type $(\text{base} = \text{base}) \rightarrow \text{Codes}(\text{base})$, which is equal to $\Omega(K(G, 1)) \rightarrow G$ as desired; thus gives us a candidate inverse to loop .

Thus, our remaining tasks are (1) defining Codes , and (2) proving that encode and loop are mutually inverse. Codes will be defined by recursion on $K(G, 1)$, so we need to give a 1-type C along with a point $c : C$ and a group homomorphism from G to $\Omega(C, c)$. We would like to take C to be type, but the universe of all types is not a 1-type. However, the collection of all 0-types (sets) is a 1-type [21, Theorem 7.1.11], so we can choose C to be set , the type of all sets. Thus, we need to give a set S and a group homomorphism from G to $\Omega(\text{set}, S)$. We clearly should take S to be the underlying set of the group G , so that, by the equations of the elimination rule, $\text{Codes}(\text{base}) \equiv G$. Fortunately, G is a set by the definition of a group.

To complete the definition of Codes , we therefore need a group homomorphism from G to $\Omega(\text{set}, G)$. This consists of a function $G \rightarrow (G = G)$ which sends e to id and sends multiplication to composition. By univalence, it suffices to give a function $G \rightarrow (G \simeq G)$ that maps each element of G to an equivalence. We choose the function that sends $x : G$ to the equivalence $- \odot x$, which given $y : G$ computes $y \odot x$; this has inverse $- \odot x^{-1}$ by the group laws. Using the group laws, one can also see that this sends e to id (because multiplication by the unit is the identity function) and multiplication to composition of paths (because of associativity), and as required. To summarize, this definition satisfies the following:

$$\begin{aligned} \text{Codes} &: K(G, 1) \rightarrow \text{set} \\ \text{Codes}(\text{base}) &\equiv G \\ \text{transport}_{\text{Codes}}(\text{loop}(x), y) &= y \odot x \end{aligned}$$

The third line follows from the defining equation for $\text{ap}_{\text{Codes}}(\text{loop}(x))$ together with the computation rule for transporting along univalence.

We now check that the two composites are the identity. First, for $\text{encode}_{\text{base}} \circ \text{loop}$, given $x : G$ we have

$$\begin{aligned} &\text{encode}_{\text{base}}(\text{loop}(x)) \\ \equiv &\text{transport}_{\text{Codes}}(\text{loop}(x), e) && \text{definition} \\ = &e \odot x && \text{transport for Codes} \\ = &x && \text{group law} \end{aligned}$$

In the other direction, we must check that for any $p : \Omega(K(G, 1))$, $\text{loop} \circ \text{encode}_{\text{base}}(p) = p$. To do so, we generalize loop to a function decode

$$\text{decode} : \prod z:K(G, 1). \text{Codes}(z) \rightarrow \text{base} = z$$

which is defined by induction on z . In the case for base , it suffices to give a function $G \rightarrow \Omega(K(G, 1))$, so we use loop . Next, we must give a family of paths

$$\text{transport}_{z.\text{Codes}(z) \rightarrow \text{base}=z}(\text{loop}(x), \text{loop}) = \text{loop}$$

for each $x : G$. By reducing transport in function and identity types and using function extensionality, it suffices to show for all $y : G$ that

$$\begin{aligned} &\text{loop}(\text{transport}_{\text{Codes}}(\text{loop}(x), y)) \\ = &\text{loop}(y \odot x) && \text{by definition Codes} \\ = &\text{loop}(x) \circ \text{loop}(y) && \text{by loop-comp} \end{aligned}$$

The obligation to show that this family of paths preserves identity and composition is immediate because we are eliminating into a set, the path space of the 1-type $K(G, 1)$.

Having defined decode , we check that

$$\prod z:K(G, 1). \prod p:\text{base} = z. \text{decode}_z(\text{encode}_z(p)) = p$$

Because we have generalized the problem, we can apply path induction, and it suffices to check the case where z is base and p is id . In this case, the problem reduces to showing that

$$\text{loop}(e) = \text{id}$$

which is proved by loop-idem . Instantiating this generalization to base and reducing the type gives a function of type

$$\prod p:\text{base} = \text{base}. \text{loop}(\text{encode}_{\text{base}}(p)) = p$$

as required.

Thus, we have established that loop and $\text{encode}_{\text{base}}$ are mutually inverse, which gives an equivalence.

