Stockholm University

Synthetic approaches to cohomology, homology and homotopy

Axel Ljungström

Doctoral Thesis in Computational Mathematics at Stockholm University, Sweden 2025

Synthetic approaches to cohomology, homology and homotopy

Axel Ljungström

Academic dissertation for the Degree of Doctor of Philosophy in Computational Mathematics at Stockholm University to be publicly defended on Wednesday 21 May 2025 at 14.00 in Lärosal 10, Hus 2, Vån 2, Albano, Albanovägen 18 and online via Zoom, public link is available at the department website.

Abstract

This thesis is based on five papers on the development of synthetic homotopy theory in homotopy type theory (HoTT), a relatively recent system of mathematics which extends Martin-Löf type theory with higher inductive types and univalence. The thesis is, in particular, concerned with the development of (co)homology theories and operations, but it also covers other (often related) cornerstone results from homotopy theory. Most results presented here have been *computer formalised*, i.e. digitally verified by a computer, in the verification software (or *proof assistant*) Cubical Agda.

Paper I presents a construction and computer formalisation of cohomology rings in HoTT. To this end, the associativity of cup products is proved – a seemingly easy problem which has turned out to be rather difficult in HoTT due to complicated coherences which arise when attempting a naïve proof. The paper also contains various computations of cohomology groups and rings, the Gysin sequence and a discussion of computational aspects of the computer implementation in Cubical Agda. The paper, in many ways, serves as an introduction to the remainder of the thesis.

Paper II presents a computer formalisation of Brunerie's (2016) computation of the fourth homotopy group of the 3-sphere. In addition to this, we provide a vastly simplified version of Brunerie's proof by introducing a new way of computing (both by hand and by normalisation in Cubical Agda) the so-called *Brunerie number*, i.e. the order of the homotopy group in question.

Paper III is devoted to solving another surprisingly difficult problem in HoTT, namely that of showing that the smash product is (1-coherent) symmetric monoidal. This is done by developing a heuristic for constructing homotopies over large iterated smash products. This heuristic is also expressed as a formal theorem which, in essence, presents iterated smash products as retracts of homotopically simpler spaces.

Paper IV presents work on cellular homology in HoTT. We develop basic theory of CW complexes and prove a constructive analogue of the cellular approximation theorem, as well as a special case of the CW-approximation theorem which states that two different notions of n-connectedness are equivalent. The cellular approximation theorem is then used to prove functoriality of cellular homology, and the special case of the CW-approximation theorem is used to prove the Hurewicz theorem. We also verify that our homology theory satisfies the Eilenberg–Steenrod axioms.

Paper V uses the cohomology theory from Paper I in order to construct the Steenrod squares, a set of cohomology operations for mod 2 cohomology. We use (and generalise) a definition of these operations due to Brunerie (2017), but go much further and prove their characterising properties, e.g. the Cartan formula and the Adem relations. This is done by reducing most problems to a 'master theorem' which is a simple-to-state but difficult-to-prove Fubini-like statement concerning so-called unordered joins. We use the squares to complete an alternative computation of the fourth homotopy group of the 3-sphere suggested in Paper II.

Keywords: *homotopy type theory, constructive mathematics, formalisation.*

Stockholm 2025
http://urn.kb.se/resolve?urn=urn:nbn:se:su:diva-241553

ISBN 978-91-8107-196-2
ISBN 978-91-8107-197-9

Department of Mathematics

Stockholm University, 106 91 Stockholm

SYNTHETIC APPROACHES TO COHOMOLOGY, HOMOLOGY AND HOMOTOPY

Axel Ljungström

Stockholm
University

Synthetic approaches to cohomology, homology and homotopy

Axel Ljungström

ISBN print 978-91-8107-196-2
ISBN PDF 978-91-8107-197-9

Printed in Sweden by Universitetsservice US-AB, Stockholm 2025

Sammanfattning på svenska

Den här avhandlingen består av fem artiklar om syntetisk homotopiteori i homotopi-typteori (HoTT), ett formellt system som utvidgar Martin-Löfs typteori i den bemärkelsen att det även innefattar högre-induktiva typer samt univalens. Avhandlingen handlar i synnerhet om (ko)homologiteorier och kohomologioperationer men behandlar även andra (ofta relaterade) nyckelresultat från homotopiteori. Det är många av resultaten som presenteras här som inte enbart är viktiga från ett matematiskt perspektiv, utan som också är väsentliga eftersom de har *datorformaliserats*, d.v.s. formellt verifierats av ett datorprogram. En majoritet av satserna i den här avhandlingen har formaliserats i bevisassistenten `Cubical Agda`.

Artikel I beskriver en konstruktion av kohomologiringar i HoTT. För att åstadkomma detta bevisas att kopprodukten (eng. *cup product*) är associativ – ett till synes enkelt problem som har visat sig vara icke-trivialt i HoTT då naiva bevismetoder lätt fastnar i komplicerade koherensproblem. Artikeln innehåller även beräkningar av kohomologigrupper och kohomologiringar, Gysinföljden samt en diskussion rörande beräkningsmässiga aspekter av vår datorimplementering i `Cubical Agda`. Artikeln kan ses som en introduktion till resten av avhandlingen.

Artikel II presenterar en datorformalisering av Bruneries (2016) beräkning av den fjärde homotopigruppen av 3-sfären. Utöver detta beskrivs även en avsevärd förenkling av Bruneries bevis. Det här åstadkoms genom att introducera ett nytt sätt att beräkna det så kallade ”Brunerie-talet”, d.v.s. ordningen av homotopigruppen i fråga.

Artikel III är helt och hållet tillägnad beviset av det faktum att smashprodukter är symmetrisk-monoidala. Det här är en sats som vid första anblick ser enkel ut men som har visat sig vara förvånansvärt svårbevisad i HoTT. Här utvecklas en heuristik som förenklar konstruktionen av homotopier över godtyckligt itererade smashprodukter vilken senare används för att bevisa satsen i fråga. Heuristiken uttrycks också som en formell sats.

Artikel IV handlar om cellulär homologi i HoTT. Vi utvecklar grundläggande teori om CW-komplex och bevisar en konstruktiv version av satsen om cellulär uppskattning (eng. *cellular approximation theorem*) samt ett specialfall av satsen om CW-uppskattning (eng. *CW-approximation theorem*) vilken säger

att två olika definitioner av n-sammanhängighet (eng. *n-connectedness*) sammanfaller. Vi använder den förstnämnda satsen för att bevisa funktorialitet för cellulär homologi och den sistnämnda satsen för att bevisa Hurewicz sats som relaterar vissa homologigrupper och homotopigrupper. Vi verifierar även att vår homologiteori uppfyller Eilenberg-Steenrod-axiomen.

I Artikel V används kohmologiteorin från Artikel I för att konstruera de så kallade Steenrodkvadraterna, ett system av kohomologioperationer för mod 2-kohomologi. Den definition som används kommer från en kort text av Brunerie (2017) men den generaliseras och används för att bevisa de egenskaper som karakteriserar Steenrodkvadraterna: i synnerhet Cartanformeln och Ademrelationerna. Det här görs genom att reducera till en huvudsats, vilken är lätt att formulera men svår att bevisa, som uttrycker ett slags Fubini-egenskap hos så kallade oordnade sammansättningar (eng. *joins*). Vi använder slutligen Steenrodkvadraterna för att fylla i en lucka i en alternativ beräkning av den fjärde homotopigruppen av 3-sfären från Artikel II.

List of Papers

This thesis is based on five papers, numbered I–V, which are included in the following order.

I. Axel Ljungström and Anders Mörtberg. Computational Synthetic Cohomology Theory in Homotopy Type Theory. To appear in *Mathematical Structures in Computer Science*. 2024. arXiv: 2401.16336

II. Axel Ljungström and Anders Mörtberg. Formalising and Computing the Fourth Homotopy Group of the 3-Sphere in Cubical Agda. Preprint. 2024. arXiv: 2302.00151

III. Axel Ljungström. Symmetric monoidal smash products in homotopy type theory. *Mathematical Structures in Computer Science* (2024), pp. 1–23. DOI: 10.1017/S0960129524000318

IV. Axel Ljungström and Loïc Pujet. Cellular Methods in Homotopy Type Theory. Preprint. 2025. URL: https://aljungstrom.github.io/files/cellular2025.pdf

V. Axel Ljungström and David Wärn. The Steenrod squares via unordered joins. To appear at the *40th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2025. URL: https://aljungstrom.github.io/files/steenrod2025.pdf

Related but excluded papers The following papers have been produced during my PhD but are not included in this thesis.

EI. Guillaume Brunerie, Axel Ljungström and Anders Mörtberg. "Synthetic Integral Cohomology in Cubical Agda". *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*. Ed. by Florin Manea and Alex Simpson. Vol. 216. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 11:1–11:19. ISBN: 978-3-95977-218-1. DOI: 10.4230/LIPIcs.CSL.2022.11

EII. Thomas Lamiaux, Axel Ljungström and Anders Mörtberg. "Computing Cohomology Rings in Cubical Agda". *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs*. CPP 2023. Boston, MA, USA: Association for Computing Machinery, 2023, pp. 239–252. ISBN: 9798400700262. DOI: 10.1145/3573105.3575677

EIII. Axel Ljungström and Anders Mörtberg. "Formalizing $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$ and Computing a Brunerie Number in Cubical Agda ". *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. Los Alamitos, CA, USA: IEEE Computer Society, 2023, pp. 1–13. DOI: 10.1109/LICS56636.2023.10175833

EIV. Stefania Damato, Thorsten Altenkirch and Axel Ljungström. Formalising inductive and coinductive containers. Submitted. 2024. arX-iv: 2409.02603

Papers EI and EII are excluded due to their overlap with Paper I. The relation between Paper EI and Paper I is that the latter is a generalisation of the former (see the coming paragraphs for details). Like Paper I, Paper EII also concerns cohomology rings but has a strong emphasis on the computer formalisation of these. My contribution to this paper is the actual mathematics behind the computations of the cup product structure on the cohomology rings presented there, but these proofs had to be omitted due to space constraints and scope. Paper I contains these details instead, so from the point of view of my personal contribution, it would be superfluous to include both papers. Paper II is an extended (journal) version of Paper EIII, which is why the latter is excluded. Paper EIV is excluded because it does not relate very strongly to any of the other papers included in this thesis. I am also not a main author of this paper.

Relation to licentiate thesis The excluded papers EI and EII were included in my licentiate thesis and thus, since these papers are, respectively, closely related to Papers I and II, there is some overlap between my licentiate thesis and this thesis. In both cases, however, the version in this thesis constitutes a significant extension to that in my licentiate thesis. Paper I generalises Paper EI from integral cohomology to cohomology for arbitrary groups and contains several new cohomology computations. Paper II extends Paper EIII primarily by means of §6 which spells out the details of and generalises the methods behind one of the main novelties of Paper EIII, namely the simplified computation of the Brunerie number.

Author's contribution Paper I is primarily written by me. Most importantly, I am responsible for all mathematical results and the computer formalisation of the paper. Mörtberg supervised the project and was mainly involved in the benchmarking of our computer implementation as well as writing parts of the introduction and the section on related and future work.

For Paper II, the situation is similar to that for Paper I. I am the main author of the paper with Mörtberg acting as a supervisor of the project. As in Paper I, I am responsible for the mathematical results and the computer formalisation while Mörtberg has assisted in benchmarking as well as writing the introduction and conclusion.

Paper III is written solely by me.

Paper IV was written together with Pujet and it is difficult to take or give sole credit for any particular result. Almost the entire paper was conceived during discussions with Pujet and we would work on all mathematical results and the computer formalisation together in person (as we shared an office at the time, this is no exaggeration). There are two main theorems in the paper: the cellular approximation theorem and the Hurewicz approximation theorem. Pujet devised the first proof sketch of the former while I devised the first one of the latter. Nevertheless, in the end, these proofs were reworked and improved on together and have changed since their first renditions.

Paper V was written with Wärn. Most of the mathematics was developed at in-person meetings which would take place whenever Wärn was in Stockholm. It is especially difficult to estimate the contributions as the project was so extensive, taking roughly 2 years to finish; during this time there have been many different approaches taken that never made it into the final paper. Very often, Wärn would suggest a certain idea (e.g. the usefulness of unordered joins rather than smash products) and I would explore whether this idea is feasible by trying to devise the relevant computer formalised proofs. Nevertheless, this is a rather simplified picture: most theory was simply developed together on the blackboard.

Acknowledgements

First and foremost, I would like to thank Anders Mörtberg. I am very lucky to have had such a present supervisor throughout my doctoral studies. The time and energy he has devoted to me over these five years is, I think, more than any PhD student anywhere could expect.

I also wish to thank all of my colleagues for making our department such a nice workplace. I especially want to thank the logic group for many nice seminars and dinners.

I must also thank my friends and family who have allowed me to maintain some level of normality during my years as a PhD student. Because she told me not to, I have to include a special thanks to Claire Caron, my wife, for putting up with me and my never-ending deadlines.

Finally, I would like to thank everyone who has taken an interest in and supported my academic endeavours, be it collaborators or old teachers. In particular, I want to acknowledge Roussanka Loukanova, who, early on in my education, encouraged me to pursue a career in mathematics and computer science.

Contents

A general introduction to (homotopy) type theory

It is never easy to be wrong. For mathematicians, however, whose whole raison d'être is *being certain*, this prospect is perhaps especially terrifying. Although the papers produced by mathematicians undergo proof-readings and peer-reviews just like in any other field, it is impossible to guarantee that they are completely void of mistakes. In a field as fragmented as mathematics, where two leading experts in their respective subfields can be so far apart academically that they have little else than the weather to chat about in the lunchroom, mistakes have an even better chance of surviving, simply due to the (often) small number of mathematicians who possess the appropriate background to appreciate the material of a given research paper. This can cause errors in influential papers to go unnoticed for years.

A particularly well-known case of an influential paper containing a non-trivial mistake is Kapranov and Voevodsky's '∞-groupoids and homotopy types' [Mik91]. The paper presented a proof of a version of the famous *homotopy hypothesis*. It was not until seven years later that Simpson [Sim98] found a mistake in their proof (and not until yet another 15 years that Voevodsky had, to his own satisfaction, understood the mistake in the original paper [Voe14]). Unhappy with this situation, Voevodsky started to put a serious amount of thought into the prospect of using digital software to check the correctness of mathematical proofs. Such computer programs tend to rely on *type theories*; these are formal systems, some of which have been suggested as an alternative to the *set theory* that is currently the most widely accepted foundation of mathematics. The idea is simple: if we are able to express our mathematical proofs using type theory rather than set theory, we can much more readily ask our computers to verify their correctness. The version of type theory to be used would be the intensional type theory of Martin-Löf [Mar84] endowed with new principles (univalence and higher inductive types) making it suitable for reasoning about the field of *homotopy theory*. The resulting type theory, *homotopy type theory* (HoTT), was designed and studied by Voevodsky and, one must stress, *several* collaborators during a special year at the Institute for Advanced Study, resulting in *the HoTT book* [UF13], the first textbook on

the subject.

This thesis is my contribution to the ongoing work of translating old results into – and developing new results in – the setting of HoTT. It is primarily concerned with results concerning concepts like homotopy, cohomology and homology. While many of the results are well known from the point of view of classical mathematics, their proofs in HoTT often differ significantly. The results are also automatically generalised when expressed in HoTT, due to its rich family of (higher-categorical) models [Shu19]. One of the key motivations behind this work is, like it was for Voevodsky, to be able to digitally verify – to *computer formalise* – these results. Indeed, many of the results presented in this thesis have been computer formalised in the *proof assistant* software `Cubical Agda` [VMA21].

Structure of this chapter This chapter has four main sections and they are all intended for a relatively broad audience of mathematicians. The first three introduce (Martin-Löf) type theory, homotopy type theory and the `Cubical Agda` proof assistant. The expert can likely jump straight to §4. I wish to emphasise here that much of the material presented in the coming sections is also included in some of the papers (albeit in a more compact form).

We start off, in §1, with an introduction to type theory. This section is aimed towards a very general mathematical audience and can be skimmed or even skipped by the expert. For the expert, we summarise this section by saying that it introduces Martin-Löf type theory with universes à la Russell: in essence, a type theory closely resembling that of `Agda`.

In §2, we introduce homotopy type theory. Like the previous section, this section is also intended to be accessible to anyone with a reasonable level of mathematical maturity. It does, however, cater more to algebraic-topological intuitions than §1. The reader who is not helped by this can skip to §2.2. The section also touches on *cubical type theory*, a version of homotopy type theory which has been used when computer formalising the results in this thesis.

In §3, we briefly discuss the `Agda` proof assistant (i.e. verification software) and its extension `Cubical Agda`. The latter proof assistant implements cubical type theory and has been used to check many of the proofs in this thesis. It should be mentioned that Paper II also includes a brief introduction to `Cubical Agda` for those who wish to skip this section.

In §4, we briefly and very generally discuss the contributions of this thesis and the scientific context it was written in. We mention related work and what kind of future projects the material in this thesis could lead to.

1 Type theory

In order to explain what homotopy type theory (HoTT) is, we will need to say a few words about type theory in general. In this thesis, 'type theory' will always refer to a specific type theory, namely the intensional type theory of Martin-Löf (MLTT)[Mar84]. This is a *dependent* and *intuitionistic* type theory which means, respectively, that it allows for dependent types (roughly, indexed sets) and that it is constructive. Let us start off this brief introduction by explaining some of the basic ideas behind type theory and why anyone would ever want to use it for doing mathematics.

1.1 What is type theory?

Type theory is a formal system which, among other things, provides an alternative foundation of mathematics. Here is a one-liner describing it (which I forbid you to quote): *type theory is obtained by taking set theory and swapping the notation '$x \in X$' for '$x : X$'.* While this, from a formal perspective, is a vast oversimplification, it should provide the traditionally schooled mathematician with enough information about the role played by types in type theory required to parse most of the material presented in this thesis. We read the statement (or *judgement*) '$x : X$' as 'x is an element (or *term*) of *type* X'. This, of course, begs the question of what the difference between types and sets actually is. There are many (not necessarily mutually compatible) answers to this, and I will only attempt to provide one point of view here. Types are, unlike sets, not to be seen as a collection of already defined objects. Rather, I claim that a type is, in essence, a description of how to manipulate and interact with objects. In a way, a type is just a label containing user instructions which we slap onto our mathematical constructions. A standard example of this is the type of natural numbers. This is an *inductive type* which can be defined by the following two clauses (inspired by Peano arithmetic [Pea89]).

- $0 : \mathbb{N}$
- $\mathsf{suc} : \mathbb{N} \to \mathbb{N}$

Above, the first clause simply postulates the existence of an element of type $\mathbb{N}$ which we call 0. The second clause postulates the existence of a function suc which takes a natural number $x : \mathbb{N}$ and returns another natural number $\mathsf{suc}(x)$. Note that this presentation only describes how to generate new natural numbers – it never presupposes that we can gather them in one big collection.

A particular consequence of this description is that if you are presented with a natural number n, you can always be certain that it is either of the form 0 or $\mathsf{suc}(n')$ for some 'smaller' natural number n'. This is not only a theorem but a meta-mathematical fact which, in the version of type theory we have in mind here, holds *by definition*. More generally, we will always assume that

inductive types, such as $\mathbb{N}$, come equipped with the appropriate induction and computation rules. In this case, this means (somewhat simplified) that our type theory primitively supports induction on the natural numbers and that any function $f : \mathbb{N} \to A$ defined by induction actually *computes* according to its inductive description.

Never ask a term its type One important feature of type theory is that all mathematical objects are typed upon construction. This means that the question of whether a certain mathematical object belongs to a certain type becomes either tautological or nonsense – *to be a mathematical object* means, in type theory, *to be an element of a type*. If you present me with some $q : \mathbb{Q}$, you cannot then ask me whether $q : \mathbb{Z}$. Indeed, unless $\mathbb{Q}$ and $\mathbb{Z}$ were two names for the *exact* the same type, q cannot be an element of both![1] From a formal perspective, this is, of course, the exact same situation as in set theory where the statement '$q \in \mathbb{Q}$ and $q \in \mathbb{Z}$' is mere syntactic sugar for 'there is some $a \in \mathbb{Z}$ which the canonical embedding $\mathbb{Z} \hookrightarrow \mathbb{Q}$ takes to $q \in \mathbb{Q}$'. Type theory, by design, often forces us to make such syntactic sugar explicit. This is perhaps not always so important when communicating our proofs to humans, but when communicating our proofs to computers, who do not possess quite the same ability to read between the lines, this is crucial.