To see that this equivalence preserves composition, it is enough to observe that the overall equivalence $G \simeq \pi_1(K(G, 1))$ is a composition of $G \simeq \Omega(K(G, 1))$ and $\Omega(K(G, 1)) = \|\Omega(K(G, 1))\|_0$. The map $G \rightarrow \Omega(K(G, 1))$ is loop , which preserves composition by construction. The map $\Omega(K(G, 1)) \rightarrow \|\Omega(K(G, 1))\|_0$ is the truncation constructor $\|- \|_0$, which preserves identity and composition because the identity and composition of $\|\Omega(K(G, 1))\|_0$ are by definition the lifting of identity and composition in $K(G, 1)$ [21, Example 6.11.4]. Thus, the group G is isomorphic to the group $K(G, 1)$. \square

3.3 Higher homotopy groups

Because $K(G, 1)$ is a 1-type (by construction), $\pi_k(K(G, 1)) = 1$ for all $k > 1$ [21, Theorem 8.3.1]. Intuitively, 1-types have no non-trivial 2-cells or above, so their higher homotopy groups are trivial. Thus, we have

COROLLARY 3.3. *The type $K(G, 1)$ has $\pi_1(K(G, 1)) = G$ and $\pi_k(K(G, 1))$ trivial otherwise.*

4. Suspension of a type with an h-structure

In this section, we prove a lemma that will be useful for constructing $K(G, 2)$; it is essentially the proof that $\pi_2(K(G, 2)) = G$, in more generality.

4.1 H-structures

DEFINITION 4.1. *Given a pointed type (A, a_0) , a coherent h-structure on (A, a_0) consists of*

- A multiplication operation $\odot : A \rightarrow A \rightarrow A$
- A left unitor $\text{unitl} : \prod a:A. a_0 \odot a = a$
- A right unitor $\text{unitr} : \prod a:A. a \odot a_0 = a$
- A coherence $\text{unitcoh} : \text{unitl}(a_0) = \text{unitr}(a_0)$
- A proof that for every $a : A$, $a \odot -$ is an equivalence.

A coherent h-structure equips A with a multiplication operation, such that a_0 is a unit for the multiplication (coherently, in the sense that the left and right unitors agree on a_0), and multiplication by any element is an equivalence. The notion of h-structure is due to Hopf. We often drop the word ‘‘coherent’’, because we will only consider coherent ones. For example:

LEMMA 4.2. *If G is abelian, then there is an h-structure on $K(G, 1)$.*

Proof. Thinking of $K(G, 1)$ as a groupoid with one object and an arrow for each $g : G$, the multiplication $K(G, 1) \rightarrow K(G, 1) \rightarrow K(G, 1)$ is a functor from the $K(G, 1)$ to $K(G, 1)^{K(G, 1)}$, which sends

the object to the identity functor, and each element $z : G$ to a natural transformation from the identity functor to itself. Such a natural transformation is given by a single arrow of $K(G, 1)$, which we take to be z ; this is natural because G is abelian. In type theory, this argument is rendered as follows: The multiplication is defined by recursion on its first argument (note that the result is 1-type), where the image of base is $\lambda z.z$, so for loop we must provide a group homomorphism from G to $\Omega(K(G, 1) \rightarrow K(G, 1), \lambda z.z)$. On elements, we need to map each element $x : G$ to a path from the identity function to itself; by function extensionality, it suffices to give

$$\Pi z : K(G, 1). z = z$$

This is defined by induction on z (the result is a path in a 1-type and therefore still a 1-type), sending base to $\text{loop}(x)$. To complete the induction, we must check that

$$\Pi y : G. \text{transport}_{x.x=y}(\text{loop}(y), \text{loop}(x)) = \text{loop}(x)$$

The calculation requires reducing the transport and applying functoriality of loop and the group laws, including commutativity. The remaining conditions for the induction principle are trivial because of the level of the type we are eliminating into. This completes the definition of the homomorphism on elements. It can be proved that it preserves identity and composition using $K(G, 1)$ induction, appealing to loop–ident and loop–comp.

To complete the definition of an h-structure, we must give left and right unitors and show that multiplication by any element is an equivalence. All of these are defined by $K(G, 1)$ induction, with short calculations. \square

4.2 π_2 of a Suspension

For the suspension type (Section 2), the merid constructor maps A into the paths in the suspension ΣA . Thus, one might think that it moves the homotopy groups up by one, shifting π_1 to π_2 , etc. However, the structure of ΣA is more complicated than that. For example, we can construct $K(G, 1)$ for any group G , but π_2 of any type is abelian by the Eckmann-Hilton argument [21, Theorem 2.1.6], so $\pi_2(\Sigma K(G, 1))$ cannot always be equal to $\pi_1(K(G, 1)) = G$. However, in certain special circumstances, the suspension does shift π_1 to π_2 . One such circumstance is as follows:

THEOREM 4.3. *Let (A, a_0) be a pointed, 0-connected, 1-type with an h-structure. Then $\pi_2(\Sigma A, N) = \pi_1(A, a_0)$.*

The idea is that A is like $K(G, 1)$ in the sense that only its paths are non-trivial: being 0-connected means that its set of points is trivial; being a 1-type means that its paths between paths (and higher) are trivial. Moreover, it has an h-structure, which ensures that its π_1 is abelian. The theorem states that under these conditions, the suspension lifts $\pi_1(A)$ to $\pi_2(\Sigma A)$.

For the remainder of this section, let (A, a_0) be a pointed 0-connected 1-type with an h-structure \odot on (A, a_0) . We call attention to one lemma used in the proof of Theorem 4.3.

LEMMA 4.4. *For all $a, a' : A$,*

$$|\text{merid}(a \odot a')| =_{\|\Omega(\Sigma A)\|_1} |\text{merid}(a) \circ ! \text{merid}(a_0) \circ \text{merid}(a')|$$

A priori, it is not obvious that the merid constructor preserves the h-structure multiplication on A . This lemma says that it does, at least under an appropriate truncation: merid of a multiplication is the same as the composite going “down” along a' , “up” along the distinguished base point a_0 (which is the unit of the multiplication), and then “down” along a . This holds when we consider these paths in the 1-truncation of the loop space. This lemma arises naturally in the proof below, but its proof requires a bit of cleverness. To prove it, we use the following induction principle:

LEMMA 4.5. *Suppose A is 0-connected with $a_0 : A$, and that $P : A \rightarrow A \rightarrow 0$ -type. Then to define a function $\Pi x, y : A. P(x, y)$, it suffices to separately give*

- $f : \Pi y : A. P(a_0, y)$
- $g : \Pi x : A. P(x, a_0)$
- such that they agree on $a_0 : f(a_0) = g(a_0)$.

This statement of the lemma is a special case of a principle formulated in homotopy type theory by Lumsdaine (see [21, Lemma 8.6.2] for the proof), which expresses the connectivity of the map from the wedge into the product. It says that we can prove something about the whole product $A \times A$ by separately considering each “axis” where x and y are fixed to be a_0 .

Proof of Lemma 4.4. We apply Lemma 4.5 to A and a_0 , with $P(a, a')$ defined by

$$|\text{merid}(a \odot a')| =_{\|\Omega(\Sigma A)\|_1} |\text{merid}(a) \circ (! \text{merid}(a_0) \circ \text{merid}(a'))|$$

Thus, it suffices to show first that

$$|\text{merid}(a_0 \odot a)| = |\text{merid}(a_0) \circ (! \text{merid}(a_0) \circ \text{merid}(a))|$$

which holds by the left unit law for $a_0 \odot -$ and an inverse/unit law for paths. Second, we must show that

$$|\text{merid}(a \odot a_0)| = |\text{merid}(a) \circ (! \text{merid}(a_0) \circ \text{merid}(a_0))|$$

which holds by the right inverse law for $- \odot a_0$ and a different inverse/unit law for paths. Finally, we must show that the above two proofs agree when they are each used to prove

$$|\text{merid}(a_0 \odot a_0)| = |\text{merid}(a_0) \circ (! \text{merid}(a_0) \circ \text{merid}(a_0))|$$

This is where we use the coherence condition imposed on h-structures; we also use path induction to show that the two inverse/unit laws for paths agree. \square

With this lemma in hand, we turn to the proof of the main theorem:

Proof of Theorem 4.3. Overall, we calculate as follows:

$$\begin{aligned} & \pi_2(\Sigma A, N) \\ &= \|\Omega(\Omega(\Sigma A, N), \text{id})\|_0 && \text{definition of } \pi_2, \Omega_2 \\ &= \Omega(\|\Omega(\Sigma A, N)\|_1, |\text{id}|) && [21, Lemma 7.3.12] \\ &= \Omega(A, a_0) && \text{main lemma} \\ &= \pi_1(A, a_0) && A \text{ is a 1-type, so } \Omega(A, a_0) \text{ is a set} \end{aligned}$$

The fourth line uses the fact that truncations and loop spaces can be commuted, incrementing the truncation level by one as it moves inside: the n -truncation of paths in A are the same as the paths in the $n + 1$ -truncation of A (which kills all cells at one level higher). The main lemma, used in the next line, is that

$$\|\Omega(\Sigma A, N)\|_1 \simeq A$$

and that this equivalence sends a_0 to $|\text{id}|$.