Often, by being strict about typing, type theory can help us make formal certain informal ideas we are used to from working in a set-theoretic foundation such as ZF (i.e. Zermelo–Fraenkel set theory – see e.g. [BM75]). Most mathematicians happily accept the difference in nature between the natural number 0 and the empty set, despite the fact that, in ZF, these objects are *exactly* the same. In a similar vein, we rarely see an algebraic topologist speak of a manifold of dimension n with $4 \in n$. We know that, although these constructions are entirely grammatical in the language of set theory, it is tacitly agreed among mathematicians that we are not really supposed to make them. The idea is, of course, that objects such as 0 and $\emptyset$ are incomparable in nature – one might even say that they are of different *type*. With any reasonable type-theoretic definitions of 0 and $\emptyset$, we could never ask whether $0 = \emptyset$ – because $0 : \mathbb{N}$ and $\emptyset : \mathsf{Set}$ are of different type, we can never compare them in this way.

Type theory and constructive mathematics The idea we have presented so far is that type theory is a system very similar to set theory but where (i) 'finitary' constructions of infinite objects are preferred to 'infinitary' (recall the type-theoretic definition of $\mathbb{N}$) and (ii) each mathematical object has a type, often forcing the mathematician to explicitly clarify informal ideas. There is, however, (at least) one more aspect of type theory worth mentioning: it is, by default, constructive. Let me mention the, to me, most interesting implications of this fact.

First and foremost, the constructive logic captured by type theory has a very

[1] OK, type theorist reading this: at least in MLTT it cannot.

rich class of models: one says that constructive logic is the *internal language of toposes* [MM94]. Now, if you do not know what a topos is, you can think of it as a kind of landscape in which we can do mathematics. All mathematicians know one topos very well, namely the topos of sets (the standard model of ZF). As it is a topos, any result proved constructively can be interpreted in the category of sets. However, unlike non-constructive results, we can also interpret them in *other* toposes – e.g. sheaves on a site, or any other permutation of serious-sounding words which may or may not excite the reader. This means that anything we prove using type theory will say something both about sets and something about, say, sheaves simultaneously. Many theorems proved in traditional mathematics are, to a significant extent, already constructive. A problem with using classical set-theoretic foundations is that they make it difficult to separate the non-constructive from the constructive parts of a proof. If we instead use a flexible constructive system like type theory, constructive reasoning is the default. When we prove a theorem, we *are* allowed to rely on non-constructive principles, but we have to make sure to explicitly state whenever we do so. This means that we can extract additional (computational) information even from non-constructive proofs by isolating those steps which do not rely on non-constructive principles.

A second implication of the use of a constructive system like type theory as a foundation is that proofs now can be interpreted as concrete, executable computer programs. The fundamental observation is that type theory, as a constructive language, really is just a programming language [Mar82; NPS90]. The guiding principle of constructive mathematics is that any time we make an existential statement, e.g. the timeless classic 'there are irrational numbers a and b such that a^b is rational', we have to provide a clear *witness* (or, in this case, *witnesses*) of this statement. A classical mathematician may provide a proof by contradiction: 'assume such a pair of irrational numbers does not exist... contradiction'. Such a proof never provides us with any concrete information about the actual values of a and b. In constructive mathematics, we cannot really prove this in any other way than by providing concrete examples (witnesses): 'consider $a = \sqrt{2}$ and $b = \log_2 9$'. Thus, a constructive proof provides more information than a non-constructive proof: it contains information on how to compute witnesses of the statement in question. Now, suppose you devised a constructive proof of something somewhat more eye-opening than the example covered here. It could be that there must be some integer solution a to whatever interesting equation you are trying to solve or that your favourite group is isomorphic to $\mathbb{Z}/n\mathbb{Z}$ for some $n : \mathbb{N}$. Constructive proofs of these statements should, at least in principle, provide enough information for us to deduce the exact values of a and n without having to do any further theorem proving. Indeed, a constructive proof of a theorem can be thought of as an *algorithm* for computing witnesses of the theorem. This idea of proofs as algorithms is especially central to type theory where the so-called *Curry–Howard correspondence* [How80] identifies, in a meaningful way, propositions and proofs with, respectively, types and elements of types. At the same time, types and elements of types have a natural interpretation in terms of, respectively, program

specifications and computer programs. We will return to these ideas in §3. For now, the main takeaway should be that a proof in type theory contains computational content which can be used to simplify or even completely remove the need for future theorem proving.

1.2 A crash course in type theory

So far, there has been a lot of talk about the general idea behind type theory but we have not actually engaged in it formally. In the following couple of pages, I will try to walk you through the very basics of the system. Before we start, I again wish to emphasise the versatile role played by types: we have already seen how a type can be used to capture one of our favourite sets, the natural numbers; in what follows, we will also see how types also can express logical propositions such as '$5 + 7 = 12$' and '2 is an even number'. As these propositions will be captured by types, proofs of these propositions will be precisely elements of these types. Thus, if we write $p : 5 + 7 = 12$, this means that p is a proof of the statement that $5 + 7 = 12$. In exactly the same way, we may write $n : \mathbb{N}$ to say that n is a natural number. Hence, in type theory, a proof is not a meta-mathematical entity but a mathematical object in the same right as, say, the natural number zero. The justification of this apparent conflation of concepts comes from the constructivist's idea of proofs as witnesses. In essence, even an element $n : \mathbb{N}$ is a proof – it is one out of ω many witnesses of the existence of natural numbers. This may take some time getting used to but should hopefully become clearer soon when we consider some concrete constructions in type theory.

Universes The first thing we will need is a universe: a special type $\mathcal{U}$ whose elements are types. Whenever we declare a new type, we write this as $A : \mathcal{U}$. Now, certainly type-theorists know better than to assume the existence of a 'type of all types'. After all, the original version of type theory was invented precisely in order to circumvent Russell's paradox [Rus03]. This can be solved by allowing for a whole hierarchy of universes, $\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \dots$, each an element of the other. This approach – often called *universes à la Russell* – is the one we adhere to in this thesis, although there are alternatives (e.g. *universes à la Tarski*) [Mar84, p. 48]. In practice, however, we will almost always be informal (as is common practice) and simply write $\mathcal{U}$ to denote an arbitrary universe. It is often sufficient to let $\mathcal{U} := \mathcal{U}_0$. We remark that the notation Type may be used instead of $\mathcal{U}$ in some of the papers featured in this thesis.

Functions Given two types $A, B : \mathcal{U}$, we will simply write $A \to B$ for the type of functions from A to B. Functions are introduced in the obvious way: we define a function $f : A \to B$ by defining, for every $a : A$, an element $f(a) : B$. When it comes to functions, there is one piece of notation you need

to know if you wish to be let into any secret gathering of type theorists: lambda notation. Lambdas are a way of introducing functions without naming them. This idea already exists in traditional mathematics where we write e.g. $x \mapsto x^2$ for the squaring function on, say, the natural numbers. Many type theorists would write this function as $\lambda\, x\,.\, x^2$ or, less traditionally, $\lambda\, x \to x^2$ or even $\lambda\,(x : \mathbb{N}) \to x^2$ to clarify the input type. This notation comes from lambda calculus [Chu32], a formal system closely related to type theory which is useful for capturing certain notions of computation.

Under the interpretation of types as logical propositions [How80], function types correspond to implication. In this case, a function $f : P \to Q$ between two types P and Q is thought of as a proof of the implication 'if P then Q'. We will return to this idea later when we speak of dependent functions.

Inductive types One important class of types is that of *inductive types*. These are types whose elements are inductively defined from a set of so-called *constructors*. We have already seen what is probably the most canonical example of an inductive type: the natural numbers $\mathbb{N}$. Recall, we defined these to be the type generated by two constructors, namely $0 : \mathbb{N}$ and $\mathsf{suc} : \mathbb{N} \to \mathbb{N}$. As touched upon earlier, we will always assume recursion/induction principles to be part of the definition of any inductive type. For $\mathbb{N}$, the recursion principle can be stated semi-formally by saying that any function $f : \mathbb{N} \to A$ is can be described (unambiguously) by providing the following data.

- $f(0) : A$
- for any $n : \mathbb{N}$, a term $f(n+1) : A$ whose definition may recursively refer to $f(n)$.

The natural numbers do not constitute the only example of an inductive type. We can, for instance, define the booleans, $\mathbb{B}$, and the unit type, $\mathbb{1}$, respectively to be the inductive types with constructors

$$\mathtt{true}, \mathtt{false} : \mathbb{B} \qquad\qquad \star_1 : \mathbb{1}.$$

The recursion principles for these types say, respectively, that a function $f : \mathbb{B} \to A$ is defined entirely by $f(\mathtt{true}) : A$ and $f(\mathtt{false}) : A$ and a function $g : \mathbb{1} \to A$ is defined entirely by $g(\star_1) : A$. We can also define the empty type $\bot$ to be the inductive type with no constructors. Its recursion principle says that there always exists a (unique) function $\epsilon : \bot \to A$, i.e. the empty function.

Dependent types, functions and sums So far, we have described a plain, non-dependent type theory. The type theory we are concerned with here, i.e. MLTT, is a *dependent* type theory. Dependent types are simply types parameterised by other types. This should not be a foreign concept to the classical mathematician who, perhaps without knowing, relies on such dependency for many crucial constructions. Now, a type being parameterised by another type

may, at least for the more algebraically inclined among us, invoke ideas of e.g. parameterised spaces, fibre products and so on. While these certainly are examples of dependent types, there are much more basic examples. Consider, for example, the n-dimensional reals, $\mathbb{R}^n$. This is a construction we often like to speak of with respect to an arbitrary natural number n. Really, $\mathbb{R}^{(-)}$ is a function which takes as input some natural number n and returns the set $\mathbb{R}^n$. In other words, it is (size-issues aside) a function $\mathbb{N} \to \mathsf{Set}$. In type theory, the idea is the same, the only difference being that we write $\mathcal{U}$ instead of Set; we say that $\mathbb{R}^{(-)} : \mathbb{N} \to \mathcal{U}$ is a *dependent type* (over $\mathbb{N}$). In general, a dependent type indexed by some type A is simply a function $B : A \to \mathcal{U}$.

Note that if we wish to speak of elements of a dependent type, we can only speak of elements of $B(a)$ given some fixed $a : A$. For instance, in the case of $\mathbb{R}^n$, it makes sense to write $(5, 7) : \mathbb{R}^2$ but not $(5, 7) : \mathbb{R}^{(-)}$. Some constructions are, however, universal enough to make sense for any input. For instance, regardless of the value of n, we always have a neutral element $0^n : \mathbb{R}^n$. In other words, $0^{(-)}$ is a function which for every $n : \mathbb{N}$ returns $0^n : \mathbb{R}^n$. This kind of function differs in one fundamental way from the functions discussed previously: the output type ($\mathbb{R}^n$) depends on the input element (n). We call such a function *dependent*. In general, given a dependent type $B : A \to \mathcal{U}$, we shall write $(a : A) \to B(a)$ for its associated dependent function type. Note that the non-dependent function type can be obtained as the special case when B is constant, i.e. when, for any $a : A$, we set $B(a) := B'$ for some (non-dependent) type B'. The reader should know that the traditional notation for the type $(a : A) \to B(a)$ is $\prod_{a:A} B(a)$ (and, in fact, dependent function types are commonly referred to as Π-types or dependent products). While I think the former notation is often clearer, the Π-notation could be useful for the reader who likes to think of types as sets. With this interpretation, a dependent type $B : A \to \mathcal{U}$ corresponds to a family of A-indexed sets, while the dependent function type $\prod_{a:A} B(a)$ is simply the product over this family.

Another crucial construction is that of dependent sums. Given a dependent type $B : A \to \mathcal{U}$, its dependent sum is a type which we denote by $(a : A) \times B(a)$, although traditionally (and sometimes in this thesis) it is also written $\sum_{a:A} B(a)$. An element of this type is a pair (a, b_a) where $a : A$ and $b_a : B(a)$. For instance, both $(2, (5, 7))$ and $(3, (5, 7, 12))$ are elements of $(n : \mathbb{N}) \times \mathbb{R}^n$. We use the notation fst and snd to denote the first and second projections, i.e. $\mathsf{fst}(a, b_a) = a$ and $\mathsf{snd}(a, b_a) = b_a$. Set-theoretically, we can view dependent sums as playing the role of indexed disjoint unions with $(a : A) \times B(a)$ corresponding to the A-indexed union $\bigsqcup_{a:A} B(a)$. Whenever B does not depend on A, we simply write $A \times B$ as this construction, by definition, is a binary product.

There is an elementary but important interplay between (dependent) functions and sums. In type theory, it is not uncommon to ask for a function $f : A \to (B \to C)$. Formally, this is a function taking an argument $a : A$ and returning another function $f(a) : B \to C$. In practice, however, we simply think of this as a function taking two arguments – one in A and one in

B – and returning something in C. Given $a : A$ and $b : B$, we *should* write $f(a)(b) : C$ for the output of f applied to first a and then b. In practice, however, it is often nicer to simply write $f(a, b)$ and think of f as a function of type $A \times B \to C$. The translation between these two representations of binary functions is called, in this direction, *uncurrying* and in the other direction *currying* (or, less commonly but perhaps more justly, had the name not been so long, *Schönfinkelisation* [Sch24]). We will use this trick in many places and without comment. Now, currying is not a concept reserved for non-dependent functions: thanks to the existence of dependent sums, we can also curry dependent functions. For the most general case, consider three types of increasing dependency $A : \mathcal{U}$, $B : A \to \mathcal{U}$ and $C : (a : A) \to B(a) \to \mathcal{U}$. First of all, we can uncurry C, viewing it instead as a family of types defined over the dependent sum over B. That is, $C : (a : A) \times B(a) \to \mathcal{U}$. Now, suppose we are given a dependent function $f : (a : A) \to (b : B(a)) \to C(a, b)$. Just like in the non-dependent case, we will write the application of f as $f(a, b)$ rather than $f(a)(b)$ which is motivated by the fact that, by uncurrying, we may view f as a dependent function $f : ((a, b) : (a : A) \times B(a)) \to C(a, b)$.

Finally, let us make a remark concerning the interplay between dependent types and inductive types. It was previously pointed out that all inductive types are thought of as automatically coming equipped with recursion rules. For instance, the recursion rule for $\mathbb{N}$ describes how functions of type $\mathbb{N} \to A$ are constructed. We never saw, however, how to construct a dependent function $f : (n : \mathbb{N}) \to B(n)$. For natural numbers, this is the obvious dependent analogue of the recursion rule: it suffices to specify $f(0)$ and provide, for any $n : \mathbb{N}$, a function $B(n) \to B(n + 1)$. Every recursion rule has such a dependent analogue which we call an *induction rule*. Of course, recursion rules are mere special cases of these more general induction rules. Just like with recursion, we will assume all inductive types to satisfy their obvious induction rules.

Dependent types as predicates Until now, our explanation of dependent types has primarily catered to set-theoretic intuitions. However, dependent types also play a crucial role in expressing logical predicates. Say, for instance, that we wish to encode the predicate $\mathsf{isEven}(n)$ expressing that a given $n : \mathbb{N}$ is even. Clearly, this predicate depends (both syntactically and semantically) on the input variable $n : \mathbb{N}$. Thus, this predicate must be encoded as a dependent type $\mathsf{isEven} : \mathbb{N} \to \mathcal{U}$. Using the recursion rule for $\mathbb{N}$, we can actually define a version of this predicate already:

$$\begin{aligned} \mathsf{isEven}(0) &:= \mathbb{1} \\ \mathsf{isEven}(1) &:= \bot \\ \mathsf{isEven}(2 + n) &:= \mathsf{isEven}(n). \end{aligned}$$

Recall that if we view types as logical propositions, elements of types correspond to proofs of these propositions. For instance, an element $p : \mathsf{isEven}(48)$ is a proof of the fact that 48 is even. Under this interpretation, the type of dependent functions plays the role of proofs of universally quantified statements. In the

case of isEven, a dependent function $(n : \mathbb{N}) \to \mathsf{isEven}(n)$ would, if it existed, constitute a proof of the statement that every natural number is even. We can now also build up more complex statements. For instance, we can encode a proposition like 'for every natural number n, if n is even, then so is $2 + n$' by the type

$$(n : \mathbb{N}) \to \mathsf{isEven}(n) \to \mathsf{isEven}(2 + n).$$

To prove the statement, we need to define $p(n, q) : \mathsf{isEven}(2+n)$ for every $n : \mathbb{N}$ and $q : \mathsf{isEven}(n)$. Actually, we can complete this proof already now. Since we *defined* $\mathsf{isEven}(2 + n)$ to be $\mathsf{isEven}(n)$, we are really trying to construct an element $p(n, q) : \mathsf{isEven}(n)$. Thus, we may simply define $p(n, q) := q$, and we have thereby constructed our first proof in type theory.

Just like dependent function types correspond to universal quantifiers, dependent sums correspond to existential quantifiers. To continue on our example of the isEven predicate, suppose we were interested in providing a proof of the statement that 'there exists an even number'. In type theory, this amounts to providing two pieces of data: a natural number n and a proof $p : \mathsf{isEven}(n)$. Hence, this existential statement can be captured by the dependent sum $(n : \mathbb{N}) \times \mathsf{isEven}(n)$ whose objects are precisely pairs of these two pieces of data. Note that with this interpretation of the existential quantifier, there may be different ways of proving the same statement – for instance, both $(0, \star_1)$ and $(2, \star_1)$ are elements of $(n : \mathbb{N}) \times \mathsf{isEven}(n)$. This is a quirk which becomes entirely obvious when we think of existence proofs in terms of witnesses.

Identity types The type theory we have described so far looks quite a bit like a system of first order logic, and indeed this intuition is not far from the truth. However, if we wish for our type theory to be able to play this role, there is one construction missing: identity types. With the type theory we have so far, we are not able to express even the most basic theorems of arithmetic such as, say, the commutativity of natural number addition. Perhaps surprisingly, the treatment of identity types is one of the key differences between set theory and type theory. We wish to define, for any type A, a dependent type $\_ = \_ : A \to A \to \mathcal{U}$ such that, for any $x, y : A$, an element $p : x = y$ constitutes a proof of the fact that x and y are equal. One naïve definition would look something like

$$(x = y) := \begin{cases} \mathbb{1} & \text{if } x \text{ and } y \text{ are exactly the same object, syntactically} \\ \bot & \text{otherwise} \end{cases}$$

but this does not look like a definition in type theory. However, the right idea is there: we wish to express that any two general elements $x, y : A$ can only be declared equal if x and y are *literally* the same element. In other words, the only equality we should be able to infer automatically is $x = x$. The way we encode this in type theory is by an ingenious use of inductive types. This idea, due to Martin-Löf [Mar84], is really what distinguishes modern intuitionistic

type theory (i.e. MLTT) from other alternatives. For any $x : A$, we define the identity type $x = \_ : A \to \mathcal{U}$ to be the family of inductive types with one constructor:

- $\mathsf{refl}_x : x = x$.

This perhaps strange-looking construction is an *indexed inductive* type. These are essentially dependent analogues of inductive types but we will not need to discuss how to formally make sense of them here. This definition of identity may take some time getting used to. The intuition should be that it is valid to form the type $x = y$ for any $x, y : A$, but the only time you can actually construct an element of it (i.e. prove the equality of x and y) is when x and y are exactly the same object.

Like any inductive type, the identity type comes with an induction rule. It is so important that we will give it a name: the *J-rule*, or simply *J*.

Definition 1 (J)**.** *Fix $x : A$ and let $B : (y : A) \times (x = y) \to \mathcal{U}$ be a dependent type equipped with an element $b_0 : B(x, \mathsf{refl}_x)$. In this case, there is a dependent function $f : (y : A) \times (p : x = y) \to B(y, p)$. Furthermore, f satisfies the equality $f(x, \mathsf{refl}_x) = b_0$ and it holds strictly (i.e. by refl_{b_0}).*

Informally, the J-rule says that 'whenever we have a proof $p : x = y$, we may always replace y by x and p by refl_x'. There are some caveats here: for instance, y has to be a fresh variable, but let us postpone these discussions for later. Let us instead look at some examples of how the J-rule can be used to prove two elementary results about the identity type. We will do this with a decreasing level of formality, with the second proof looking much more like one written by the working type theorist.