This equivalence is constructed by another application of the encode-decode method. The technique is a small elaboration of the one used above, where we characterized a loop space by proving a statement of the form $\Omega(X, x_0) = Y$. In the present theorem, it is not possible to characterize the entire loop space (which would imply a characterization of all higher homotopy groups), but we can get partial information by characterizing a truncation, proving a statement of the form $|\Omega(X, x_0)|_1 = Y$.

Let

$$\begin{aligned} P : \Sigma A &\rightarrow \text{type} \\ P(x) &:= \|\mathbb{N} =_{\Sigma A} x\|_1 \end{aligned}$$

Our goal is to construct another fibration

$$\text{Codes} : \Sigma A \rightarrow 1\text{-type}$$

such that $\text{Codes}(\mathbb{N}) \equiv A$, and to use it to prove that $\|\Omega(\Sigma(A), \mathbb{N})\|_1 = A$ by proving that $P(\mathbb{N}) = \text{Codes}(\mathbb{N})$.

Let \odot be the multiplication operation of the h-structure on (A, a_0) . We define Codes by suspension recursion, with

$$\begin{aligned}\text{Codes}(\mathbb{N}) &:= A \\ \text{Codes}(\mathbb{S}) &:= A \\ \text{apCodes}(\text{merid}(x)) &:= \text{ua}(x \odot -)\end{aligned}$$

That is, the fiber over both \mathbb{N} and \mathbb{S} is A , and the $\text{merid}(x)$ path is sent to multiplication by x , which is an equivalence by the definition of h-structure, and therefore determines a path by univalence. Since A is a 1-type, Codes actually defines a fibration of 1-types.³ A short calculation shows that this definition satisfies

$$\begin{aligned}\text{transport}_{\text{Codes}}(\text{merid}(a), a') &= a \odot a' \\ \text{transport}_{\text{Codes}}(!\text{merid}(a_0)), a &= a\end{aligned}$$

essentially because transporting at a applies the equivalence $a \odot -$ that we put into the definition of Codes (and, in the second case, $a_0 \odot -$ is the identity function, so its inverse is also the identity).

Because Codes is a 1-type, we can define

$$\text{encode} : \Pi x : \Sigma A. P(x) \rightarrow \text{Codes}(x)$$

by truncation recursion, where $p : \mathbb{N} = x$ is sent to $\text{transport}_{\text{Codes}}(p, a_0)$.

In the other direction, observe that we have a map

$$\begin{aligned}\text{decode}' : A &\rightarrow \|\mathbb{N} = \mathbb{N}\|_1 \\ \text{decode}'(a) &= |\text{merid}(a_0) \circ \text{merid}(a)|\end{aligned}$$

That is, given an a , to get a loop at \mathbb{N} in the suspension, we can go down along a and back up along the base point a_0 .

Using these two equations, it is easy to check the first composite, which shows that

$$\text{encode}_{\mathbb{N}}(\text{decode}'(a)) = a$$

The calculation is as follows:

$$\begin{aligned}&\text{encode}(\text{decode}'(a)) \\ &= \text{encode}(|!\text{merid}(a_0) \circ \text{merid}(a)|) \\ &= \text{transport}_{\text{Codes}}(!\text{merid}(a_0) \circ \text{merid}(a), a_0) \\ &= \text{transport}_{\text{Codes}}(!\text{merid}(a_0), \text{transport}_{\text{Codes}}(\text{merid}(a), a_0)) \\ &= \text{transport}_{\text{Codes}}(!\text{merid}(a_0), a \odot a_0) \\ &= \text{transport}_{\text{Codes}}(!\text{merid}(a_0), a) \\ &= a\end{aligned}$$

For the other composite, we first generalize decode' to

$$\text{decode} : \Pi x : \Sigma A. \text{Codes}(x) \rightarrow P(x)$$

This is defined by suspension induction. In the case for \mathbb{N} , we need a function $A \rightarrow \|\mathbb{N} = \mathbb{N}\|_1$, for which we choose decode' . In the case for \mathbb{S} , we need a function $A \rightarrow \|\mathbb{N} = \mathbb{S}\|_1$, which we define by

$$a \mapsto |\text{merid}(a)|$$

(which is similar to decode' , but it does not come ‘‘back up’’ along $\text{merid}(a_0)$). To complete the definition, we need, to give a case for $\text{merid}(a)$, which is a path showing that the image of north (decode') and the image of south ($a \mapsto |\text{merid}(a)|$) are equal up to $\text{merid}(a)$:

$$\text{transport}_{x.\text{Codes}(x) \rightarrow P(x)}(\text{merid}(a), \text{decode}'(a)) = (a \mapsto |\text{merid}(a)|)$$

Applying some reductions for transport at function, path, and using the above reduction rule for $\text{transport}_{\text{Codes}}(\text{merid}(a), -)$ shows that it suffices to give, for all a' , a path

$$|\text{merid}(a) \circ !\text{merid}(a_0) \circ \text{merid}(a')| = |\text{merid}(a \odot a')|$$

³ We elide the proofs that A is a 1-type in the \mathbb{N} and \mathbb{S} cases and the path between them (which is trivial because being an n -type is a proposition, so a path between two n -types (A, nA) and (B, nB) is the same as a path between A and B).

This is exactly Lemma 4.4.

Now that we have defined decode , we can prove that

$$\Pi x : \Sigma A. \Pi p : P(x). \text{decode}_x(\text{encode}_x(p)) = p$$

Because the conclusion is a path in a 1-type, it is a 1-type, we can do truncation induction on p , to get a path $p' : \mathbb{N} = x$. By path induction, it suffices to show

$$\text{decode}_{\mathbb{N}}(\text{encode}_{\mathbb{N}}(\text{id})) = \text{id}$$

After expanding the definitions, this follows from a simple path manipulation.

By construction, we have that $\text{decode}(\mathbb{N}) \equiv \text{decode}'$, so in particular we have that

$$\Pi p : P(\mathbb{N}). \text{decode}'(\text{encode}_{\mathbb{N}}(p)) = p$$

which complete the proof that decode' and $\text{encode}_{\mathbb{N}}$ are mutually inverse, proving the main lemma that $\|\Omega(\Sigma A, \mathbb{N})\|_1 \simeq A$.

Finally, to show that this equivalence is a group isomorphism, we must check that the equivalence of types $\pi_2(\Sigma A, \mathbb{N}) = \pi_1(A, a_0)$ that we have constructed preserves composition. This can be proved by inspecting the above equivalence. Modulo shuffling truncations around, the main step of the equivalence is to apply the main lemma *inside of* $\Omega(-)$. Thus, the main step of the functions that this equivalence constructs are given by $\text{ap}(f)$ for some f , and ap always preserves composition [21, Lemma 2.2.2]. \square

5. $K(G, n)$

Next, we construct $K(G, n)$ for arbitrary n . As discussed in the introduction, the idea for constructing $K(G, n)$ is to suspend $K(G, 1)$ $n - 1$ times and then truncate. First, we prove a lemma about iterated suspensions.

5.1 Iterated Suspensions

DEFINITION 5.1. *Suppose a pointed type (A, a_0) that is 0-connected. Define $\Sigma^n(A)$ by induction on A , with*

$$\begin{aligned}\Sigma^0(A) &:= A \\ \Sigma^{n+1}(A) &:= \Sigma(\Sigma^n(A))\end{aligned}$$

Observe that $\Sigma^n(A)$ is pointed, with a_0 the point of $\Sigma^0(A)$ and \mathbb{N} the point of $\Sigma^{n+1}(A)$. Additionally, in general, if C is n -connected, then ΣC is $n + 1$ -connected [21, 8.2.1]; iterating this fact starting at 0 gives that $\Sigma^n(A)$ is n -connected, because A is 0-connected by assumption.

In the previous section, we discussed how the suspension does not always shift the homotopy group up by one, and investigated one circumstance in which it does. The Freudenthal suspension theorem [25] is a general result that gives additional circumstances in which the suspension preserves the homotopy groups of a space, depending on the connectivity of the space being suspended. We use the following corollary:

COROLLARY 5.2. *If A is n -connected and pointed, with $n \geq 0$, then*

$$\|A\|_{2n} = \|\Omega(\Sigma A, \mathbb{N})\|_{2n}$$

Lumsdaine’s proof appears as [21, Corollary 8.6.4]. From the suspension theorem, we can deduce a corollary about the homotopy groups of iterated suspensions:

LEMMA 5.3. *Suppose A is n -connected and pointed, with $n \geq 0$, and let n, k be positive numbers such that $k \leq 2n - 2$. Then*

$$\pi_{k+1}(\Sigma^n(A)) = \pi_k(\Sigma^{n-1}(A))$$

Proof. The proof is a simple calculation; we elide the base points to make it easier to read:

$$\begin{aligned}
& \pi_{k+1}(\Sigma^n(A)) \\
= & \|\Omega^k(\Omega(\Sigma^n(A)))\|_0 && \text{definition } \pi, \Omega^{k+1} \\
= & \Omega_k(\|\Omega(\Sigma^n(A))\|_k) && [21, \text{Corollary 7.3.14}] \\
= & \Omega_k(\|\Omega(\Sigma(\Sigma^{n-1}(A)))\|_k) && \text{definition } \Sigma^n(A), n \text{ is positive} \\
= & \Omega_k(\|\Sigma^{n-1}(A)\|_k) && \text{Corollary 5.2} \\
= & \|\Omega_k(\Sigma^{n-1}(A))\|_0 && [21, \text{Corollary 7.3.14}] \\
= & \pi_k(\Sigma^{n-1}(A)) && \text{definition } \pi
\end{aligned}$$