Lemma 2 (Symmetry)**.** *For any $x, y : A$, if $x = y$, then $y = x$.*

A rather formal proof. The goal is to construct, for any $x, y : A$, a function $g : x = y \to y = x$. To this end, let us apply Definition 1 with $B(y, p) := (y = x)$. We have an element of $B(x, \mathsf{refl}_x)$, namely refl_x. We thus obtain a dependent function $f : (y : Y) \times (p : x = y) \to y = x$. We set $g(p) := f(y, p)$ and we are done. □

Lemma 3 (Transitivity)**.** *Let $x, y, z : A$. If $x = y$ and $y = z$, then $x = z$.*

A not so formal but close-to-practice proof. Let $p : x = y$ and $q : y = z$. We wish to produce an element $r : x = z$. By applying J to p, we may replace every y by x. After doing so, we have $q : x = z$, and thus we may simply set $r := q$ and we are done. □

As a sanity check, we should probably verify one of the defining features of any reasonable definition of equality: it is preserved by function application. This construction is important, so we will give it a name.

Lemma 4 (Function application)**.** *For any $f : A \to B$ and $x, y : A$, there is a function $\mathsf{ap}_f : x = y \to f(x) = f(y)$.*

Proof/Construction. We define $\mathsf{ap}_f(p)$ for $p : x = y$. By J, it is enough to define it when x is y and p is refl_x. In this case, we need to define $\mathsf{ap}_f(\mathsf{refl}_x) : f(x) = f(x)$ which we do by $\mathsf{ap}_f(\mathsf{refl}_x) := \mathsf{refl}_{f(x)}$ □

It is crucial when applying J to some identity proof $p : x = y$ that the statement we are trying to prove is defined for *any* y. In other words, y must be arbitrary and, in particular, cannot depend on x. We know, thanks to Hofmann and Streicher [HS98], that this would allow us to prove statements which are known to be independent (and even undesirable if we later wish to extend our type theory to obtain homotopy type theory). Consider, for instance, the following non-theorem and its accompanying non-proof.

Non-Theorem 1. *For any $p : x = x$, we have $p = \mathsf{refl}_x$.*

'Proof'. We apply J to p, replacing x by x and p by refl_x. We are left to prove $\mathsf{refl}_x = \mathsf{refl}_x$ which holds by $\mathsf{refl}_{\mathsf{refl}_x}$. △

The issue with the 'proof' above is that J does not apply to the identity type $x = x$ as the endpoint depends on x (indeed, it is literally x). For J to apply, we should be able to replace this endpoint x with an arbitrary element $y : A$. However, if we do this, the statement we are trying to prove is that $p = \mathsf{refl}_x$ for any $p : x = y$. This is not well-typed as the types of p and refl_x now differ (one mentions y and one does not). The significance of our failure to prove this statement will become much clearer soon when we discuss homotopy type theory.

We should briefly mention that there is another notion of identity: *strict* or *definitional* equality. So far, we have often seen statements like $f(x) := x^2$. I have assumed, without comment, that the reader parses this as '$f(x)$ is defined to be x^2'. When we make such a definition, $f(x)$ and x^2 become *definitionally* or *strictly* equal – from the point of view of the type theory, they are syntactically the exact same object. This notation will be used whenever two elements are syntactically the same (even if we did not explicitly define them as such). That is to say, we will see the notation $x := y$ used if $\mathsf{refl}_x : x = y$. Note that this notion of equality is meta-mathematical rather than a construction within the type theory.

Now, back to the actual identity type of our type theory: another important construction related to identity types is the *transport* function.

Definition 5 (Transport)**.** *Let $x, y : A$ and let $B : A \to \mathcal{U}$ be a dependent type. For any $p : x = y$, there is a function*

$$\mathsf{transport}^B(p, -) : B(x) \to B(y)$$

defined by applying J to p: that is, we define $\mathsf{transport}^B(\mathsf{refl}_x, -) : B(x) \to B(x)$ *by* $\mathsf{transport}^B(\mathsf{refl}_x, b) := b$.

On the one hand, thinking of types as propositions, the transport function simply captures Leibniz's law: if x and y are equal, then any property that holds for x also holds for y. On the other hand, the transport function allows us to define a notion of dependent identity.

Definition 6 (Dependent identity)**.** *Let $x, y : A$, $p : x = y$, $B : A \to \mathcal{U}$, $b_x : B(x)$ and $b_y : B(y)$. We define the type of dependent identities between b_x and b_y over p, denoted $b_x =_p^B b_y$, by*

$$(b_x =_p^B b_y) := (\mathsf{transport}^B(p, b_x) = b_y).$$

Whenever p is refl above, the dependent identity type above coincides precisely with the usual one.

The dependent identity type may seem daunting at first, but from a formal point of view, it is crucial. Consider something as simple as the axiom of graded commutativity for a graded ring $R_\bullet$. Informally, we would like to capture this by saying that for any $x : R_i$ and $y : R_j$, we have that $xy = (-1)^{ij}yx$. However, from a strictly formal perspective, the latter expression is not well-typed. Indeed, we have $xy : R_{i+j}$ while $(-1)^{ij}yx : R_{j+i}$ and the types appearing here are only equal up to an identification $c : i + j = j + i$. Thus, if we were to be entirely strict, we should write '$xy =_c^R (-1)^{ij}yx$'. Of course, when doing informal mathematics, we would never be this rigorous and simply use the usual identity type, abusing notation. It should be pointed out, however, that we are not, at this point, quite justified in accepting this convention. Indeed, there could be another identity proof $c' : i + j = j + i$, so which type do we mean when we informally write '$xy = (-1)^{ij}yx$'? Do we mean '$xy =_c^R (-1)^{ij}yx$' or do we mean '$xy =_{c'}^R (-1)^{ij}yx$'? For now, let us just accept it – we will touch upon these kinds of issues when we discuss homotopy type theory in §2.

There are also situations where the notion of dependent identity is more than just a formal necessity. For instance, it plays a crucial role in the description of identity types of dependent sums. To illustrate this, let $B : A \to \mathcal{U}$ be a dependent type and let $(x, b_x), (y, b_y) : (a : A) \times B(a)$ be pairs in its dependent sum. It is natural to ask what it means to prove these two pairs equal. A naive answer would be to say that $(x, b_x) = (y, b_y)$ if $x = y$ and $b_x = b_y$. However, the latter equality is not well-typed. Indeed, $b_x : B(x)$ and $b_y : B(y)$ belong to different types which are only equal up to an identity proof $p : x = y$. What we should say is that $(x, b_x) = (y, b_y)$ if there is an identity proof $p : x = y$ and a dependent identity $b_x =_p^B b_y$.

Since we covered the identity types of (dependent) sums, it only makes sense to say a few words about the identity types of (dependent) functions. This is a big topic which, you guessed it, will be better explained when we discuss homotopy type theory (at this point, the reader will hopefully have started anticipating

Type theory	Set theory	Logic
$A : \mathcal{U}$	A is a set	A is a proposition
$x : A$	$x \in A$	x is a proof of A
$A \to B$	$A \to B$	A implies B
$B : A \to \mathcal{U}$	B_a is a family of sets	B is a predicate
$(a : A) \to B(a)$	$\prod_{a \in A} B_a$	$\forall\, a\,.\, B(a)$
$(a : A) \times B(a)$	$\bigsqcup_{a \in A} B_a$	$\exists\, a\,.\, B(a)$
$x = y$	$x = y$	$x = y$

Table 1: Interpretations of type theory

that the identity types may play an important role further on). The obvious notion of equality for (dependent) functions is called *function extensionality*.

Definition 7 (Function extensionality)**.** *The principle of function extensionality says that two (possibly dependent) functions $f, g : (a : A) \to B(a)$ are equal if for every $(a : A)$, we have that $f(a) = g(a)$.*

We cannot prove function extensionality without modifying our type theory (see e.g. [Str93, §3.7]). Fortunately, when we later add the univalence axiom to get *homotopy* type theory, we will actually be able to prove it. If we do not wish to go that far, simply assuming function extensionality as an axiom is completely fine.

On a somewhat related note, we should also mention the dependent analogue of function application (Lemma 4). The original construction was given for plain functions but can now also be given for dependent functions. Given $f : (a : A) \to B(a)$ and $x, y : A$ we can define $\mathsf{ap\text{-}d}_f : (p : x = y) \to f(x) =_p^{B \circ f} f(y)$. Just like regular ap, it is defined by a simple application of the J-rule.

Over and out With these definitions, the reader should, at least in principle, have all they need to go out and do type theory. The system we have defined here can be used to provide a foundation of constructive mathematics. If we want more power, we may need to spice things up with a few extra axioms and/or constructions. For instance, we can add choice/LEM to get classical mathematics, but there are also other alternatives – one of the hottest ones will be covered in the following section. Before we get there, let us summarise the constructions we have seen so far and what they (roughly) translate to in set theory and logic. If you know a bit of type theory, you may have seen it 100 times before and, if you know even a bit more than that, you may have typed it up equally many times: I present to you, Table 1.

2 Homotopy type theory

If the reader feels confused after reading the above section on identity types, they need not worry: identity types in MLTT continue to confuse (and intrigue) even the most seasoned logicians today, years after their invention. In traditional mathematics, as pointed out earlier, identity proofs are meta-mathematical objects which we never have to manipulate directly (proof theorists may ignore this sentence). By allowing proofs to live in the same world as 'ordinary' mathematical objects (such as sets, numbers, etc.), we run the risk of these bringing with them strange artefacts. For instance, let us consider a simple task like defining the pre-image of a map $f : A \to B$ over some element $b : B$. In type theory, we could define it to be the dependent sum $f^{-1}(b) := (a : A) \times (f(a) = b)$. We would like to treat the elements of this type as (a 'subset' of the) elements of A, but we cannot immediately do so: its elements are pairs (a, p) where $a : A$ and $p : f(a) = b$. Suppose we have two different proofs $p, q : f(a) = b$ – which of the pairs (a, p) and (a, q) should we use to represent the element a? This definition really only makes sense if we can prove that $(a, p) = (a, q)$. Now, we could certainly try: the first components are equal and we prove this by refl_a. We now need to show that $p = q$ – how do we do that? We would need the following principle to hold for the type B.

Definition 8 (UIP)**.** *A type A satisfies the principle of uniqueness of identity proofs (UIP) if for any $x, y : A$ and $p, q : x = y$, we have that $p = q$.*

When we speak of the *general UIP*, we will mean the principle that UIP holds for all types. If the general UIP were to hold, our definition of 'subtypes' such as $f^{-1}(b)$ would be unproblematic. Thus, the general UIP appears to be crucial to the development of ordinary mathematics and, for a long time, type theorists were stuck trying to figure out its status [Str93; Coq92]. In the 'standard model' of type theory – the set model (see e.g. [Hof97]) – where types are interpreted as sets, the general UIP certainly holds. However...

Theorem 9 (Hoffman and Streicher [HS98])**.** *The general UIP is independent of type theory.*

The proof of Theorem 9 proceeds by considering a model of type theory in *groupoids* rather than sets, i.e. in categories where every morphism is invertible. The reader who is not impressed by category theory may equivalently think of groupoids as groups with a *partially defined* multiplication. The intuition behind this model is, *very* simplified, that elements x and y of a type can be taken to correspond to objects $[\![x]\!]$ and $[\![y]\!]$ of a suitable category, while an identity $p : x = y$ is interpreted as a morphism $[\![p]\!] \in \mathsf{Hom}([\![x]\!], [\![y]\!])$. In this model, these Hom-sets are non-trivial, thereby refuting UIP. So, we seem to be forced to make a decision: we either restrict the models of our type theory by adding the general UIP as an axiom or we simply bite the bullet and accept that 'subtypes' like the pre-image of a function are out of reach.

Well, if I could decide... It turns out that we actually are able to get quite far without the general UIP – the principle still holds for many types of interest, namely those with *decidable equality*. Let us spell out what this means. We say that a type A is *decidable* if there is an element of type $A + \neg A$ where '$+$' denotes the coproduct (roughly 'disjoint union') and $\neg A := A \to \bot$. The fact that this statement does not hold for all types may surprise you but, remember, type theory is a constructive language and this statement is essentially the law of excluded middle. We say that a type A has *decidable equality* if for any $x, y : A$, the identity type $x = y$ is decidable.

Theorem 10 (Hedberg [Hed98])**.** *UIP holds for any type with decidable equality.*

Fortunately for us, many of the types that we are interested in have decidable equality: the empty type, the unit type, the natural numbers, the rationals, and so on. As type theorists, we often think of types satisfying UIP as sets (as opposed to thinking of all types as such). These types are, just like sets, 'discrete' – more on this soon. Nonetheless, the fact that many types can be shown to satisfy UIP means that our type theory is still able to capture much of classical reasoning.

So, how are we to think of types which do not satisfy UIP? Sure, we can think of them as groupoids according to the groupoid model, but this does not quite capture the whole story: while the groupoid model captures the independence of UIP, it validates 'higher' instances of UIP. That is, it validates, for instance, the statement that for all $x, y : A$, the identity type $x = y$ satisfies UIP. This is, however, not provable either: for a counter model, simply consider 2-groupoids instead of 1-groupoids. We can continue to play this game ad infinitum, so let us kill ω birds with one stone and simply say that all higher instances of general UIP fail in the model of ∞-*groupoids* [BG11]. These are ∞-categories where all morphisms, including the higher ones, are invertible. For the uninitiated, ∞-groupoids are widely used in modern algebraic topology and homotopy theory as models of homotopy types (i.e. spaces up to weak homotopy equivalence). We will often simply refer to them as *spaces*. This interpretation of type theory is rather remarkable – just by giving up UIP, we obtain a type theory with models not only in *sets* but in this far richer class of mathematical objects. The fact that UIP holds for those types which we are used to thinking of as sets makes sense in this setting: here, sets are nothing but discrete spaces. In fact, it is standard terminology in (homotopy) type theory to refer to types satisfying UIP as *sets*.

2.1 Reinterpreting types as spaces

HoTT is obtained from type theory by adding so-called higher inductive types [LS20] and univalence [Voe10] to type theory. These concepts are far more intuitive if we have a good grasp of how type theory is interpreted in spaces, so let us devote some time to explaining this. Let us change the order

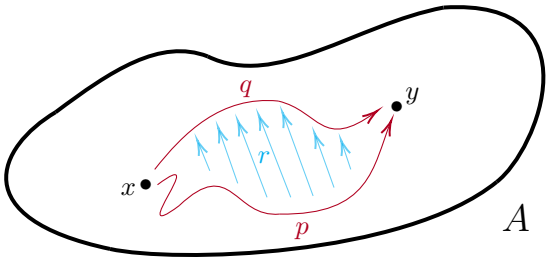


Figure 1: The type A as a space with points, paths and a homotopy

of presentation from §1.2 and start with plain types, function types and identity types. Dependent types and their sums and functions will be introduced after this. In order to understand these, we will need to use a bit more homotopy-theoretic jargon, so we will save them for last in order to avoid scaring anyone away immediately.

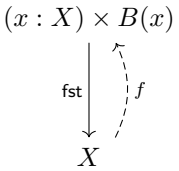
Types, functions and identity types When modelling types as spaces, they are not plain topological spaces but rather (and this is somewhat simplified) equivalence classes of these. Nevertheless, for intuition, we will think of each type $X : \mathcal{U}$ as a concrete (nice) topological space. With this interpretation, an element of type $x : X$ corresponds to a point in the space X. From now on, we will often call elements of types *points*. A function $f : X \to Y$ should be understood as a continuous function between spaces. In other words, *all functions are continuous* (this is not magic – you are simply working in a category where these are all the maps that exist).

The interpretation of identity types is where things really get interesting. Given two points $x, y : A$, an identity proof $p : x = y$ corresponds to a (continuous) path from x to y in A. The reason this idea is so powerful is that we can iterate it in order to provide an interpretation of 'higher' identity proofs: these are simply 'higher paths'. That is, if we interpret two identity proofs $p, q : x = y$ as paths, then an identity proof $r : p = q$ is interpreted as a 'path of paths', i.e. a homotopy of paths. This is illustrated in Figure 1. Naturally, we can keep going to get an interpretation of higher paths as higher homotopies and, suddenly, our type theory has turned into homotopy theory. By function extensionality, this interpretation also means that an equality $f = g$ really is a homotopy of functions.

When working in HoTT, the idea of identity proofs as paths is so rooted that it has taken careful proof reading to eliminate all premature instances of the word 'path' in the preceding sections. Finally, I can relax: from this point on, we will consistently refer to identity proofs as *paths*. We can now interpret the logical laws of the identity paths proved in §1.2 as path operations. The law of reflexivity, which is witnessed by $\mathsf{refl}_x : x = x$, is simply to be interpreted as

the constant path never leaving x. The law of symmetry, i.e. Lemma 2, is path reversal. Given a path $p : x = y$, we will write $p^{-1} : y = x$ for its inverse path. Finally, the law of transitivity, i.e. Lemma 3, is simply path composition. Given paths $p : x = y$ and $q : y = z$, we will write $p \cdot q : x = z$ for their composition. We can also interpret the J-rule in terms of paths, but let us first revisit the remaining concepts from type theory.

Dependent types and their functions and sums If types are spaces, one may reasonably think of dependent types as parameterised spaces or *fibrations*. Let $X : \mathcal{U}$ be a type and let $B : X \to \mathcal{U}$ be a dependent type. If we interpret B as a fibration, we may interpret $(x : X) \times B(x)$ as its *total space*. This interpretation is justified: $(x : X) \times B(x)$ consists of pairs of points $x : X$ in the base space of the fibration together with a point in the fibre $B(x)$. On a similar note, we interpret $(x : X) \to B(x)$ as the *space of sections* of the fibration. The reason for this interpretation is that a dependent function $f : (x : X) \to B(x)$ equally well may be described as a function $f : (x : X) \to (x : X) \times B(x)$ such that, for all $x : X$, we have that $\mathsf{fst}(f(x)) = x$ as illustrated in the following perhaps familiar diagram.



Path induction One of the perhaps most mysterious rules for newcomers to type theory is the J-rule (Definition 1). The principle tells us that sections of fibrations $B : (x : X) \times (x_0 = x) \to \mathcal{U}$ are uniquely determined by a choice of point in the fibre $B(x_0, \mathsf{refl}_{x_0})$. This principle can be generalised. To this end, say that a type X is *contractible* if it is pointed, i.e. inhabited by some $\star_X : X$, and equipped with a section $(x : X) \to \star_X = x$, i.e a proof that $\star_X$ is the unique point. Now, you hopefully find it reasonable that for an arbitrary contractible type X and fibration $B : X \to \mathcal{U}$, a section $(x : X) \to B(x)$ is uniquely described by the choice of a point in $B(\star_X)$. For any type X and $x_0 : X$, the total space $(x : X) \times (x_0 = x)$ should be contractible, and thus we have reduced the J-rule, or *path induction* as we will now start calling it, to a slogan everyone remembers from school: *the total space of the path space fibration is contractible*. This is rather remarkable – even though the original formulation of J was grounded purely in logico-mathematical considerations, it turns out to perfectly match one of the founding principles of homotopy theory. This concludes the homotopical interpretation of type theory. Let us

summarise, in Table 2, our new interpretation of types as spaces in an updated version of *the table*.

Type theory	Homotopy theory	Set theory	Logic
$A : \mathcal{U}$	A is a space	A is a set	A is a proposition
$x : A$	x is a point in A	$x \in A$	x is a proof of A
$A \to B$	$A \to B$ (cont.)	$A \to B$	A implies B
$B : A \to \mathcal{U}$	B is a fibration	B_a is a set family	B is a predicate
$(a : A) \to B(a)$	Sections of B	$\prod_{a \in A} B_a$	$\forall\, a\,.\, B(a)$
$(a : A) \times B(a)$	Total space of B	$\bigsqcup_{a \in A} B_a$	$\exists\, a\,.\, B(a)$
$x = y$	Path space $P(x, y)$	$x = y$	$x = y$

Table 2: Interpretations of type theory, V2

2.2 Getting HoTT

Although we have provided an interpretation of types as spaces, we have not quite described how spaces are manifested within type theory. That is, we have not really described how to *do homotopy theory* in type theory. The truth is that we are not quite there yet – we will need some additional machinery in order to internalise notions from the homotopical model. This additional machinery is what extends our type theory to make it *homotopy* type theory (or, recall, HoTT). We will add two things to our type theory: higher inductive types and univalence.

Higher inductive types One of the key ways of introducing new types is by means of *inductive types*. However, as pointed out earlier, types formed in this way often (but not always, as we know is the case for identity types) correspond to sets. As inductive types are defined in terms of their points, the homotopical model will often see them as discrete spaces. Nevertheless, if we wish to be able to do homotopy theory in our type theory, we better be able to capture other spaces than discrete ones – at the very least, we would need to be able to capture basic cell complexes. To this end, we introduce *higher inductive types* (HITs) [LS20]. HITs are obtained by letting inductive types allow not only for point constructors (such as $0 : \mathbb{N}$ and $\mathsf{suc} : \mathbb{N} \to \mathbb{N}$) but also for *path constructors* which, as the name suggests, allow us to specify paths between, for instance, the points introduced by the point constructors. The canonical example of a HIT is probably the circle, $\mathbb{S}^1$, so let us define it. We define it to be the (higher) inductive type with the following constructors.

- $\mathsf{base} : \mathbb{S}^1$
- $\mathsf{loop} : \mathsf{base} = \mathsf{base}$

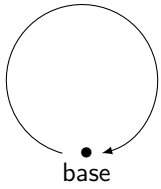


Figure 2: $\mathbb{S}^1$ as a HIT

This definition may look strange at first, but all it says is that $\mathbb{S}^1$ the space generated by one point $\mathsf{base} : \mathbb{S}^1$ and a path (or 1-cell) loop identifying this point with itself. In Figure 2, you will find a picture of the situation. Indeed, this idea corresponds well to the definition of the so-called minimal simplicial circle in classical homotopy theory [Lod92]. However, unlike the traditional minimal simplicial circle which often is rather restrictive (it is not fibrant), the implementation of $\mathbb{S}^1$ as a HIT provides a perfectly workable representation of the circle. The significance of this is that type theory often allows us to work directly with (homotopy types of) spaces in their most naïve form – this is certainly a simplifying feature.

Like plain inductive types, HITs come with recursion and induction principles. Let us start by investigating the recursion principle for $\mathbb{S}^1$. As functions are to be interpreted as continuous functions between spaces, we simply need to understand what data is encoded by a continuous map out of $\mathbb{S}^1$. Viewing $\mathbb{S}^1$ as a cell complex, the answer to this question is simple: a map $f : \mathbb{S}^1 \to X$ is described by a point $x : X$ (the image of base, the 0-cell) and a loop $p : x = x$ (the image of loop, the 1-cell). That is, given this data, the recursion principle of $\mathbb{S}^1$ tells us that we may define $f : \mathbb{S}^1 \to X$ by

$$\begin{aligned} f(\mathsf{base}) &:= x \\ \mathsf{ap}_f(\mathsf{loop}) &:= p \end{aligned}$$

where ap_f denotes function application – recall Lemma 4. Let us define a concrete function, the degree 2 function on $\mathbb{S}^1$, to showcase the recursion principle. In this case, the recursion principle simply states that the following definition is valid.

$$\begin{aligned} 2 \cdot (-) &: \mathbb{S}^1 \to \mathbb{S}^1 \\ 2 \cdot \mathsf{base} &:= \mathsf{base} \\ \mathsf{ap}_{2\cdot(-)}(\mathsf{loop}) &:= \mathsf{loop} \cdot \mathsf{loop} \end{aligned}$$

The induction principle is no different but, to state it, we require the dependent identity type – which will now be called the dependent *path* type – from Definition 6. In order to define a dependent function $g : (x : \mathbb{S}^1) \to B(x)$, we need to provide, just like before, a point $x : B(\mathsf{base})$ and a *dependent* path

$p : x =^{B}_{\mathsf{loop}} x$. We write

$$g(\mathsf{base}) := x$$
$$\mathsf{ap\text{-}d}_g(\mathsf{loop}) := p.$$

Note that the only difference from $\mathbb{S}^1$-recursion is that we have to use the dependent analogue of ap.

HITs can be used to express more than just typical cell complexes. They can, among other things, be used to capture set quotients. For instance, we can define $\mathbb{Z}/2\mathbb{Z}$ to be the HIT generated by

- $[-] : \mathbb{N} \to \mathbb{Z}/2\mathbb{Z}$
- $\mathsf{quot} : (n : \mathbb{N}) \to [n] = [2 + n]$.

In general, this naïve definition of set quotients does not work – it may turn out that the resulting type is no longer a set (i.e. no longer satisfies UIP). To remedy this, we can simply add another (higher) path constructor guaranteeing that the HIT is a set. A way of doing this more generally is by defining an operation called the *set truncation* $\|-\|_0 : \mathcal{U} \to \mathcal{U}$ which takes a type and forces it to be a set. For any A, this type can be defined as the HIT with the following constructors.

- $|-| : A \to \|A\|_0$
- $\mathsf{squash} : (x, y : \|A\|_0) \times (p, q : x = y) \to \mathsf{ap}_{|-|}(p) = \mathsf{ap}_{|-|}(q)$

Notice that this is not just a HIT – it is a HIT with a *higher* and *recursive* path constructor. We allow for such HITs too.

To showcase why this construction is important, consider defining $A/\!\sim$ for some A with an equivalence relation $\sim$ on it. One naïve definition would mimic that of $\mathbb{Z}/2\mathbb{Z}$: define the type $\widehat{A/\!\sim}$ to be the HIT with the following generators.

- $[-] : A \to \widehat{A/\!\sim}$
- $\mathsf{quot} : (x, y : A) \times (x \sim y) \to [x] = [y]$

In general, this type does not capture $A/\!\sim$. Consider, for instance, when A is the trivial type and $\sim$ is the trivial relation. In this case, the above type really captures $\mathbb{S}^1$ (their constructors are in one-to-one correspondence), which is not a set. For this reason, the appropriate definition is $A/\sim := \|\widehat{A/\sim}\|_0$.

The truncation HIT can be generalised to an n-truncation $\|-\|_n : \mathcal{U} \to \mathcal{U}$ for $n \geq 0$, but let us leave the introduction of these to the papers included in this thesis. In fact, there is also a lower level of truncation: the (-1)-truncation or *propositional* truncation. We have so far seen how HITs can be

used to internalise both homotopy-theoretic and set-theoretic notions. What about notions from the last entry in Table 2, i.e. logic? For the purpose of internalising logic, the propositional truncation is crucial. It is defined just like set truncation but with the path constructor in one dimension lower. For any type A, we define $\|A\|_{-1}$ to be the HIT generated by the following constructors.

- $|-| : A \to \|A\|_{-1}$
- $\mathsf{squash} : (x, y : \|A\|_{-1}) \to x = y$

The idea is that $\|A\|_{-1}$ is an exact copy of A but where all (possibly zero) elements have been identified – it is a *proposition*, to use HoTT terminology. This is useful when internalising e.g. logical quantifiers. Recall that given a predicate $B : A \to \mathcal{U}$, we can encode existential quantification by letting it be the dependent sum $(a : A) \times B(a)$. As mentioned earlier, this encodes a very strong constructive notion of existence which allows us to have several proofs (witnesses) of the same existentially quantified statement. In practice, this notion of existence is often too strong, and if we just wish to do elementary logic, we need to identify these different proofs. This is precisely the job done by the propositional truncation: it allows us to define an appropriate internal notion of (logical) existence by $\exists\,(a : A) \times B(a) := \|(a : A) \times B(a)\|_{-1}$.

In conclusion, HITs have many important roles when encoding 'standard' mathematics into type theory. They have, however, not yet quite reached their full potential. In particular, we are lacking tools for defining fibrations over them. Suppose, for instance, that we were to define some interesting fibration $B : \mathbb{S}^1 \to \mathcal{U}$ (say, the one encoding the universal covering of $\mathbb{S}^1$). The recursion principle for $\mathbb{S}^1$ tells us that this boils down to providing two pieces of data: a type X and a loop $p : X = X$. While we do always have a loop $\mathsf{refl}_X : X = X$, setting $p := \mathsf{refl}_X$ amounts to saying that B is constant. If we wish to construct more interesting fibrations than constant ones, we need new ways to construct paths between types. This is one of many applications of the univalence axiom.

Univalence One feature of our homotopical model of type theory is that its objects really are spaces up to homotopy. In other words, we have identified any two spaces X and Y which are (weakly) homotopy equivalent. This is something that is not quite reflected in our type theory. To remedy this, we may add Voevodsky's *univalence axiom* [Voe10], which says, somewhat simplified, that any two equivalent types are equal.

In order to internalise this, we first need to define a suitable notion of equivalence of types. Concretely, what we need to do is to define a predicate $\mathsf{isEquiv} : (X \to Y) \to \mathcal{U}$. One obvious attempt at a definition would be to say that a map $f : X \to Y$ is an equivalence if we can find a map $g : Y \to X$ such that it and f cancel out. Such a map g is called a *quasi-inverse* (following [UF13, §4.1]) and we can define the type of quasi-inverses of f as follows.

$$\mathsf{qinv}(f) := (g : Y \to X) \times ((x : X) \to g(f(x)) = x) \times ((y : Y) \to f(g(y)) = y)$$

For most practical purposes, quasi-inverses will be how we think of equivalences in HoTT. However, it turns out that setting $\mathsf{isEquiv}(f) := \mathsf{qinv}(f)$ does not quite make for an appropriate definition. The issue is that the type $\mathsf{qinv}(f)$ is not a proposition (i.e. it could contain two or more distinct elements). Even though the inverse function g promised by $\mathsf{qinv}(f)$ is unique, the additional pieces of data can be filled out in distinct ways. That is, we can provide different homotopies witnessing the cancellations of f and g. One option would, of course, be to simply use the propositional truncation from the previous section, setting $\mathsf{isEquiv}(f) := \|\mathsf{qinv}(f)\|_{-1}$. While this definition *will* turn out to be equivalent to the standard one, we will go with a more standard definition (which does not require HITs). In what follows, we define the fibre of a map $f : X \to Y$ over $y : Y$ by $\mathsf{fib}_f(y) := (x : X) \times (f(x) = y)$. Recall that a type X is contractible if it is an inhabited proposition, i.e there is some $x_0 : X$ together with a section $(x : X) \to x_0 = x$. It turns out that the property of being a proposition is closed under Π-types (sections) and that $\mathsf{isContr}(X)$ always is a proposition. This gives for a rather ingenious type-theoretic definition of equivalences due to Voevodsky [Voe10].

Definition 11 (Equivalences)**.** *We say that a map $f : X \to Y$ is an equivalence if for every $y : Y$, we have that $\mathsf{fib}_f(y)$ is contractible. That is, we define*

$$\mathsf{isEquiv}(f) := (y : Y) \to \mathsf{isContr}(\mathsf{fib}_f(y)).$$

We write $X \simeq Y$ for the type of equivalences, i.e. the pair of points (f, e_f) where $f : X \to Y$ and $e_f : \mathsf{isEquiv}(f)$. We often simply write $f : X \simeq Y$ and take e_f implicit.

In practice, we almost always treat equivalences in terms of quasi-invertible functions: we have a bi-implication $\mathsf{qinv}(f) \leftrightarrow \mathsf{isEquiv}(f)$ (consult [UF13, §4] for a more in-depth discussion). Definition 11 is, however, often crucial, as many proofs make use of the fact that $\mathsf{isEquiv}(f)$ is a proposition.

At this point, a good guess for what the formal statement of the univalence axiom could be is that it simply postulates a function $\mathsf{ua} : X \simeq Y \to X = Y$. This is almost it – ua will certainly be postulated into existence, but we need to know a little bit more about it than just the fact that it exists. Concretely, we would like to know what its inverse looks like. For this reason, the univalence axiom is introduced the other way around. This is done by first noting that, for all types X and Y, there is a map $\mathsf{coe} : X = Y \to X \simeq Y$ defined by $\mathsf{coe}(p) := \mathsf{transport}^{X \simeq (-)}(p, \mathsf{idEquiv}_X)$ (or, if you prefer to define it directly by path induction on p, it is simply given by $\mathsf{coe}(\mathsf{refl}_X) := \mathsf{idEquiv}_X$). Here, $\mathsf{idEquiv}_X$ denotes the identity $X \simeq X$.

Definition 12 (The univalence axiom)**.** *The univalence axiom is the statement that for any two types X and Y, the function $\mathsf{coe} : X = Y \to X \simeq Y$ is itself an equivalence (with an inverse called ua).*

The exact statement of univalence will not be too important for us here – the existence of $\mathsf{ua} : X \simeq Y \to X = Y$ is the core idea. We obtain HoTT by

adding, along with HITs, the univalence axiom to type theory. From now on, this will be the setting we work in.

One of the key features we obtain from the univalence axiom is the ability to transport proofs and constructions between equivalent structures. Whenever we have two types X and Y s.t. $X \simeq Y$, *any* predicate $B : \mathcal{U} \to \mathcal{U}$ which holds for X will also hold for Y (and vice versa). This is both a blessing and a curse: a corollary of this principle is that we only are able to define predicates which are homotopy invariant. This prevents us from defining several constructions in the arsenal of the working algebraic topologist. For instance, we can never hope to capture exactly what it means for a type to be a manifold[2]: if we had a predicate $\mathsf{isManifold} : \mathcal{U} \to \mathcal{U}$ expressing that an arbitrary type is a manifold, univalence would tell us that whenever $X \simeq Y$, we have an implication $\mathsf{isManifold}(X) \to \mathsf{isManifold}(Y)$ which certainly looks strange – this property should not be stable under weak homotopy equivalence.

As mentioned briefly earlier, univalence is crucial for the construction of non-trivial fibrations over HITs. Let us return to the example of defining a non-trivial fibration $B : \mathbb{S}^1 \to \mathcal{U}$. Like before, the data we need to provide will consist of a space X and an identification $p : X = X$. Unlike before, however, we now have other options than simply setting $p := \mathsf{refl}_X$. Instead, we can invoke univalence, which will allow us to replace the demand of a path $p : X = X$ with the demand of an equivalence $e : X \simeq X$ (so that we simply can set $p := \mathsf{ua}(e)$). Let us consider a concrete example due to [LS13] for $X = \mathbb{Z}$. In this case, we can set e to be the equivalence $1 + (-) : \mathbb{Z} \simeq \mathbb{Z}$. That is, we can define $B : \mathbb{S}^1 \to \mathcal{U}$ by

$$\begin{aligned} B(\mathsf{base}) &:= X \\ \mathsf{ap}_B(\mathsf{loop}) &:= \mathsf{ua}(1 + (-)). \end{aligned}$$

This fibration is non-trivial. To see this, note that we get a map $\mathsf{transport}^B(\mathsf{loop}, -) : B(\mathsf{base}) \to B(\mathsf{base})$. If our fibration were constant, it would be equal to the identity. For our fibration, however, this is not the case. We choose a point a in $B(\mathsf{base})$ – i.e. in $\mathbb{Z}$ – and compute

$$\mathsf{transport}^B(\mathsf{loop}, a) = \mathsf{coe}(\mathsf{ua}(1 + (-)))(a) = 1 + a. \tag{1}$$

Above, the first equality is a simple manipulation of transports (using that coe was also described as a transport) and the second equality holds by definition of the univalence axiom. You may recognise the fibration constructed from the classical construction of the universal covering of $\mathbb{S}^1$: each fibre is $\mathbb{Z}$ and the canonical loop lifts to addition by 1. We can proceed to show that its total space is contractible and obtain that $\Omega(\mathbb{S}^1) \simeq \mathbb{Z}$, where $\Omega(\mathbb{S}^1) := (\mathsf{base} = \mathsf{base})$. Thus, we have proved our first 'real' homotopy-theoretic result in HoTT. This proof, which is originally due to [LS13], is remarkably simple compared to its classical counterpart (when the classical proof is given from scratch, that is).

[2]This is a half-truth: there is work on defining an appropriate notion of a *homotopy* manifold in HoTT by [Buc+24].

We never have to verify the continuity of any of the maps involved, and we never have to worry about the possible existence of pathological loops in $\mathbb{S}^1$.

Univalence has many other important features. Perhaps most importantly, it can be used to *prove* function extensionality [Voe10]. In addition to this, it is, among other things, crucially used in the characterisation of path spaces of truncations and can be used to prove Aczel's *structure identity principle* (SIP) [Acz11] which roughly says that any two isomorphic structures (such as rings, groups, etc.) are equal. The SIP implies that any predicate that we can form in HoTT about, say, groups automatically is invariant under isomorphism, thereby making formal an informal principle employed by mathematicians on a daily basis. There is, of course, a lot more that can be said about the SIP and other applications of univalence, but let us be content with this so far – the reader who also reads the five papers included in this thesis should come across plenty of other applications of univalence.

2.3 Cubical type theory

One of the neat things about type theory is its close connection to *computation*. In general, we expect any type-theoretic construction to compute. As we can interpret types as program specifications, we should always be able to extract a concrete algorithm from any type-theoretic construction we make. For instance, if we construct a function $f : \mathbb{Z} \to \mathbb{Z}$ in type theory, we should be able to implement it as a computer program. If we wish to know what, say, $f(0)$ computes to, we should never have to find this out 'manually' (by mathematical proof); we should be able to simply run the program we constructed. We say that all of our constructions have *computational content*.

One issue with HoTT is that we have simply postulated the univalence axiom. What we never did, however, was to explain its computational content. This means that univalence becomes a computational black box – if our function f is described using univalence, we can no longer hope to be able to automatically extract an algorithm from it. One example is Equation (1). If we were only interested in the case when a is a fixed number, say $a := 0$, we should not have to do any work: the left-hand side should *compute* to 1, so the proof should be given by refl_1. However, as the left-hand side is described in terms of univalence, this does not work – the computation gets stuck.

It is still open whether 'standard' HoTT (systems like the one described here) can be equipped with a computational univalence axiom. One thing we can do, however, is to alter the type theory slightly. The resulting type theory, a *cubical type theory* (CuTT), will have slightly less general models (in its natural model, spaces are modelled as cubical sets rather than what we have so far called 'spaces' [BCH14; Coh+18; Ang+19]) but will have a computational univalence axiom. This flavour of HoTT is also currently better suited for implementing HITs [CHM18; CH19]. This will be the type theory all computer formalisations in this thesis are built on (more on this in §3). Nevertheless, most

of the thesis is written in an informal style which is agnostic with respect to flavour of HoTT. One should point out that there are several different versions of CuTT; the one we present here is closest to the one by [Coh+18] and, from now on, 'CuTT' will always mean this particular version.

The key difference between plain HoTT and CuTT is the treatment of the identity type. To get some intuition, let us start by giving an alternative definition of identity types in HoTT. In HoTT, we can define the type I, representing the unit interval, to be the HIT with the following constructors.

- $i_0, i_1 : I$
- $i_{0=1} : i_0 = i_1$

Now, consider a function $f : I \to A$. Such a function encodes precisely two points $f(i_0), f(i_1) : I$ together with a path $\mathsf{ap}_f(i_{0=1}) : f(i_0) = f(i_1)$. Thus, for any points $x, y : A$, a function $f : I \to A$ with $f(i_0) := x$ and $f(i_1) := y$ captures precisely the identity type $x = y$. This is, in spirit, the approach taken when defining the identity type in CuTT: here, the identity type is not the usual inductive one (although it is consistent to add it) but one described in terms of a primitive interval (pre-)type I. This type comes with two points i_0 and i_1 but is not a HIT – that would be rather circular, as the notion of a HIT presupposes a notion of identity. This type lives in a special universe, which essentially makes it untouchable – it is only there to provide us with a definition of paths and is not there to be reasoned about (again, this would lead to circularity). So, a path $x = y$ in CuTT is *literally* a function $f : I \to A$ such that $f(i_0) := x$ and $f(i_1) := y$. For instance, we can define $\mathsf{refl}_x : x = x$ to be the constant path $\mathsf{refl}_x(i) := x$. The interval does come with some primitive operations which are needed to actually make it useful. In particular, it comes with a path reversal operation $\sim : I \to I$ and meet and join operations $\vee, \wedge : I \times I \to I$. These satisfy the following (strict) equalities

$$\begin{array}{rrr} \sim i_0 := i_1 & i_0 \wedge j := i_0 & i_0 \vee j := j \\ \sim i_1 := i_0 & i_1 \wedge j := j & i_1 \vee j := i_1 \\ \sim (\sim i) := i & i \wedge j := j \wedge i & i \vee j := j \vee i \end{array}$$

as well as the appropriate distribution and associativity laws turning I into a de Morgan algebra. For instance, $\sim$ is what gives rise to symmetry: for a path $p : x = y$, we can define its inverse path by $p^{-1}(i) := p(\sim i)$. Another example is the principle of path induction, which is a theorem in CuTT (rather than a built-in rule). This is proved by showing that the total space of the path fibration is contractible, which can be done by providing paths defined in terms of $\wedge$. For full transparency, we should mention that the proof of path induction also uses a $\mathsf{transport}$ – an operation which is in fact another primitive of CuTT. Nevertheless, this is more detail than we will ever need. The fact that we have path induction means that we, for all intents and purposes, can treat CuTT

as any other implementation of HoTT and use the same informal language we have used so far.

One remarkable thing about CuTT is that it makes function extensionality trivial. Indeed, a proof that two functions $f, g : A \to B$ are equal boils down to providing a path $p : I \to (A \to B)$ s.t. $p(i_0) := f$ and $p(i_1) := g$. If we uncurry this, it is equivalent to saying that we need to provide a function $p : I \times A \to B$ s.t. $p(i_0, x) := f(x)$ and $p(i_1, x) := g(x)$. These two ways of viewing the path type $f = g$ amounts precisely to function extensionality.