□

Aside from expanding definitions, the only lemmas necessary are that $\|\Omega_k(A)\|_n = \Omega_k(\|A\|_{n+k})$, which is [21, Corollary 7.3.14], and the corollary of the Freudenthal suspension theorem described above. We apply Corollary 5.2 with $n = 0$, observing that A is 0-connected by assumption, so $\Sigma^{n-1}(A)$ is $n - 1$ -connected as mentioned above. Thus, the suspension theorem tells us that $\|\Omega(\Sigma(\Sigma^n(A)))\|_{2n-2} = \|\Sigma^{n-1}(A)\|_{2n-2}$. Because $k \leq 2n - 2$ by assumption, we also have

$$\|\Omega(\Sigma(\Sigma^n(A)))\|_k = \|\Sigma^{n-1}(A)\|_k$$

because we can apply $\| - \|_k$ to both sides and use the fact that $\| \|A\|_n \|_m = \|A\|_{\min(n,m)}$ [21, Lemma 7.3.15], and the min in this case is k .

To see that this equivalence preserves composition, it is enough to observe once again that, modulo shuffling truncations, the main step is applying an equivalence inside Ω^k , and is therefore an instance of ap , which always preserves composition.

5.2 Constructing $K(G, n)$

Finally, we tie all of these results together to construct $K(G, n)$.

Let (A, a_0) be a pointed, 0-connected 1-type with an h-structure, as in Section 4. The idea is that A is like $K(G, 1)$, so we use the following indexing:

$$K'(A, n) := \|\Sigma^{n-1}(A)\|_n$$

Observe that $K'(A, n)$ is $(n - 1)$ -connected, because $\Sigma^{n-1}(A)$ is $(n - 1)$ -connected, and truncation preserves connectedness (adding paths can only help). $K'(A, n)$ is also pointed, with the base point being $|b|$, where b is the base point of $\Sigma^{n-1}(A)$.

THEOREM 5.4. *For all positive numbers n and k , $\pi_n(K'(A, n)) = \pi_1(A)$ and $\pi_k(K'(A, n))$ is trivial otherwise.*

Proof. For $k < n$ (below the diagonal), π_k is trivial because $K'(n, A)$ is $n - 1$ connected, and $\pi_k(A)$ of an $n - 1$ -connected space is trivial whenever $k \leq n - 1$ [21, Lemma 8.3.2].

For $k > n$ (above the diagonal), $K'(A, n)$ is defined to be an n -type, so π_k is trivial by [21, Lemma 8.5.1].

For $k = n$ (on the diagonal), we reason as follows. If $n = 1$, then we need to show that

$$\pi_1(K'(A, 1)) = \pi_1(A)$$

By definition $\pi_1(K'(A, 1)) = \|\Omega(\|A\|_1)\|_0$ and $\pi_1(A) = \|\Omega(A)\|_0$. These are equal because A was assumed to be a 1-type, so $\|A\|_1 = A$ [21, Corollary 7.3.7].

If $n = 2$, then we need to show that

$$\pi_2(K'(A, 2)) = \pi_1(A)$$

We calculate as follows:

$$\begin{aligned}
& \pi_2(K'(A, 2)) \\
= & \|\Omega^2(\|\Sigma A\|_2)\|_0 && \text{definition} \\
= & \|\|\Omega^2(\Sigma A)\|_0\|_0 && [21, \text{Corollary 7.3.14}] \\
= & \|\Omega^2(\Sigma A)\|_0 && [21, \text{Corollary 7.3.7}] \\
= & \pi_2(\Sigma A) && \text{definition} \\
= & \pi_1(A) && \text{Theorem 4.3}
\end{aligned}$$

In the final line, observe that A satisfies the assumptions of Theorem 4.3 by assumption.