There are more primitives than the ones introduced so far. For instance, there is the notorious Glue type which is used to (constructively) *prove* univalence, thereby endowing it with computational content. We will never explicitly rely on this construction either.

Two things that do become simpler in CuTT are dependent paths and HITs. When defining dependent identity in §1.2, we pointed out that plain paths really are special cases of their dependent analogue; yet, we used non-dependent identity to define dependent identity. In CuTT, dependent paths are simple. Let $B : I \to \mathcal{U}$ be any 'line' of types (i.e. a path of types $B(i_0) = B(i_1)$ in the universe $\mathcal{U}$) and let $x : B(i_0)$ and $y : B(i_1)$. A dependent path from x to y over B is simply a dependent function $p : (i : I) \to B(i)$ s.t. $p(i_0) := x$ and $p(i_1) := y$ hold strictly. In the special case when B is constant, such a (no longer dependent) function p is a regular path. Removing the need for $\mathsf{transport}$s in the definition of dependent paths often strips away a fair bit of bureaucracy and makes dependent paths very natural. For instance, the identity type of a dependent sum $(x : X) \times B(x)$ is easy to understand: proving an equality $(x, b_x) = (y, b_y)$ is done by providing a dependent function $p : (i : I) \to (x : X) \times B(x)$ s.t. $p(i_0)$ and $p(i_1)$ reduce to respective pair. In order to define $p(i)$ for $i : I$, we thus need to provide an element $p_1(i) : X$ s.t. $p_1(i_0) := x$ and $p_1(i_1) := y$ as well as an element $p_2(i) : B(p_1(i))$ s.t. $p_2(i_0) := b_x$ and $p_2(i_1) := b_y$. This is exactly the principle described in §1.2, but this time it holds definitionally.

Arguably, HITs also become more natural in CuTT and the only fully fledged computer program implementations of these rely on some form of CuTT (e.g. `Cubical Agda` [VMA21]). In CuTT there is (in theory) no real difference between ordinary inductive types and HITs. As paths are described as functions out of the interval, we can see path constructors appearing in HITs as the special case of constructors taking input from the interval. Naïvely, the HIT defining $\mathbb{S}^1$ could simply be thought of as a plain inductive type with constructors

- $\mathsf{base} : \mathbb{S}^1$
- $\mathsf{loop} : I \to \mathbb{S}^1$

but with the caveat that $\mathsf{loop}(i_0) := \mathsf{base}$ and $\mathsf{loop}(i_1) := \mathsf{base}$. This makes it easier to implement HITs in proof assistants.

3 Proof assistants

One thing that is easy to forget is that, despite all the buzzwords from homotopy theory we have used here, type theory is a programming language. This means that a proof written in type theory can effectively be understood by a computer. Indeed, it is, to some extent, used in the foundations of almost every programming language. Some functional languages, like `Haskell`, are very close to the type theory we have described in §1. On the other hand, if a computer can understand type theory, then it can understand statements and proofs in logic (by the Curry–Howard correspondence). For a proof in type theory to be correct, it simply needs to type check – something computers are very good at verifying.

Concretely, this means that type theory works incredibly well as a language for so-called *proof assistants*. These are essentially programming languages developed for the particular task of checking the correctness of proofs. If we use type theory, 'verifying a proof' simply means type checking it. Hence, a type-theoretic proof assistant is essentially a programming language whose syntax only allows us to write correct code. There are several examples of proof assistants and almost all of them use some form of type theory. This even applies to proof assistants such as `Lean` [Mou+15], despite the fact that it primarily intended for 'traditional' set-theoretic mathematics (another point for type theory as a foundation of mathematics!). In all papers presented in this thesis, we have used the proof assistant `Agda` [Agda], or rather its cousin `Cubical Agda` [VMA21], to verify all key results. We say that these results have been *(computer) formalised* (or, sometimes, *mechanised*). It may seem like I am downplaying this part of the thesis by introducing it only here, in the last paragraphs of the introduction, but this is not the message I want the reader to take away. The computer formalisations in this thesis are as much a contribution as the mathematical results.

3.1 Agda

The computer formalisation of this thesis uses the proof assistant `Cubical Agda` [VMA21] which adds certain features from CuTT to the proof assistant `Agda` [Agda]. Let us start by discussing the latter, as it is the basis of `Cubical Agda`. `Agda` is a type-theoretic language incredibly close to MLTT which, recall, is what we mean by type theory in this thesis. In fact, the reader has already been introduced to some of `Agda`'s syntax without realising: for instance, the notation $(x : X) \to B(x)$ for dependent functions is not traditionally used in type theory, but in `Agda` it is. To give a very brief overview of how the language works, let us go through how the basic concepts from §1.2 are implemented in `Agda`.

Universes Agda is equipped with a hierarchy of universes, usually written Set ℓ where ℓ is a universe level (i.e. an index). Here, we will rename Set to Type and simply omit the index, with the convention that Type without a universe level denotes the lowest universe Type ℓ-zero. If we wish to define a new type, the syntax is as follows.

```
myType : Type
myType = someDefinition
```

Functions Functions are introduced pretty much exactly like we would write them using pen and paper. For instance (presupposing a definition of the natural numbers and a squaring operation), here is how we can define the function $x \mapsto x^2 + x$:

```
myFun : ℕ → ℕ
myFun x = x ² + x
```

We can also define the same function using lambda notation:

```
myFunLambda : ℕ → ℕ
myFunLambda = λ x → x ² + x
```

These two definitions will be equal *by definition* in Agda. Another way of introducing functions is by pattern matching, but for that we need to see how inductive types are dealt with.

Inductive types, dependent types and dependent functions and sums All the inductive types from §1.2 can easily be captured using Agda's primitive data syntax. The syntax is almost identical to our presentation.

```
data ℕ : Type where
  zero : ℕ
  suc : ℕ → ℕ
```

```
data Bool : Type where
  true : Bool
  false : Bool
```

```
data 𝟙 : Type where
  * : 𝟙
```

```
data ⊥ : Type where
```

We clarify that the last type, $\bot$, is the data type with no constructors, i.e. it is the empty type. The types above all come automatically equipped with their recursion/induction principles – Agda supports *pattern matching* (compatible with HoTT [CDP14]) which is its way of capturing these notions. To showcase how this works, while simultaneously showing how to define dependent types, let us define the isEven-predicate from §1.2.

```
isEven : ℕ → Type
isEven zero = 𝟙
```

```
isEven (suc zero) = ⊥
isEven (suc (suc n)) = isEven n
```

Above, we have pattern matched on a variable m of type $\mathbb{N}$ and `Agda` thereby introduces a case-split: either $m := 0$ or $m := \mathsf{suc}(n)$ for some $n : \mathbb{N}$. We can now carry out our first proof in `Agda` – the proof that $2 + n$ is even if n is even.

```
isEven2+ : (n : ℕ) → isEven n → isEven (suc (suc n))
isEven2+ n p = p
```

This is an instance of a dependent function. For completeness, let us also prove the riveting theorem 'there exists an even number' to showcase the implementation of the dependent sum type. We provide 2 (which, in `Agda`, is short for suc (suc zero)) as a witness.

```
existsEven : Σ[ n ∈ ℕ ] (isEven n)
existsEven = 2 , *
```

Identity types We can easily define Martin-Löf's inductive identity type in `Agda`:

```
data _≡_ (x : A) : A → Type where
  refl : x ≡ x
```

We use the triple bar notation to avoid clashing with '=' which already is used in `Agda`'s syntax. Just like the other inductive types we have implemented, the identity type automatically comes equipped with its induction principle, which again can be used by pattern matching (i.e. by asking `Agda` to 'case split' on a variable of type $x \equiv y$). Here are all of the elementary definitions concerning identity types from §1.2:

```
_⁻¹ : x ≡ y → y ≡ x
refl ⁻¹ = refl
```

```
_·_ : x ≡ y → y ≡ z → x ≡ z
refl · q = q
```

```
ap : (f : A → B) → x ≡ y → f x ≡ f y
ap f refl = refl
```

```
transport : (B : A → Type) (p : x ≡ y) → B x → B y
transport B refl x = x
```

Despite how confusing the J-rule (path induction) may be, the above definition should showcase how easy it is to use in `Agda`. As discussed, the J-rule does not work whenever the end-point is not free. A special case of this proscription was presented as Non-Theorem 1. We could attempt to prove it in `Agda` by

```
haligh-a-Lie : (p : x ≡ x) → p ≡ refl
haligh-a-Lie refl = refl
```

but, fortunately, `Agda` will complain:

```
I'm not sure if there should be a case for the constructor refl,
because I get stuck when trying to solve the following unification
problems (inferred index ≟ expected index):
  x₁ ≟ x₁
```

Computation/normalisation When we prove in Agda, we often rely on constructions computing/normalising. For instance, if we were interested in showing that for any path $p : x \equiv y$, we have that $p \cdot p^{-1} = \mathsf{refl}_x$, we would prove this as follows (of course, using J).

```
rCancel : (p : x ≡ y) → p ∙ p ⁻¹ ≡ refl
rCancel refl = refl
```

The reason this proof works is that after pattern matching on p, Agda will rewrite the target type to $\mathsf{refl} \cdot \mathsf{refl}^{-1} \equiv \mathsf{refl}$ – a type which Agda now will start simplifying or *normalising*. Agda will see that we *defined* refl^{-1} to be refl and, in turn, $\mathsf{refl} \cdot \mathsf{refl}$ to again be refl. Hence, by computing all the functions involved, Agda simplifies to the goal to make it $\mathsf{refl} \equiv \mathsf{refl}$, which we can prove by refl.

There are also ways in which Agda's computational features can provide less fundamental aid in our theorem proving. Suppose, for instance, that we are interested in computing some function – call it $\mathsf{ackermann} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ – at some concrete input. Say we wish and we wish to prove that this output is equal to some other number – call it $2 \uparrow^3 4 - 3$. One way of doing this is by some form of traditional 'manual' proof: the way we do it in classical mathematics. Another way would be to simply compute the function (this works if it is constructively defined). Agda's computational feature allows for this proof approach.

```
trustMe : ackermann (5 , 1) ≡ 2 ↑³ 4 - 3
trustMe = refl
```

The above proof (when it finishes type checking) is a verified computation or a 'proof by computation'. This is a powerful proof technique available in constructive proof assistants like Agda which will be explored in some of the papers in this thesis.

3.2 Cubical Agda

You may have noticed that the constructions covered in our brief introduction to Agda all were constructions from plain type theory. However, we did not say anything about the two key notions from HoTT: HITs and univalence. This is because they do not have native support in plain Agda.[3] In Cubical Agda, however, they are supported – this is my reason for having chosen to carry out my formalisations in this language. This is not to say that HITs and univalence (and HoTT in general) cannot be meaningfully reasoned about in vanilla Agda – some of the largest libraries of computer formalised univalent

[3]There are however still ways of implementing HITs, e.g. using 'Licata's trick' [Lic11].

mathematics use this version, e.g `HoTT-Agda` [Bru+18], `agda-unimath` [Rij+] and `TypeTopology` [Ec]. In plain `Agda`, we simply postulate these concepts into existence. While this is perfectly sufficient for formalising results from HoTT, it does come with the caveat that it breaks computation. My choice of `Cubical Agda` is not ideological – I simply appreciate its computational aspects. Let us showcase the new features. We will be brief as this topic is also covered in Paper II.

Paths Although we can still define the usual inductive path type à la Martin-Löf in `Cubical Agda`, the primary way of capturing paths in this proof assistant is via the interval (pre-)type described in §2.3. We keep using the same notation for path types but note that, in the following pages, they are defined cubically. We can mimic the construction of the constant path and path reversal by copying the definitions in §2.3 pretty much verbatim:

```
refl : (x : A) → x ≡ x
refl x i = x
```

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

In addition to the above, we can also prove function extensionality in 1 line.

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

Above, the curly brackets indicate implicit arguments. It would be natural to ask what happened to the definition of path composition; we do not talk about this (it uses a primitive called `hcomp` which takes some time getting used to and is never used explicitly in this thesis).

HITs We can define HITs in `Cubical Agda` just like we defined plain inductive types in `Agda`. For instance, here is the definition of $\mathbb{S}^1$.

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

We have access to its recursion and induction principles immediately via pattern matching. For instance, we can define the canonical (-1)-degree map by

```
inv𝕊¹ : 𝕊¹ → 𝕊¹
inv𝕊¹ base = base
inv𝕊¹ (loop i) = loop (~ i)
```

and prove that it is an involution

```
inv𝕊¹Involution : (x : 𝕊¹) → inv𝕊¹ (inv𝕊¹ x) ≡ x
inv𝕊¹Involution base = refl
inv𝕊¹Involution (loop i) = refl
```

Univalence Arguably, *the* key feature of `Cubical Agda` is that it natively supports the univalence axiom. Actually, using the word 'axiom' is wrong – here, it is a theorem. Its main component, the function lifting equivalences to paths, can be constructed in `Agda` using the following code

```
ua : A ≃ B → A ≡ B
ua e i = Glue B (λ { (i = i0) → (A , e) ; (i = i1) → (B , idEquiv B) })
```

This definition is, of course, not very illuminating. It uses strange new syntax together with a new (built-in) Glue type. Like most users of `Cubical Agda`, we will simply accept this definition and move on. It is very uncommon that we actually have to understand this definition in order to make use of univalence. Instead, let us showcase this function by defining the fibration giving rise to the universal covering of the $\mathbb{S}^1$ described in §2.2. The fibration, which we here call helix, can be defined by

```
helix : 𝕊¹ → Type
helix base = ℤ
helix (loop i) = ua +1Equiv i
```

The equality proved in Equation (1) with $a := 0$ can now, thanks to the computational content of univalence, be proved simply by refl.

```
helixComputes : transport (λ i → helix (loop i)) 0 ≡ 1
helixComputes = refl
```

Papers I and, in particular, Paper II will discuss statements similar to the above which also can be proved by computation in `Cubical Agda`. This concludes this very brief introduction to `Cubical Agda`.

4 The context and contributions of this thesis

The publication of the HoTT book [UF13] kick-started the large-scale project of developing (both old and new) mathematics in the framework of HoTT, including the formalisation of these results in HoTT-based proof assistants like `(Cubical) Agda`. These results are not just interesting because they are more directly available for computer formalisation – they also generalise their classical counterparts. Indeed, HoTT can be interpreted in any suitably structured ∞-topos [Shu19] and thus provides an immediate generalisation of traditional mathematics with its usual model in (the very particular ∞-topos of) sets.

So far, we have seen quick progress in several traditional domains of mathematics, such as algebraic geometry [ZM23; ZH24; CCH24; Che+24], (higher) group theory [BDR18; Swa22; Bez+25], set theory [Jon+23; GS24; Gra+24], graph theory [PG24] and other subjects. The area within which we have seen probably the most activity, however, is that of (synthetic[4]) homotopy theory.

[4]Here, 'synthetic' means that we are manipulating the homotopy types of spaces directly.

We have already seen an example of how HoTT can be used to reason about homotopy theory in §2.1, in particular when discussing the construction of the universal covering of $\mathbb{S}^1$ (following Licata and Shulman [LS13]). In the last 10 years, we have seen an impressive amount of work dedicated to the study of homotopy theory in HoTT. There have been significant contributions made to e.g. the study of homotopy groups/types of spheres [LS13; Bru16; BR18; Bak23; Cag+24], cohomology [Bru16; BF18; Doo18; Wär23; CF23], homology [Gra18; CS23], path spaces of pushouts [KR21; Wär24], H-spaces [BR18; Buc+23], acyclic spaces [BdR25], and so on.[5] Much of this research is computer formalised in proof assistants such as `Agda`, `Cubical Agda`, `Coq` and `Lean`. My intention with this thesis is to make a contribution to these efforts.

The first three papers of this thesis are dedicated to filling in some of the gaps in the literature by giving a formal account of details missing in some key proofs and constructions in Brunerie's landmark PhD thesis on computing $\pi_4(\mathbb{S}^3)$ in HoTT. In particular, Paper I introduces a complete construction of cohomology rings, a primary contribution being the previously missing proof of the associativity of the cup product, and Paper II presents, relying on the results from Paper I, a complete computer formalisation of Brunerie's proof in `Cubical Agda`. Paper III attacks the closely related problem of proving the symmetric monoidality of smash products, something which Brunerie [Bru16] only sketched in his thesis and left as future work.

The last two papers instead explore new avenues for synthetic homotopy theory in HoTT by addressing open ends left by other papers in the field. Paper IV investigates to which extent it is possible to use Buchholtz and Favonia's definition of cellular cohomology [BF18] to define cellular *homology*. In particular, the paper includes a construction of cellular homology in terms of a constructive version of the cellular approximation theorem. This approach can, completely analogously, be used to obtain Buchholtz and Favonia's cohomology theory, and thus the paper provides a unified approach to cellular homology and cohomology in HoTT. Paper V, on the other hand, consists of a synthetic development of the Steenrod squares in HoTT. The paper contains proofs of the fact that these cohomology operations satisfy a collection of defining axioms (such as the Cartan formula and stability). The proofs of these properties were left open by Brunerie [Bru17], whose definition of the Steenrod squares we use. This paper is perhaps particularly interesting because it heavily relies on concepts which are, to some extent, especially natural in the language of HoTT, such as unordered pairs and joins.

Loose ends and future work There are several research problems suggested or made open by the work in this thesis. This particularly applies to Papers IV and V, but these problems naturally touch upon material from the first three papers. Paper IV was, in fact, intended as a foundation for an ongoing project together with Mörtberg and Pujet on using implementations of cellular

[5] There are of course many more topics studied and papers produced, but I cannot mention them all here.

homology and cohomology in `Cubical Agda` to solve computational problems such as that of computing the Brunerie number in Paper II or the numbers in §6 of Paper I. On the other hand, the proof of the (cellular) Hurewicz theorem (and, in particular, its formalisation) is part of an ongoing project, led by Barton and Milner, on the formalisation of Barton and Campion's [Bar22] proof of the Serre finiteness theorem in `Cubical Agda`.

For Paper V, the continuation is very clear: we would like to be able to define not only the Steenrod squares but the higher Steenrod p-powers for any prime p. This is an interesting problem which really puts the expressivity of HoTT to the test. Indeed, the proofs are already involved in the case of $p = 2$. If we are to generalise these proofs, we need to define p-fold (unordered) smash products or joins. These are constructions we simply do not currently know how to define.

Summaries of included papers

Paper I

Paper I, *Computational Synthetic Cohomology Theory in Homotopy Type Theory*, concerns the development of cohomology and summarises, to a large extent, the first half of my PhD. The paper generalises work by Brunerie et al. [BLM22] (included in my licentiate thesis [Lju23]) and explains the mathematics underpinning the first complete computer formalisation of cohomology rings in HoTT by Lamiaux et al. [LLM23]. Concretely, it extends the definition of the cup product by Brunerie et al. to cohomology with arbitrary ring coefficients – a definition which is new to HoTT and significantly simplifies pre-existing definitions. As a result, we are able to provide a complete construction of cohomology rings in HoTT. This problem was previously open due to the perceived difficulty of proving associativity. We also verify that our cohomology theory satisfies a version of the Eilenberg–Steenrod axioms and construct the Gysin sequence (generalising the construction by Brunerie [Bru16]).

In addition to this, we carry out some concrete computations of cohomology groups and rings. For instance, we compute the cohomology ring of $\mathbb{R}P^\infty$ in $\mathbb{Z}/2\mathbb{Z}$-coefficients – something which is crucially used in Paper V.

We finally discuss some new 'Brunerie numbers', i.e. numbers which are definable in `Cubical Agda` and contain non-trivial information about, in this case, cohomological constructions. We benchmark these in order to better understand the feasibility of 'proof by computation' in `Cubical Agda`. All results in the paper have been formalised.

Paper II

Paper II, *Formalising and Computing the Fourth Homotopy Group of the* 3*-Sphere in Cubical Agda*, is an extended version of a paper by Ljungström and Mörtberg [LM23]. This paper is written in a mock-`Cubical Agda` language and

presents a computer formalisation of Brunerie's 2016 proof of $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$. This formalisation contains results and constructions such as Whitehead products, (a special case of) the James construction, the Hopf invariant, cohomology rings (borrowed from Paper I) and much more. As Brunerie's thesis contained some minor but seemingly difficult gaps, providing a formalisation of his results constituted a, to some, important open problem in HoTT.