Finally, if $n > 2$, then we must show that $\pi_n(K'(A, n)) = \pi_1(A)$. By the inductive hypothesis $\pi_{n-1}(K'(A, n - 1)) = \pi_1(A)$, so it suffices to show that

$$\pi_{n+1}(K'(A, n + 1)) = \pi_n(K'(A, n))$$

This is almost just Lemma 5.3, but we need to deal with the outer truncations in the definition of $K'(A, n)$. First, we can show that for any l , π_l is unchanged by this extra truncation:

$$\begin{aligned}
& \pi_l(K'(A, l)) \\
= & \|\Omega^l(\|\Sigma^{l-1}(A)\|_l)\|_0 \\
= & \Omega^l(\|\|\Sigma^{l-1}(A)\|_l\|_l) \\
= & \Omega^l(\|\Sigma^l(A)\|_l) \\
= & \pi_l(\Sigma^{l-1}(A))
\end{aligned}$$

The proof uses [21, Corollary 7.3.14] and [21, Corollary 7.3.7].

Using this fact twice, we can calculate as follows:

$$\begin{aligned}
& \pi_{n+1}(K'(A, n + 1)) \\
= & \pi_{n+1}(\Sigma^n(A)) \\
= & \pi_n(\Sigma^{n-1}(A)) && \text{Lemma 5.3} \\
= & \pi_n(K'(A, n))
\end{aligned}$$

We appeal to Lemma 5.3 with $n = k$; the condition that $k \leq 2n - 2$ is satisfied whenever n is at least 2, which is the assumption here. The fact that we cannot appeal to stability when $n = 2$, is why we needed to prove Lemma 4.3 separately; this analogous to the fact that the Freudenthal suspension theorem implies that $\pi_3(S^3) = \pi_2(S^2)$ but not that $\pi_2(S^2) = \pi_1(S^1)$.

One can calculate that this equivalence preserves composition, using the fact that Lemma 5.3 and Lemma 4.3 do. □

Finally, given an abelian group G , we can define

$$K(G, n) := K'(K(G, 1), n)$$

and prove

COROLLARY 5.5. *If G is an abelian group, then $\pi_n(K(G, n)) = G$ and $\pi_k(K(G, n))$ is trivial otherwise.*

Proof. A simple induction shows that $K(G, 1)$ is 0-connected. It is a 1-type by construction. By Lemma 4.2, it has an h-structure. Therefore, by the above result applied with A as $K(G, 1)$, we have that $\pi_n(K(G, n)) = \pi_n(K'(K(G, 1), n)) = \pi_1(K(G, 1))$, which is equal to G by Theorem 3.2. By the above result with $A = K(G, 1)$, $\pi_k(K(G, n)) = \pi_k(K'(K(G, 1), n)) = 1$ otherwise. This preserves composition because Theorem 5.4 and Corollary 3.3 do. This completes the construction of $K(G, n)$. □

6. Formalization

An Agda formalization of these results is available at <http://www.github.com/dlicata335/hott-agda>, starting in the file `homotopy/KGn.agda`. The formalization of the results specific to $K(G, n)$, including a proof of the corollary of the Freudenthal suspension theorem that we used (Corollary 5.2), is about 1000 lines

of code. Without Freudenthal, it is about 750 lines of code, which is pleasingly close to the TeX source for the body of this paper, which is about 800 lines of code. This is certainly an apples-to-oranges comparison, since the paper includes much more explanation, while the Agda proof includes all details. But it at least suggests that the overhead of formalizing homotopy theory in this style is low relative to giving an informal exposition. The proof builds on a significant library of around 10,000 lines of code, which includes the lemmas that we cited from the Homotopy Type Theory book [21]. The fact that the proofs are small relative to the library (much of which was written before this particular formalization) is encouraging, because it suggests that the library contains relevant and reusable results.

The presentation in this paper was derived from the formalization, and, for the most part, follows it closely. In this paper, we have provided citations to the Homotopy Type Theory book [21] for theorems that appear there. The formalization, much of which was written before/independently of the book, sometimes has different proofs of these theorems than the book does. In the informal presentation, we have elided some details, and have adopted some abuses of notation. For example, we sometimes write $A : n$ -type, when the real element of that type is a pair (A, p) where $p : n\text{-type}(A)$, and we sometimes refer to a pointed type (A, a_0) by its underlying type A . Some of these would be supported by the canonical structures and type class mechanism of Coq, which Agda does not provide. We also abbreviate some calculations, e.g. relying on the reader to apply “reduction” rules involving transport in particular fibrations. These steps are propositional equalities in Agda, but they have a computational flavor, and might be computation rules in future proof assistants. Finally, the index arithmetic for things like π_n is messy in the current version of our library, because there are different types for positive numbers, natural numbers, and numbers ≥ -2 ; it would be helpful to engineer a better integer library that could support all these types.

The one aspect of the proof that we have not yet formalized is the proofs that the equivalences constructed preserve composition. In Agda, this would be a rather tedious proof, because the equivalences are constructed using many non-computational steps involving univalence and higher-inductives. In a future proof assistant for homotopy type theory with good computational properties, the calculation should be much simpler, because normalizing the equivalences should do much of the work. Agda code for the part of the proof not done by normalization would be comparable in size to the three paragraphs we have spent on this aspect of the proof here.