Another open problem (partially) solved in this paper is that of computing the Brunerie number. This is a number $n : \mathbb{N}$ s.t. $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/n\mathbb{Z}$. The formalisation of Brunerie's thesis showed that this number has absolute value 2, but actually extracting the value of n purely by computation in e.g. `Cubical Agda` – the problem of computing the Brunerie number – turns out to be very difficult. All attempts to compute Brunerie's original definition of this number have simply run out of memory. We provide a new and simplified definition of the Brunerie number which we *can* compute. This is done by slightly altering the definition of homotopy groups in terms of joins, which in turn allows us to better understand the interaction between certain maps and Whitehead products (one such product is what gives rise to the original Brunerie number).

Paper III

Paper III, *Symmetric monoidal smash products in homotopy type theory*, addresses another long-standing open problem in HoTT, namely that of verifying that the smash product is (1-coherent) symmetric monoidal. This was previously open due to the difficult coherence problems which appear in the proof of, in particular, the pentagon axiom (see e.g. [Bru18]).

The paper shows how a seemingly unrelated result by Cavallo [Cav21] regarding pointed functions and homogeneous types can be used to automatically infer certain higher coherences when proving equalities of maps defined over smash products. We start by providing an informal heuristic that captures this idea. We then use this heuristic to provide a proof of the fact that smash products are symmetric monoidal and, in particular, satisfy the pentagon axiom. We end the paper with a formal statement intended to capture the heuristic. The statement drastically reduces the amount of data needed when constructing homotopies over iterated smash products and should be useful when working with large smash products in general. All results have been formalised in `Cubical Agda`.

Paper IV

Paper IV, *Cellular Methods in Homotopy Type Theory* concerns the development of the basic theory of CW complexes and cellular homology in HoTT. In this paper, we revisit a definition of cellular cohomology in HoTT by Buchholtz

and Favonia [BF18] and attempt to develop the analogous homology theory instead. In particular, in order to obtain functoriality, we need to prove a suitable version of the cellular approximation theorem. The usual formulation of this theorem is not constructive and, for this reason, a large part of the contribution in this paper is the constructivisation of this classical result. On a similar note, we prove a special case of the CW-approximation theorem which says that the notion of an n-connected finite CW complex coincides with the notion of an n-connected type (whenever this type is a finite CW complex) – another seemingly non-constructive result. This allows us to prove the Hurewicz theorem for our homology theory.

In addition to the above, we prove the Eilenberg–Steenrod axioms for our theory. In order to state them, we also need to provide certain pushouts of cellular maps with a CW structure. Most results have been formalised in `Cubical Agda`.

Paper V

In Paper V, *The Steenrod squares via unordered joins*, we lay out the basic theory of the Steenrod squares in HoTT. The Steenrod squares are a family of cohomology operations in $\mathbb{Z}/2\mathbb{Z}$ cohomology introduced in HoTT by [Bru17]. While Brunerie provided a definition, the question of whether these squares satisfy any of their classical characterising properties has remained open. In this paper, we revisit Brunerie's construction and analyse it in terms of so-called unordered joins. Using these, we are able to prove a certain Fubini-like theorem from which we can deduce several of the Steenrod squares' properties (the Cartan formula and Adem relations, among other things). The constructions we provide are very much native to the language of HoTT and come with both advantages and disadvantages which are discussed at several points in the paper. Key technical results are supported by a formalisation in `Cubical Agda`.

Bibliography

[Acz11] Peter Aczel. On Voevodsky's Univalence Axiom. Talk given at the Third European Set Theory Conference. 2011. URL: https://staff.cs.manchester.ac.uk/~petera/Recent-Slides/Edinburgh-2011-slides_pap.pdf.

[Agda] The Agda Development Team. The Agda Programming Language. 2023. URL: http://wiki.portal.chalmers.se/agda/.

[Ang+19] Carlo Angiuli et al. "Syntax and Models of Cartesian Cubical Type Theory". Preprint. Feb. 2019. URL: https://github.com/dlicata335/cart-cube.

[Bak23] Raymond Baker. "Eckmann-hilton and the Hopf Fibration In Homotopy Type Theory". Honor's thesis. University of Colorado Boulder, Apr. 2023.

[Bar22] Reid Barton. Finite presentability of homotopy groups of spheres. Talk at the Seminar on Homotopy Type Theory at CMU, presenting joint work with Tim Campion. 2022. URL: https://www.cmu.edu/dietrich/philosophy/hott/seminars/previous.html.

[BCH14] Marc Bezem, Thierry Coquand and Simon Huber. "A Model of Type Theory in Cubical Sets". *19th International Conference on Types for Proofs and Programs (TYPES 2013)*. Ed. by Ralph Matthes and Aleksy Schubert. Vol. 26. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014, pp. 107–128. ISBN: 978-3-939897-72-9. DOI: 10.4230/LIPIcs.TYPES.2013.107.

[BDR18] Ulrik Buchholtz, Floris van Doorn and Egbert Rijke. "Higher Groups in Homotopy Type Theory". *Proceedings of the 33rd Annual ACM/ IEEE Symposium on Logic in Computer Science*. LICS '18. Oxford, United Kingdom: Association for Computing Machinery, 2018, pp. 205–214. ISBN: 9781450355834. DOI: 10.1145/3209108.3209150.

[BdR25] Ulrik Buchholtz, Tom de Jong and Egbert Rijke. Epimorphisms and Acyclic Types In Univalent Foundations. *The Journal of Symbolic Logic* (Feb. 2025), pp. 1–36. ISSN: 1943-5886. DOI: 10.1017/jsl.2024.76.

[Bez+25] Marc Bezem et al. Symmetry. *https://github.com/UniMath/SymmetryBook*. Commit: 8f95b7d. 25th Mar. 2025.

[BF18] Ulrik Buchholtz and Kuen-Bang Hou Favonia. "Cellular Cohomology in Homotopy Type Theory". *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. Oxford, United Kingdom: Association for Computing Machinery, 2018, pp. 521–529. ISBN: 9781450355834. DOI: 10.1145/3209108.3209188.

[BG11] Benno van den Berg and Richard Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society* 102.2 (2011), pp. 370–394. DOI: 10.1112/plms/pdq026.

[BLM22] Guillaume Brunerie, Axel Ljungström and Anders Mörtberg. "Synthetic Integral Cohomology in Cubical Agda". *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*. Ed. by Florin Manea and Alex Simpson. Vol. 216. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 11:1–11:19. ISBN: 978-3-95977-218-1. DOI: 10.4230/LIPIcs.CSL.2022.11.

[BM75] Donald W. Barnes and John M. Mack. "Zermelo-Fraenkel Set Theory". *An Algebraic Introduction to Mathematical Logic*. New York, NY: Springer New York, 1975, pp. 52–61. ISBN: 978-1-4757-4489-7. DOI: 10.1007/978-1-4757-4489-7_6.

[BR18] Ulrik Buchholtz and Egbert Rijke. The Cayley-Dickson Construction in Homotopy Type Theory. *Higher Structures* 2.1 (2018), pp. 30–41.

[Bru+18] Guillaume Brunerie et al. Homotopy Type Theory in Agda. 2018. URL: https://github.com/HoTT/HoTT-Agda.

[Bru16] Guillaume Brunerie. "On the homotopy groups of spheres in homotopy type theory". PhD thesis. Université Nice Sophia Antipolis, 2016. arXiv: 1606.05916.

[Bru17] Guillaume Brunerie. "The Steenrod squares in homotopy type theory". Abstract at *23rd International Conference on Types for Proofs and Programs (TYPES 2017)*. 2017. URL: https://types2017.elte.hu/proc.pdf#page=45.

[Bru18] Guillaume Brunerie. "Computer-generated proofs for the monoidal structure of the smash product". *Homotopy Type Theory Electronic Seminar Talks*. Nov. 2018. URL: https://www.uwo.ca/math/faculty/kapulkin/seminars/hottest.html.

[Buc+23] Ulrik Buchholtz et al. Central H-spaces and banded types. 2023. arXiv: 2301.02636.

[Buc+24] Ulrik Buchholtz et al. Tangent bundles and Euler classes. Extended abstract at *Workshop on Homotopy Type Theory/Univalent Foundations (HoTT/UF 2024)*. 2024. URL: https://hott-uf.github.io/2024/HoTTUF_2024_paper_22.pdf.

[Cag+24] Pierre Cagne et al. “On symmetries of spheres in univalent foundations”. *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS ’24. Tallinn, Estonia: Association for Computing Machinery, 2024. ISBN: 9798400706608. DOI: 10.1145/3661814.3662115.

[Cav21] Evan Cavallo. Pointed functions into a homogeneous type are equal as soon as they are equal as unpointed functions. Agda formalization, part of the cubical library. 2021. URL: https://agda.github.io/cubical/Cubical.Foundations.Pointed.Homogeneous.html#1616.

[CCH24] Felix Cherubini, Thierry Coquand and Matthias Hutzler. A foundation for synthetic algebraic geometry. *Mathematical Structures in Computer Science* 34.9 (Oct. 2024), pp. 1008–1053. ISSN: 1469-8072. DOI: 10.1017/s0960129524000239.

[CDP14] Jesper Cockx, Dominique Devriese and Frank Piessens. Pattern matching without K. *SIGPLAN Not.* 49.9 (Aug. 2014), pp. 257–268. ISSN: 0362-1340. DOI: 10.1145/2692915.2628139.

[CF23] J. Daniel Christensen and Jarl G. Taxerås Flaten. Ext groups in Homotopy Type Theory. 2023. arXiv: 2305.09639.

[CH19] Evan Cavallo and Robert Harper. Higher Inductive Types in Cubical Computational Type Theory. *Proceedings of the ACM on Programming Languages* 3.POPL (Jan. 2019), 1:1–1:27. ISSN: 2475-1421. DOI: 10.1145/3290314.

[Che+24] Felix Cherubini et al. Projective Space in Synthetic Algebraic Geometry. 2024. arXiv: 2405.13916.

[CHM18] Thierry Coquand, Simon Huber and Anders Mörtberg. “On Higher Inductive Types in Cubical Type Theory”. *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS ’18. Oxford, United Kingdom: ACM, 2018, pp. 255–264. ISBN: 978-1-4503-5583-4. DOI: 10.1145/3209108.3209197.

[Chu32] Alonzo Church. A Set of Postulates for the Foundation of Logic. *Annals of Mathematics* 33.2 (1932), pp. 346–366. ISSN: 0003486X, 19398980. URL: http://www.jstor.org/stable/1968337 (visited on 12/02/2025).

[Coh+18] Cyril Cohen et al. “Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom”. *21st International Conference on Types for Proofs and Programs (TYPES 2015)*. Ed. by Tarmo Uustalu. Vol. 69. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 5:1–5:34. ISBN: 978-3-95977-030-9. DOI: 10.4230/LIPIcs.TYPES.2015.5.

[Coq92] Thierry Coquand. “Pattern matching with dependent types”. Workshop on Logical Frameworks. Preliminary proceedings. Båstad, 1992.

[CS23] J Daniel Christensen and Luis Scoccola. The Hurewicz theorem in homotopy type theory. *Algebraic & Geometric Topology* 23 (July 2023), pp. 2107–2140. DOI: 10.2140/agt.2023.23.2107.

[DAL24] Stefania Damato, Thorsten Altenkirch and Axel Ljungström. Formalising inductive and coinductive containers. Submitted. 2024. arX-iv: 2409.02603.

[Doo18] Floris van Doorn. "On the Formalization of Higher Inductive Types and Synthetic Homotopy Theory". PhD thesis. Carnegie Mellon University, May 2018. arXiv: 1808.10690.

[Ec] Martín H. Escardó and contributors. TypeTopology. Agda development. URL: https://github.com/martinescardo/TypeTopology.

[Gra+24] Daniel Gratzer et al. The category of iterative sets in homotopy type theory and univalent foundations. *Mathematical Structures in Computer Science* 34.9 (Oct. 2024), pp. 945–970. ISSN: 1469-8072. DOI: 10.1017/s0960129524000288.

[Gra18] Robert Graham. Synthetic Homology in Homotopy Type Theory. Preprint. 2018. arXiv: 1706.01540.

[GS24] Håkon Robbestad Gylterud and Elisabeth Stenholm. Univalent Material Set Theory. 2024. arXiv: 2312.13024.

[Hed98] Michael Hedberg. A coherence theorem for Martin-Löf's type theory. *Journal of Functional Programming* 8.4 (1998), pp. 413–436. DOI: 10.1017/S0956796898003153.

[Hof97] Martin Hofmann. "Syntax and Semantics of Dependent Types". *Semantics and Logics of Computation*. Ed. by Andrew M. Pitts and P.Editors Dybjer. Publications of the Newton Institute. Cambridge University Press, 1997, pp. 79–130.

[How80] William Alvin Howard. "The Formulae-as-Types Notion of Construction". *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*. Ed. by Haskell Curry et al. Academic Press, 1980.

[HS98] Martin Hofmann and Thomas Streicher. "The groupoid interpretation of type theory". *Twenty-Five Years of Constructive Type Theory*. Ed. by Giovanni Sambin and Jan Smith. Vol. 36. Oxford Logic Guides. Oxford University Press, 1998, pp. 83–111.

[Jon+23] Tom de Jong et al. "Set-Theoretic and Type-Theoretic Ordinals Coincide". *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, June 2023, pp. 1–13. DOI: 10.1109/lics56636.2023.10175762.

[KR21] Nicolai Kraus and Jakob von Raumer. "Path spaces of higher inductive types in homotopy type theory". *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '19. Vancouver, Canada: IEEE Press, 2021.

[Lic11] Dan Licata. Running Circles Around (In) Your Proof Assistant; or, Quotients that Compute. post on the Homotopy Type Theory blog: https://homotopytypetheory.org/2011/04/23/running-circles-around-in-your-proof-assistant/. 2011.

[Lju23] Axel Ljungström. "A Cubical Formalisation of Cohomology Theory and $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$". Licentiate thesis. Stockholm University, May 2023.

[Lju24] Axel Ljungström. Symmetric monoidal smash products in homotopy type theory. *Mathematical Structures in Computer Science* (2024), pp. 1–23. DOI: 10.1017/S0960129524000318.

[LLM23] Thomas Lamiaux, Axel Ljungström and Anders Mörtberg. "Computing Cohomology Rings in Cubical Agda". *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs*. CPP 2023. Boston, MA, USA: Association for Computing Machinery, 2023, pp. 239–252. ISBN: 9798400700262. DOI: 10.1145/3573105.3575677.

[LM23] Axel Ljungström and Anders Mörtberg. "Formalizing $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$ and Computing a Brunerie Number in Cubical Agda ". *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. Los Alamitos, CA, USA: IEEE Computer Society, 2023, pp. 1–13. DOI: 10.1109/LICS56636.2023.10175833.

[LM24a] Axel Ljungström and Anders Mörtberg. Computational Synthetic Cohomology Theory in Homotopy Type Theory. To appear in *Mathematical Structures in Computer Science*. 2024. arXiv: 2401.16336.

[LM24b] Axel Ljungström and Anders Mörtberg. Formalising and Computing the Fourth Homotopy Group of the 3-Sphere in Cubical Agda. Preprint. 2024. arXiv: 2302.00151.

[Lod92] Jean-Louis Loday. "Cyclic Spaces and S1-Equivariant Homology". *Cyclic Homology*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 223–252. ISBN: 978-3-662-21739-9. DOI: 10.1007/978-3-662-21739-9_7.

[LP25] Axel Ljungström and Loïc Pujet. Cellular Methods in Homotopy Type Theory. Preprint. 2025. URL: https://aljungstrom.github.io/files/cellular2025.pdf.

[LS13] Daniel R. Licata and Michael Shulman. "Calculating the Fundamental Group of the Circle in Homotopy Type Theory". *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '13. New Orleans, LA, USA: IEEE Computer Society, 2013, pp. 223–232. ISBN: 978-0-7695-5020-6. DOI: 10.1109/LICS.2013.28.

[LS20] Peter LeFanu Lumsdaine and Michael Shulman. Semantics of higher inductive types. *Mathematical Proceedings of the Cambridge Philosophical Society* 169.1 (2020), pp. 159–208. DOI: 10.1017/S0305041190001 5X.

[LW25] Axel Ljungström and David Wärn. The Steenrod squares via unordered joins. To appear at the *40th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2025. URL: https://aljungstrom.github.io/files/steenrod2025.pdf.

[Mar82] Per Martin-Löf. "Constructive Mathematics and Computer Programming". *Logic, Methodology and Philosophy of Science VI*. Ed. by L. Jonathan Cohen et al. Vol. 104. Studies in Logic and the Foundations of Mathematics. Elsevier, 1982, pp. 153–175. DOI: https://doi.org/10.1016/S0049-237X(09)70189-2.

[Mar84] Per Martin-Löf. *Intuitionistic type theory. Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980*. Vol. 1. Studies in Proof Theory. Bibliopolis, 1984. ISBN: 88-7088-105-9.

[Mik91] Vladimir Voevodsky Mikhail Kapranov. ∞-groupoids and homotopy types. *Cahiers de Topologie et Géométrie Différentielle Catégoriques* 32.1 (1991), pp. 29–46.

[MM94] Saunders Mac Lane and Ieke Moerdijk. "Topoi and Logic". *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. New York, NY: Springer New York, 1994, pp. 267–346. ISBN: 978-1-4612-0927-0. DOI: 10.1007/978-1-4612-0927-0_8.

[Mou+15] Leonardo de Moura et al. "The Lean Theorem Prover (System Description)". *Automated Deduction – CADE-25*. Ed. by Amy P. Felty and Aart Middeldorp. Berlin, Germany: Springer International Publishing, 2015, pp. 378–388. ISBN: 978-3-319-21401-6. DOI: 10.1007/978-3-319-21401-6_26.

[NPS90] Bengt Nordström, Kent Petersson and Jan M. Smith. *Programming in Martin-Löf's type theory: an introduction*. USA: Clarendon Press, 1990. ISBN: 0198538146.

[Pea89] Giuseppe Peano. *Arithmetices Principia Novo Methodo Exposita*. lat. Augustae Taurinorum: Bocca, 1889. URL: http://eudml.org/doc/203509.

[PG24] Jonathan Prieto-Cubides and Håkon Robbestad Gylterud. On planarity of graphs in homotopy type theory. *Mathematical Structures in Computer Science* 34.4 (2024), pp. 281–321. DOI: 10.1017/S0960129524000100.

[Rij+] Egbert Rijke et al. *The agda-unimath library*. URL: https://github.com/UniMath/agda-unimath/.

[Rus03] Bertrand Russell. *Principles of Mathematics*. Appendix B: The Doctrine of Types. New York, Routledge, 1903.

[Sch24] M. Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische Annalen* 92.3 (Sept. 1924), pp. 305–316. ISSN: 1432-1807. DOI: 10.1007/BF01448013.

[Shu19] Michael Shulman. *All* $(\infty, 1)$*-toposes have strict univalent universes.* Preprint. Apr. 2019. arXiv: 1904.07004.

[Sim98] Carlos Simpson. Homotopy types of strict 3-groupoids. 1998. arXiv: math/9810059 [math.CT].

[Str93] Thomas Streicher. "Investigations Into Intensional Type Theory". Habilitation thesis. Ludwig-Maximilians-Universität München, 1993. URL: https://www2.mathematik.tu-darmstadt.de/~streicher/HabilStreicher.pdf.

[Swa22] Andrew W Swan. On the Nielsen-Schreier Theorem in Homotopy Type Theory. *Logical Methods in Computer Science* Volume 18, Issue 1 (Jan. 2022). ISSN: 1860-5974. DOI: 10.46298/lmcs-18(1:18)2022.

[UF13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics.* Institute for Advanced Study: Self-published, 2013. URL: https://homotopytypetheory.org/book/.

[VMA21] Andrea Vezzosi, Anders Mörtberg and Andreas Abel. Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types. *Journal of Functional Programming* 31 (2021), e8. DOI: 10.1017/S0956796821000034.

[Voe10] Vladimir Voevodsky. "Univalent Foundations Project". A modified version of an NSF grant application. Oct. 2010. URL: http://www.math.ias.edu/vladimir/files/univalent_foundations_project.pdf.

[Voe14] Vladimir Voevodsky. "Univalent Foundations". Talk at Institute for Advanced Study, Princeton. Mar. 2014. URL: https://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/2014_IAS.pdf.

[Wär23] David Wärn. Eilenberg–Maclane spaces and stabilisation in homotopy type theory. *Journal of Homotopy and Related Structures* 18.2 (Sept. 2023), pp. 357–368. ISSN: 1512-2891. DOI: 10.1007/s40062-023-00330-5.

[Wär24] David Wärn. Path spaces of pushouts. 2024. arXiv: 2402.12339.

[ZH24] Max Zeuner and Matthias Hutzler. "The Functor of Points Approach to Schemes in Cubical Agda". en. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. DOI: 10.4230/LIPICS.ITP.2024.38.

[ZM23] Max Zeuner and Anders Mörtberg. “A Univalent Formalization of Constructive Affine Schemes”. *28th International Conference on Types for Proofs and Programs (TYPES 2022)*. Ed. by Delia Kesner and Pierre-Marie Pédrot. Vol. 269. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 14:1–14:24. ISBN: 978-3-95977-285-3. DOI: 10.4230/LIPIcs.TYPES.2022.14.

Papers

Paper I

Mathematical Structures in Computer Science (2019), 1–00
doi:10.1017/xxxxx

CAMBRIDGE
UNIVERSITY PRESS

ARTICLE

Computational Synthetic Cohomology Theory in Homotopy Type Theory

Axel Ljungström, and Anders Mörtberg

Department of Mathematics, Stockholm University, Sweden axel.ljungstrom@math.su.se
Department of Mathematics, Stockholm University, Sweden anders.mortberg@math.su.se

(Received xx xxx xxx; revised xx xxx xxx; accepted xx xxx xxx)

Abstract

This paper discusses the development of synthetic cohomology in Homotopy Type Theory (HoTT), as well as its computer formalisation. The objectives of this paper are (1) to generalise previous work on integral cohomology in HoTT by the current authors and Brunerie (2022) to cohomology with arbitrary coefficients and (2) to provide the mathematical details of, as well as extend, results underpinning the computer formalisation of cohomology rings by the current authors and Lamiaux (2023). With respect to objective (1), we provide new direct definitions of the cohomology group operations and of the cup product, which, just as in (Brunerie et al., 2022), enable significant simplifications of many earlier proofs in synthetic cohomology theory. In particular, the new definition of the cup product allows us to give the first complete formalisation of the axioms needed to turn the cohomology groups into a graded commutative ring. We also establish that this cohomology theory satisfies the HoTT formulation of the Eilenberg–Steenrod axioms for cohomology and study the classical Mayer–Vietoris and Gysin sequences. With respect to objective (2), we characterise the cohomology groups and rings of various spaces, including the spheres, torus, Klein bottle, real/complex projective planes, and infinite real projective space. All results have been formalised in `Cubical Agda` and we obtain multiple new numbers, similar to the famous 'Brunerie number', which can be used as benchmarks for computational implementations of HoTT. Some of these numbers are infeasible to compute in `Cubical Agda` and hence provide new computational challenges and open problems which are much easier to define than the original Brunerie number.

Keywords: Synthetic Cohomology Theory, Synthetic Homotopy Theory, Homotopy Type Theory, Univalent Foundations, Constructive Mathematics.

Competing interests: The authors declare none.

Acknowledgements: The authors are grateful to Guillaume Brunerie and Thomas Lamiaux for collaborating with us and co-authoring earlier related work in (Brunerie et al., 2022) and (Lamiaux et al., 2023). We are also grateful to Evan Cavallo for insightful comments and many cubical tricks. The first author would also like to thank David Wärn for many fruitful discussions on related work which inspired the proof of Proposition 71. Finally, we wish to thank Dan Christensen for his helpful comments on an older version of this paper.

This paper is based upon research supported by the Swedish Research Council (Vetenskapsrådet) under Grant No. 2019-04545. The research has also received funding from the Knut and Alice Wallenberg Foundation through the Foundation's program for mathematics.

1. Introduction

A fundamental idea in algebraic topology is that spaces can be analysed in terms of homotopy invariants – functorial assignments of algebraic objects to spaces. The close correspondence between types and spaces in Homotopy Type Theory and Univalent Foundations (HoTT/UF) allows these concepts to be developed *synthetically* using type theory. This was explored in the HoTT Book (The Univalent Foundations Program, 2013), written during the IAS special year on HoTT/UF in 2012–2013, and has since led to the formalisation of many results from homotopy theory in HoTT/UF. Using these results, homotopy groups of many spaces – represented as types – have been characterised. However, just like in classical algebraic topology, these groups tend to be complicated to compute. Because of this, other topological invariants that are easier to compute, like cohomology, have also been developed synthetically in HoTT/UF.

Intuitively, the cohomology groups $H^n(X, G)$ of a space X, relative to an abelian group G, characterise the connected components of X as well as its $(n+1)$-dimensional holes. Figure 1 depicts the circle, $\mathbb{S}^1$, and two circles which have been glued together in a point – that is, it is the *wedge sum* $\mathbb{S}^1 \vee \mathbb{S}^1$. The fact that these spaces have a different number of holes is captured, using for instance singular cohomology, by the existence of isomorphisms $H^1(\mathbb{S}^1, G) \cong G$ and $H^1(\mathbb{S}^1 \vee \mathbb{S}^1, G) \cong G \times G$, which geometrically means that they have one, respectively two, 2-dimensional holes (that is, empty interiors). As cohomology groups are homotopy invariants, this then means that the spaces cannot be continuously deformed into each other.

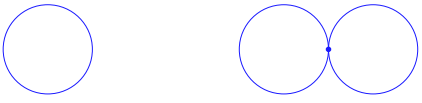


Figure 1: $\mathbb{S}^1$ and $\mathbb{S}^1 \vee \mathbb{S}^1$.

The usual formulation of singular cohomology using cochain complexes relies on taking the underlying set of topological spaces when defining the singular cochains (Hatcher, 2002). This operation is *not* invariant under homotopy equivalence, which makes it impossible to use when formalising cohomology synthetically. For the construction of cohomology in HoTT/UF, we instead rely on Brown representability (Brown, 1962) and define cohomology groups as homotopy classes of maps into Eilenberg–MacLane spaces (as defined in HoTT by Licata and Finster (2014)). This approach to synthetic cohomology theory was initially studied at the IAS special year (Shulman, 2013) and is classically provably equivalent to the singular cohomology of the spaces we consider in this paper. It has since been expanded with many classical results, for example the Eilenberg–Steenrod axioms for cohomology (Cavallo, 2015), cellular cohomology (Buchholtz and Hou (Favonia), 2018), the Atiyah-Hirzebruch and Serre spectral sequences (van Doorn, 2018), and it plays a key role in the synthetic proof that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$ by Brunerie (2016).

This paper develops the theory of synthetic cohomology from a computational perspective with the aim of characterising cohomology groups and rings of various spaces. This is achieved by extending earlier work by Brunerie et al. (2022) where integral cohomology $H^n(X, \mathbb{Z})$ was developed in `Cubical Agda`. Brunerie et al. (2022) also computed $\mathbb{Z}$-cohomology groups of various classical spaces synthetically, including the spheres, torus, real/complex projective planes, and Klein bottle. This was aided by a new synthetic construction of the group structure on $H^n(X, \mathbb{Z})$ with better computational properties than the one already defined in HoTT by Licata and Finster (2014). Brunerie et al. (2022) also redefined the cup product $\smile : H^n(X, \mathbb{Z}) \times H^m(X, \mathbb{Z}) \to H^{n+m}(X, \mathbb{Z})$ which led to substantially simplified proofs and better computational properties compared to the one originally defined synthetically by Brunerie (2016). Lamiaux et al. (2023) organised this into a ring $H^*(X, R)$, for a general ring R, and various such cohomology rings

were computed for different spaces. The authors have also used some of these results, and extensions thereof, in a complete `Cubical Agda` formalisation of Brunerie's synthetic proof that $\pi_4(\mathbb{S}^3) \simeq \mathbb{Z}/2\mathbb{Z}$ (Ljungström and Mörtberg, 2023). However, due to page constraints many proofs and constructions were omitted from (Brunerie et al., 2022), (Lamiaux et al., 2023), and (Ljungström and Mörtberg, 2023). The aim of the current paper is to spell these details out, develop new results, and generalise various results already present in the previous papers and the general HoTT literature. Hence, our aim is to provide a more comprehensive synthetic treatment of cohomology in HoTT/UF, with a view towards effective computations. We summarise the main contents and contributions as follows.

- We spell out the details of how to generalise the computationally well-behaved integral cohomology theory of Brunerie et al. (2022) to general cohomology groups $H^n(X, G)$. We first do this for Eilenberg–MacLane spaces in Section 3.1, which then induce the corresponding operations on cohomology in Section 4.
- In Section 3.2 we prove that $K(G, n) \simeq \Omega K(G, n+1)$ by a direct encode-decode proof, avoiding the use of the Freudenthal suspension theorem.
- In Sections 3.3 and 3.4 we spell out the details of how to generalise the theory of integral cohomology rings of Brunerie (2016) to general cohomology rings $H^*(X, R)$ as in (Lamiaux et al., 2023). In particular, this involves a general form of the cup product $\smile : H^n(X, G_1) \times H^m(X, G_2) \to H^{n+m}(X, G_1 \otimes G_2)$, inspired by a Book HoTT formalisation by Baumann (2018).
- To ensure that our definition of $H^n(X, G)$ is sensible, we verify that it satisfies the HoTT formulation of the Eilenberg–Steenrod axioms for cohomology in Section 4.2, which implies the existence of the Mayer–Vietoris sequence in Section 4.3, as originally shown in HoTT by Cavallo (2015).
- We generalise the synthetic definition of the Thom isomorphism and Gysin sequence of Brunerie (2016) from $H^n(X, \mathbb{Z})$ to cohomology with arbitrary commutative ring coefficients in Section 4.4.
- The cohomology theory is then used to compute various $H^n(X, G)$ and $H^*(X, R)$ for concrete values of X, G, and R in Section 5. We do this for the spaces studied by Brunerie et al. (2022) (spheres, torus, real/complex projective planes, and Klein bottle), generalising those cohomology group computations from $\mathbb{Z}$ to arbitrary G. We also provide details for how to carry out the cohomology ring computations of Lamiaux et al. (2023). Finally, we also have some new synthetic computations of these invariants for $\mathbb{R}P^\infty$.

The present paper is carefully written so that all proofs are constructive. Furthermore, all results have been formalised in `Cubical Agda` (Vezzosi et al., 2021) which has enabled us to do concrete computations with the operations defined in the paper. A summary file linking the paper to the formalised results can be found here: `www.github.com/agda/cubical/blob/master/Cubical/Papers/ComputationalSyntheticCohomology.agda`

In order to facilitate efficient computer computations we have been careful to give as direct constructions as possible. This allows us to define various new numbers similar to the famous 'Brunerie number' (Brunerie, 2016), but which are much simpler and still intractable to compute in `Cubical Agda`. In Section 6, we collect various open computational problems and benchmark challenges for `Cubical Agda` and related systems in which univalence and HITs have computational content. However, while everything has been formalised in `Cubical Agda`, we have been careful to write the paper in the informal style of the HoTT Book. This means that the results in the paper, except for those in Section 6, can all be interpreted in suitably structured ∞-topoi (Shulman, 2019), despite this not yet being known for results formalised in the particular cubical type theory of `Cubical Agda`.

While the paper is written in Book HoTT, we take some liberties and sometimes deviate from the notational conventions of the book. We detail this, as well some convenient cubical ideas that are useful also when working in Book HoTT, in Section 2.

2. Background on HoTT/UF and notations

We assume familiarity with the contents and notations of the HoTT Book (The Univalent Foundations Program, 2013), from here on referred to as (HoTT Book). In this section we briefly recall a few key definitions and introduce some of our notational conventions.

Definition 1 (Binary ap)**.** *Given a binary function $f : A \to B \to C$, we denote by ap^2_f the function*

$$(a_0 = a_1) \times (b_0 = b_1) \to f(a_0, b_0) = f(a_1, b_1)$$

The following path exists[a] *for each $p : a_0 = a_1$ and $q : b_0 = b_1$:*

$$\mathsf{ap}^2_f\text{–}\mathsf{funct}(p, q) : \mathsf{ap}^2_f(p, q) = \mathsf{ap}_{f(-,b_0)}(p) \cdot \mathsf{ap}_{f(a_1,-)}(q)$$

We define the *(homotopy) fibre* of a function $f : A \to B$ over a point $b : B$ as in (HoTT Book, Definition 4.2.4). That is, $\mathsf{fib}_f(b) := \Sigma_{a:A}(f\,a = b)$. We use the *contractible maps* definition of equivalences in (HoTT Book, Definition 4.4.1) and say that a function $f : A \to B$ is an *equivalence* if, for each $b : B$, we have that $\mathsf{fib}_f(b)$ is contractible (HoTT Book, Definition 3.11.1). We will simply say *proposition* for mere propositions (HoTT Book, Definition 3.1.1) and use *set* and *n-type* (with $n \geq -2$) as in (HoTT Book, Section 3.1 and 7).

A *pointed type* (HoTT Book, Definition 2.1.7) is a pair $(A, \star_A)$ where A is a type and $\star_A : A$ is a chosen basepoint of A. We often omit $\star_A$ and simply write A for the pointed type $(A, \star_A)$. We always take $\star_A$ to denote the basepoint of a pointed type A. Given two pointed types A and B, we denote by $A \to_\star B$ the type of *pointed functions* from A to B (HoTT Book, Definition 8.4.1). That is, it is the type of pairs (f, p) where $f : A \to B$ is a function and $p : f(\star_A) = \star_B$. We often simply write $f : A \to_\star B$ and take p implicit. If f further is an equivalence, we write $f : A \simeq_\star B$.

A class of pointed types particularly important for us is that of *H-spaces*. We borrow the definition (including notation) from Brunerie (2016, Definition 2.5.1).

Definition 2 (H-spaces)**.** *An H-space is a pointed type A equipped with a multiplication $\mu : A \times A \to A$ and homotopies*

$$\mu_\mathsf{l} : (a : A) \to \mu(\star_A, a) = a$$
$$\mu_\mathsf{r} : (a : A) \to \mu(a, \star_A) = a$$
$$\mu_\mathsf{lr} : \mu_\mathsf{l}(\star_A) = \mu_\mathsf{r}(\star_A)$$

The notion of an H-space is closely related to *homogeneous types*.

Definition 3 (Homogeneous types)**.** *A pointed type A is homogeneous if, for every $a : A$ there is a pointed equivalence $(A, \star_A) \simeq_\star (A, a)$.*

The loop space of a pointed type A is defined as $\Omega(A) := (\star_A = \star_A)$. Path composition defines an invertible H-space structure on $\Omega(A)$. Consequently, $\Omega(A)$ is also homogeneous.

[a] Note that we could have defined ap^2_f in terms of ap so that $\mathsf{ap}^2_f\text{–}\mathsf{funct}$ holds definitionally. We choose not to do this in order to stay neutral with respect to the flavour of HoTT our proof holds in – the usual definition of ap^2_f in cubical type theory does not make this kind of functoriality hold definitionally.

We will also rely on a few standard higher inductive types (HITs). For spheres, and many other constructions, we need suspensions. We use the standard definition from HoTT Book, Section 6.5, which we recall here:

Definition 4 (Suspensions)**.** *The suspension of a type A, denoted ΣA, is defined as a HIT with the following constructors:*

- $\mathsf{north}, \mathsf{south} : \Sigma A$
- $\mathsf{merid} : A \to \mathsf{north} = \mathsf{south}$

Using this, we define the n-spheres by:

$$\mathbb{S}^n = \begin{cases} \mathsf{Bool} & \text{when } n = 0 \\ \Sigma \mathbb{S}^{n-1} & \text{otherwise} \end{cases}$$

In fact, suspensions are just a special case of *(homotopy) pushouts*:

Definition 5 (Homotopy pushouts)**.** *Given a span of functions $A \xleftarrow{f} C \xrightarrow{g} B$, we define its (homotopy) pushout, denoted $A \sqcup^C B$, as a HIT with the following constructors:*

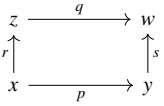
- $\mathsf{inl} : A \to A \sqcup^C B$
- $\mathsf{inr} : B \to A \sqcup^C B$
- $\mathsf{push} : (c : C) \to \mathsf{inl}\,(f(c)) = \mathsf{inr}\,(g(c))$

We will also make use of n-truncations. These are defined using the 'hub and spoke' definition of (HoTT Book, Section 7.3) and, as usual, we denote the n-truncation of a type A by $\|A\|_n$ and write $|\,a\,| : \|A\|_n$ for its canonical elements. Following (HoTT Book, Definition 7.5.1), we say that a type A is *n-connected* if $\|A\|_n$ is contractible and a function $f : A \to B$ is said to be *n-connected* if the fibre of f over any $b : B$ is n-connected.

2.1 Dependent paths and cubical thinking

While this paper is written in Book HoTT, it is still often helpful to use ideas from cubical type theory and 'think cubically'. One reason for this is that iterated path types are conveniently represented by higher cubes. This cubical approach to Book HoTT was explored in depth by Licata and Brunerie (2015) and we will here outline some cubical ideas relevant to this paper.

We will often speak of *dependent* path types – in particular squares of paths. Given two paths $p : x = y$ and $q : z = w$, we cannot ask whether $p = q$, since this is not well-typed. We could, however, ask whether p and q are equal *modulo* composition with some other paths $r : x = z$ and $s : y = w$. We say that we ask for a *filler* of the square

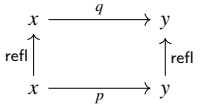


by which we mean a proof of the identity

$$\mathsf{transport}^{X \mapsto X}\left(\mathsf{ap}^2_{=}(r, s),\, p\right) = q$$

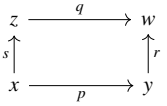
(or, equivalently, $r^{-1} \cdot p \cdot s = q$, or $\mathsf{Square}\; r\; p\; q\; s$ using the notation of Licata and Brunerie (2014, Section IV.C). In cubical type theory, this would simply amount to providing a term of type

$\mathsf{PathP}\ (\lambda i.\, r\, i = s\, i)\ p\ q$. We note also that the type of filler of a degenerate square

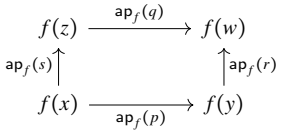


precisely coincides with the type $p = q$.

As with regular paths, we may apply functions also to squares. For instance, given a function $f : A \to B$ and a filler sq of the following square in A:

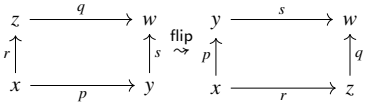


we get the following filler in B:

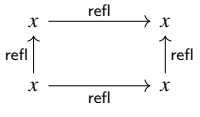


We will, with some minor abuse of notation[b], denote this filler by $\mathsf{ap}_{\mathsf{ap}_f}(\mathsf{sq})$

Let us give another example of manipulations of squares which will play a crucial roll in Section 5. There is a map which takes a square filler and returns a filler of the same square, flipped along its diagonal:



The map is easily defined by path induction. In cubical type theory, flip simply takes a square $I^2 \to A$ and flips the order of the arguments. This operation is homotopically non-trivial in a crucial sense. Consider the type of fillers of the square

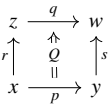


This is precisely the path type $\mathsf{refl}_x = \mathsf{refl}_x$ or, in other words, $\Omega^2(A, x)$. Since all sides of the square are the same, flip defines an endofunction $\Omega^2(A, x) \to \Omega^2(A, x)$. One might suspect that it reduces to the identity in this degenerate setting, but this is actually not the case. Rather, we get the following:

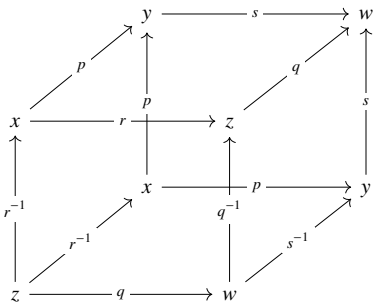
Lemma 6. *Given $p : \Omega^2(A, x)$, we have $\mathsf{flip}(p) = p^{-1}$.*

[b]Technically, the outer ap should be dependent function application, apd, if we were to follow HoTT Book notation. We will not distinguish the two and use ap for both.

Proof. Let Q be the following filler.



Consider the following cube:



Let its bottom be filled by Q^{-1} and its top be filled by $\mathsf{flip}(Q)$. All sides have their obvious fillers defined by path induction on p, q, r and s. We claim that the cube has a filler. We prove this by first applying path induction on p,r and s, transforming Q to a path $\mathsf{refl}_x = q$. Finally, after applying path induction on Q we are left to fill a cube with $\mathsf{refl}_{\mathsf{refl}_x}$ on each side, which is trivial.

Specialising the above argument to the case when p, q, r and s are all refl_x and Q is arbitrary, we have our lemma. □

3. Eilenberg–MacLane spaces

In order to define representable cohomology in HoTT, we will need to define Eilenberg–MacLane spaces. These are spaces $K(G, n)$ associated to an abelian[c] group G and a natural number n such that $\pi_n(K(G, n)) \simeq G$ and all other homotopy groups vanish. These spaces carry the structure of an H-space which we will later see induces the group structure on cohomology. We will also see that they come equipped with a graded multiplication which, also in Section 4, will be lifted to define cohomology rings.