7. Conclusion

In this paper, we have described a construction of Eilenberg-MacLane spaces $K(G, n)$, illustrating many tools and techniques of using homotopy type theory for computer-checked proofs in homotopy theory. In future work, we plan to build on this formalization to explore the applications mentioned in the introduction. In particular, many computational techniques exist in the literature for computing cohomology groups of spaces and it is an interesting question to see if these can similarly be implemented in type theory. Another piece of future work would be to investigate whether work on the computational interpretation of homotopy type theory [4] can be used to simplify the proofs given here.

Acknowledgments

We thank the participants in the Institute for Advanced Study special year on univalent foundations, especially Peter LeFanu Lumsdaine and Guillaume Brunerie and Mike Shulman, for helpful discussions about this work.

References

- [1] J. F. Adams. *Stable homotopy and generalised homology*. University of Chicago Press, Chicago, Ill., 1974. Chicago Lectures in Mathematics.
- [2] S. Awodey and M. Warren. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 2009.
- [3] B. Barras, T. Coquand, and S. Huber. A generalization of Takeuti-Gandy interpretations. To appear in *Mathematical Structures in Computer Science*, 2013.
- [4] M. Bezem, T. Coquand, and S. Huber. A model of type theory in cubical sets. Preprint, September 2013.
- [5] G. Brunerie. Truncations and truncated higher inductive types. <http://homotopytypetheory.org/2012/09/16/truncations-and-truncated-higher-inductive-types/>, 2012.
- [6] Coq Development Team. *The Coq Proof Assistant Reference Manual, version 8.2*. INRIA, 2009. Available from <http://coq.inria.fr/>.
- [7] N. Gambino and R. Garner. The identity type weak factorisation system. *Theoretical Computer Science*, 409(3):94–109, 2008.
- [8] R. Garner. Two-dimensional models of type theory. *Mathematical Structures in Computer Science*, 19(4):687–736, 2009.
- [9] M. Hofmann and T. Streicher. The groupoid interpretation of type theory. In *Twenty-five years of constructive type theory*. Oxford University Press, 1998.
- [10] C. Kapulkin, P. L. Lumsdaine, and V. Voevodsky. The simplicial model of univalent foundations. arXiv:1211.2851, 2012.
- [11] D. R. Licata. Running circles around (in) your proof assistant; or, quotients that compute. <http://homotopytypetheory.org/2011/04/23/running-circles-around-in-your-proof-assistant/>, April 2011.
- [12] D. R. Licata and G. Brunerie. $\pi_n(s^n)$ in homotopy type theory. In *Certified Programs and Proofs*, 2013.
- [13] D. R. Licata and R. Harper. Canonicity for 2-dimensional type theory. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2012.
- [14] D. R. Licata and M. Shulman. Calculating the fundamental group of the circle in homotopy type theory. In *IEEE Symposium on Logic in Computer Science*, 2013.
- [15] P. L. Lumsdaine. Weak ω -categories from intensional type theory. In *International Conference on Typed Lambda Calculi and Applications*, 2009.
- [16] P. L. Lumsdaine. Higher inductive types: a tour of the menagerie. <http://homotopytypetheory.org/2011/04/24/higher-inductive-types-a-tour-of-the-menagerie/>, April 2011.
- [17] P. L. Lumsdaine and M. Shulman. Higher inductive types. In preparation, 2013.
- [18] U. Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Chalmers University of Technology, 2007.
- [19] M. Shulman. Homotopy type theory VI: higher inductive types. http://golem.ph.utexas.edu/category/2011/04/homotopy_type_theory_vi.html, April 2011.
- [20] M. Shulman. Univalence for inverse diagrams and homotopy canonicity. To appear in *Mathematical Structures in Computer Science*; arXiv:1203.3253, 2013.
- [21] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations Of Mathematics*. Institute for Advanced Study, 2013. Available from homotopytypetheory.org/book.
- [22] B. van den Berg and R. Garner. Types are weak ω -groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011.
- [23] V. Voevodsky. Univalent foundations of mathematics. Invited talk at WoLLIC 2011 18th Workshop on Logic, Language, Information and Computation, 2011.
- [24] M. A. Warren. *Homotopy theoretic aspects of constructive type theory*. PhD thesis, Carnegie Mellon University, 2008.
- [25] G. W. Whitehead. *Elements of homotopy theory*, volume 61 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1978. ISBN 0-387-90336-4.