The definition of Eilenberg–MacLane spaces and their H-space structure in HoTT is due to Licata and Finster (2014). The special case of integral Eilenberg–MacLane spaces, $K(\mathbb{Z}, n)$, was considered by Brunerie (2016), who gave an alternative and very compact definition in terms of truncated spheres. This definition of $K(\mathbb{Z}, n)$ was also considered by Brunerie et al. (2022). In this section, we will generalise the optimised proofs regarding integral Eilenberg–MacLane spaces and their H-space structure found in (Brunerie et al., 2022) to a definition of general Eilenberg–MacLane spaces following Licata and Finster (2014).

In order to ease the notation when describing the induction principles of $K(G, n)$, we will define it one step at a time. The crucial step is $K(G, 1)$, which will be defined in terms of the following construction.

Definition 7. *Given a type A, we define $L(A)$ by the HIT*

[c]Technically, this is only needed when $n > 1$, but in this paper we will, for simplicity, always assume G to be abelian.

- $\star : L(A)$
- $\mathsf{loop}_k : A \to \star = \star$

Given a group G, the type $L(G)$ is *almost* the first Eilenberg–MacLane space of G. By adding a constructor connecting the group structure on G with the H-space structure on $\Omega(L(G))$, we approximate it further.

Definition 8 (Raw Eilenberg–MacLane spaces)**.** *We define the first raw Eilenberg–MacLane space, denoted $\widetilde{K}(G,1)$, by the following HIT*

- $\iota : L(A) \to \widetilde{K}(G,1)$
- *For every $g_1, g_2 : G$, a filler $\mathsf{sq}(g_1, g_2)$ of the square*

$$\begin{array}{ccc} \iota(\star) & \xrightarrow{\mathsf{ap}_\iota(\mathsf{loop}_k\ (g_1 + g_2))} & \iota(\star) \\ {\scriptstyle \mathsf{refl}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_\iota(\mathsf{loop}_k\ g_2)} \\ \iota(\star) & \xrightarrow[\mathsf{ap}_\iota(\mathsf{loop}_k\ g_1)]{} & \iota(\star) \end{array}$$

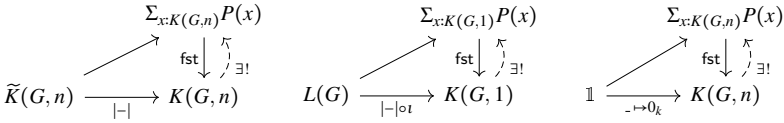
For $n > 1$, we define $\widetilde{K}(G,n) := \Sigma(\widetilde{K}(G,n-1))$.

In practice, we will omit the ι and simply write, for example, $\star : \widetilde{K}(G,1)$ and $\mathsf{loop}_k\, g : \Omega(\widetilde{K}(G,1))$. We let 0_k denote the basepoint of $\widetilde{K}(G,n)$. That is, $0_k := \star$ when $n = 1$ and $0_k := \mathsf{north}$ when $n > 1$.

Definition 9 (Eilenberg–MacLane spaces)**.** *Given an integer $n \geq 1$ and an abelian group G, we define the nth Eilenberg–MacLane space of G by $K(G,n) := \|\widetilde{K}(G,n)\|_n$. The zeroth Eilenberg–MacLane space is simply $K(G,0) := G$.*

The type $K(G,n)$ is pointed by 0_G when $n = 0$ and $|\,0_k\,|$ when $n \geq 1$. With some abuse of notation, we will simply write 0_k to denote the basepoint of $K(G,n)$.

Eilenberg–MacLane spaces come with important elimination principles. Let $n \geq 1$, the fundamental elimination principle of $K(G,n)$ is given in Figure 2a and says that, given a fibration of n-types $P : K(G,n) \to n\text{-}\mathsf{Type}$, in order to define a section $(x : K(G,n)) \to P(x)$, it suffices to describe its action on canonical elements $|\,x\,| : K(G,n)$, where $x : \widetilde{K}(G,n)$. If P is set-valued and $n = 1$, it suffices to define the section for elements in $L(G)$ – see Figure 2b. In other words, we do not need to prove that our construction respects the sq-constructor. Finally, it is easy to see that if P is a family of $(n-2)$-types, then any section $(x : K(G,n)) \to P(x)$ is uniquely determined by its action on the basepoint $0_k : K(G,n)$ – see Figure 2c.



(a) n-type elimination (b) 0-type elimination for $K(G,1)$ (c) $(n-2)$-type elimination

Figure 2: Elimination principles for $K(G,n)$

We will, in Section 3.2, see that $\Omega^n K(G,n) \simeq G$ but, before this, let us establish that $K(G,n)$ is $(n-1)$-connected. These two facts imply that $\pi_n(K(G,n)) \simeq G$ and that all other homotopy groups vanish, which therefore shows that $K(G,n)$ indeed is an Eilenberg–MacLane space.

Proposition 10. *The type $K(G,n)$ is $(n-1)$-connected.*

Proof. We want to show that $\|K(G,n)\|_{n-1}$ is contractible. When $n=0$, the statement is trivial, since $\|A\|_{-1}$ is contractible for any pointed type A. Let $n \geq 1$, we choose the centre of contraction to be $|\,0_k\,|$. We now want to construct a section $(x : \|K(G,n)\|_{n-1}) \to |\,0_k\,| = x$. Since $\|K(G,n)\|_{n-1}$ is an $(n-1)$-type, all path types over it are $(n-2)$-types. Thus, it suffices to construct the section over $|\,0_k\,|$, which is trivial by $\mathsf{refl} : |\,0_k\,| = |\,0_k\,|$. □

The type $K(G,n)$ is functorial in G. That is, for any homomorphism $\phi : G_1 \to G_2$, we have an induced map $\phi_n : K(G_1,n) \to K(G_2,n)$. We define this as a map $\widetilde{K}(G_1,n) \to \widetilde{K}(G_2,n)$. When $n=0$, we set $\phi_0 :\equiv \phi$. For $n=1$, ϕ_1 is defined by

$$\begin{aligned} \phi_1(\star) &:\equiv \star \\ \mathsf{ap}_{\phi_1}(\mathsf{loop}_k\, g) &:\equiv \mathsf{loop}_k\,(\phi_0(g)) \end{aligned}$$

This respects the sq constructor of $\widetilde{K}(G_1,n)$ since ϕ is assumed to be a homomorphism. For $n \geq 2$, we define it recursively using functoriality of Σ and the fact that $\widetilde{K}(G,n) :\equiv \Sigma(\widetilde{K}(G,n-1))$:

$$\begin{array}{ccc} \widetilde{K}(G_1,n) & \xrightarrow{\phi_n} & \widetilde{K}(G_2,n) \\ \| & & \| \\ \Sigma(\widetilde{K}(G_1,n-1)) & \xrightarrow{\Sigma(\phi_{n-1})} & \Sigma(\widetilde{K}(G_2,n-1)) \end{array}$$

We also remark that there is a map $\sigma_n : K(G,n) \to \Omega(K(G,n+1))$. We define it for raw Eilenberg–MacLane spaces by

$$\sigma_n(x) :\equiv \begin{cases} \mathsf{loop}_k\, g & \text{if } n=0 \\ \mathsf{merid}\, x \cdot (\mathsf{merid}\, 0_k)^{-1} & \text{otherwise} \end{cases}$$

As $\Omega(K(G,n+1))$ is an n-type, this definition also induces, via n-type elimination for Eilenberg–MacLane spaces (see Figure 2a), a map $K(G,n) \to \Omega(K(G,n+1))$:

$$\begin{array}{ccccc} \widetilde{K}(G,n) & \xrightarrow{\sigma_n} & \Omega(\widetilde{K}(G,n+1)) & \xrightarrow{\mathsf{ap}_{|-|}} & \Omega(K(G,n+1)) \\ \downarrow & & & \nearrow & \\ K(G,n) & & & & \end{array}$$

With some abuse of notation, we also denote this map by σ_n. The following fact follows immediately by construction of σ_n.

Lemma 11. *Given a group homomorphism $\phi : G_1 \to G_2$, we have*

$$\mathsf{ap}_{\phi_{n+1}}(\sigma_n(x)) =_{\Omega(K(G_2,n))} \sigma_n(\phi_n(x))$$

for $x : K(G_1,n)$.

3.1 Group structure

One of the key contributions of Brunerie et al. (2022) was the computationally optimised definition of the H-space structure on $K(\mathbb{Z},n) := \|\mathbb{S}^n\|_n$ (when $n \geq 1$). Our goal here is to rephrase this optimised construction to $K(G,n)$ defined as above. The following proposition is crucial. It is a special case of (HoTT Book, Lemma 8.6.2.) for which we provide a direct proof in order to make future constructions relying on it more computationally efficient.

Proposition 12. *Let $n,m \geq 1$ and assume we are given a fibration $\widetilde{K}(G_1,n) \times \widetilde{K}(G_2,m) \to (n+m-2)$-Type, sections $f : (x : \widetilde{K}(G_1,n)) \to P(x,0_k)$ and $g : (y : \widetilde{K}(G_2,m)) \to P(0_k,y)$, and a path $p : f(0_k) = g(0_k)$. In this case, there is a unique section*

$$f \veebar g : ((x,y) : \widetilde{K}(G_1,n) \times \widetilde{K}(G_2,m)) \to P(x,y)$$

equipped with homotopies

$$l : (x : \widetilde{K}(G_1,n)) \to (f \veebar g)(x,0_k) = f(x)$$
$$r : (y : \widetilde{K}(G_2,m)) \to (f \veebar g)(0_k,y) = g(y)$$

satisfying $l(0_k)^{-1} \cdot r(0_k) = p$.

Proof. An easy way of proving this is to note that the canonical map

$$i_\vee : \widetilde{K}(G_1,n) \vee \widetilde{K}(G_2,m) \to \widetilde{K}(G_1,n) \times \widetilde{K}(G_2,m)$$

is $((n-1)+(m-1))$-connected, thereby inducing a section via the map

$$f \vee g : (x : \widetilde{K}(G_1,n) \vee \widetilde{K}(G_1,m)) \to P(i_\vee(x))$$

From the point of view of computer formalisation, however, this proof leads to poor computational behaviour of subsequent definitions. For this reason, we provide a direct proof.

We proceed by induction on n and m. We first consider the base-case: $n = m = 1$. We are to construct a section $f \veebar g : ((x,y) : \widetilde{K}(G_1,1) \times \widetilde{K}(G_2,1)) \to P(x,y)$. Since P is a set, it suffices to define a section $f \veebar g : ((x,y) : L(G_1) \times L(G_2)) \to P(x,y)$. We proceed by $L(G_1)$-induction on x. For the point constructor, we define

$$(f \veebar g)(\star, y) := g(y)$$

We now need to define $\mathsf{ap}_{(f \veebar g)(-,y)}(\mathsf{loop}_k\, a)$ for $a : G_1$. Since P is set valued, it suffices to do so when y is $\star$. We define

$$\mathsf{ap}_{(f \veebar g)(-,\star)}(\mathsf{loop}_k\, a) := p^{-1} \cdot \mathsf{ap}_f(\mathsf{loop}_k\, a) \cdot p$$

We note that this construction satisfies the additional properties by default: we construct $l : (x : \widetilde{K}(G_1,1)) \to (f \veebar g)(x,0_k) = f(x)$ by induction on x. Since this is a proposition, it suffices to construct $l(0_k) : g(0_k) = f(0_k)$, which we do by $l(0_k) := p^{-1}$. We construct $r : (y : \widetilde{K}(G_2,1)) \to (f \veebar g)(0_k,x)$ simply by $r(y) := \mathsf{refl}$, since the equality holds by definition. We finally need to show that $l(0_k)^{-1} \cdot r(0_k) = p$ which is immediate by construction of l and r.

We proceed by induction on n, letting $n \geq 2$ and omitting the case when $n = 1$ and $m \geq 2$ which is entirely symmetric. Let us define $f \veebar g : ((x,y) : \widetilde{K}(G_1,n) \times \widetilde{K}(G_2,m)) \to P(x,y)$. We stress that since $n \geq 2$, we have that $\widetilde{K}(G_1,n) = \Sigma\widetilde{K}(G_1,n-1)$ by definition. Hence, we may define $f \veebar g$ by suspension induction on x. Let us start with the action on point constructors.

$$(f \veebar g)(\mathsf{north}, y) := g(y)$$
$$(f \veebar g)(\mathsf{south}, y) := \mathsf{transport}^{P(-,y)}(\mathsf{merid}\, 0_k, g(y))$$

We now need to define $\mathsf{ap}_{f \veebar g}(-,y)(\mathsf{merid}\, a) : g(y) =^{P(-,y)}_{\mathsf{merid}\, a} \mathsf{transport}^{P(-,y)}(\mathsf{merid}\, 0_k, g(y))$ for $a : \widetilde{K}(G_1,n-1)$ and $y : \widetilde{K}(G_2,m)$. The type of $\mathsf{ap}_{f \veebar g}(-,y)(\mathsf{merid}\, a)$ is an $(n+m-3)$-type, and

thus, by induction hypothesis, it suffices to construct the following:

$$f' : (a : \widetilde{K}(G_1, n-1)) \to g(0_k) =^{P(-,0_k)}_{\mathsf{merid}\, a} \mathsf{transport}^{P(-,0_k)}(\mathsf{merid}\, 0_k, g(0_k))$$
$$g' : (y : \widetilde{K}(G_2, m)) \to g(y) =^{P(-,y)}_{\mathsf{merid}\, 0_k} \mathsf{transport}^{P(-,y)}(\mathsf{merid}\, 0_k, g(y))$$
$$p' : f'(0_k) = g'(0_k)$$

By composition with p, the target type of f' is equivalent to $f(\mathsf{north}) =^{P(-,0_k)}_{\mathsf{merid}\, a} f(\mathsf{south})$, of which we have a term: $\mathsf{ap}_f(\mathsf{merid}\, a)$. The target type of g' is equivalent to $g(y) = g(y)$ and we simply give the term refl. The construction of p' is an easy but technical lemma. Thus we have constructed $f \triangledown g$. Now note that $r : (y : \widetilde{K}(G_2, m)) \to (f \triangledown g)(0_k, y) = g(y)$ holds by refl. The path $l : (x : \widetilde{K}(G_1, n-1)) \to (f \triangledown g)(x, 0_k) = g(y)$ is easily constructed using the corresponding l- and r-paths from the inductive hypothesis. This can be done so that $l(0_k) = r(0_k)$ holds almost by definition. □

Our goal now is to define addition $+_k : K(G,n) \times K(G,n) \to K(G,n)$. When $n = 0$, it is simply addition in G. For $n \geq 1$, we note that, by the elimination principle for $K(G,n)$ in Figure 2a, it suffices to define a corresponding operation $+_k : \widetilde{K}(G,n) \times \widetilde{K}(G,n) \to K(G,n)$. When $n = 1$, we define it explicitly by

$$\star +_k y := y$$
$$\mathsf{ap}_{\_ +_k \star}(\mathsf{loop}_k\, g) := \mathsf{loop}_k\, g$$
$$\mathsf{ap}_{x \mapsto \to \mathsf{ap}_{\_ +_k x}(\mathsf{loop}_k\, g_1)}(\mathsf{loop}_k\, g_2) := Q$$

where Q is a filler of the following square

$$\begin{array}{ccc} \star & \xrightarrow{\mathsf{loop}_k\, g_1} & \star \\ {\scriptstyle \mathsf{loop}_k\, g_2}\uparrow & & \uparrow{\scriptstyle \mathsf{loop}_k\, g_2} \\ \star & \xrightarrow[\mathsf{loop}_k\, g_1]{} & \star \end{array}$$

which is easily constructed using that loop_k preserves composition by definition of $K(G,1)$. All other cases are immediate since $K(G,1)$ is 1-truncated. For higher values of n, the construction is made straightforward by Proposition 12: Indeed, in this case, we have that $K(G,n)$ is of h-level $n \leq n + n - 2$. Therefore, we may define $+_k$ simply by

$$x +_k 0_k := x \tag{1}$$
$$0_k +_k y := y \tag{2}$$

Proposition 12 also requires us to prove the above definitions to coincide when $x = 0_k$ and $y = 0_k$, which they do by refl. Thus, we have defined $+_k : \widetilde{K}(G,n) \times \widetilde{K}(G,n) \to K(G,n)$, which lifts to an addition $K(G,n) \times K(G,n) \to K(G,n)$ which we, abusing notation, will also denote by $+_k$.

The laws governing $+_k$ are remarkably easy to prove.

Proposition 13. *The addition $+_k$ has 0_k as a left- and right-unit. Furthermore, these are coherent in the sense that we have a filler of the following square*

$$\begin{array}{ccc} 0_k + 0_k & \xrightarrow{\mathsf{r\text{-}unit}(0_k)} & 0_k \\ \| & & \| \\ 0_k + 0_k & \xrightarrow[\mathsf{l\text{-}unit}(0_k)]{} & 0_k \end{array}$$

Proof. The statement follows from the group structure on G when $n = 0$. When $n = 1$ the target type is a set, since $K(G,1)$ is 1-truncated (and hence path types over it are sets). By the set-elimination

rule for $K(G,n)$ (see Figure 2b), it suffices to show that $|x|_k +_k 0_k = 0_k +_k |x|$ for $x : L(G)$. These equalities hold definitionally by $L(G)$ induction. Hence, we also get the filler of the square by refl. When $n \geq 2$, the statement follows immediately by the defining equations (1) and (2), using the family of paths l and r and the coherence between them, as given in Proposition 12. □

In fact, modulo $\mathbb{N}$-induction, by the direct proof of Proposition 12, we get that the left-and right-unit laws are definitionally equal at 0_k, with both definitionally equal to refl.

Proposition 14. *The addition $+_k$ is commutative.*

Proof. When $n = 0$, the statement is simply that G is abelian, which we have assumed. Let $n \geq 1$. By truncation elimination, it suffices to construct paths $|x|_k +_k |y| = |y| +_k |x|$ for $x, y : \widetilde{K}(G,n)$. This path type is $(n-1)$-truncated which is sufficiently low for Proposition 12 to apply. Hence it suffices to construct paths $p_x : |x| + 0_k = 0_k + |x|$ and $q_y : 0_k +_k |y| = |y| + 0_k$ and show that $p_{0_k} = q_{0_k}$. We construct both paths as compositions of the left- and right-unit laws of $+_k$ in the obvious way. These definitions coincide over 0_k by the second part of Proposition 13. □

The following is equally direct to prove (using Proposition 12), so we omit its proof.

Proposition 15. *The addition $+_k$ is associative.*

Propositions 13, 14 and 15 amount precisely to showing that $K(G,n)$ is a commutative, associative H-space. There is, however, more to the story. In HoTT, it is easy to define an inversion on $K(G,n)$. When $n = 0$, it is simply negation in G. When $n \geq 1$, we have two different options. One approach is to note that for any $x : K(G,n)$, the map $y \mapsto x + y$ is an equivalence. This is proved by observing that the property of being an equivalence is a proposition. The elimination rule of $K(G,n)$ into $(n-2)$-types tells us that it is enough to show that the map is an equivalence when x is 0_k. In this case, the map is simply the identity, and thus also an equivalence. The unique element in the fibre of $y \mapsto x + y$ over 0_k is taken to be the inverse of x. This proof also gives the desired cancellations laws by construction. This is the approach used in, for instance, (Brunerie, 2016; Licata and Finster, 2014).

While elegant, the above approach has one major disadvantage: the action of $-_k$ on any element other than 0_k is described using transports. One consequence of this is that the terms produced by $-_k$ often are hard to work with directly. Another consequence is that terms described using $-_k$ tend to explode in size, which leads to poor computational behaviour in implementations in proof assistants. For these reasons, we choose the obvious direct construction of $-_k$. We define it as a map $-_k : \widetilde{K}(G,n) \to \widetilde{K}(G,n)$. When $n = 1$, we define

$$\begin{aligned} -_k \star &:= \star \\ \mathsf{ap}_{-_k}(\mathsf{loop}_k\, g) &:= (\mathsf{loop}_k\, g)^{-1} \\ \mathsf{ap}_{\mathsf{ap}_{-_k}}(\mathsf{sq}\, g_1\, g_2) &:= Q \end{aligned}$$

where Q is a filler of the square

$$\begin{array}{ccc} \star & \xrightarrow{\mathsf{loop}_k\,(g_1+g_2)^{-1}} & \star \\ \| & & \uparrow{\scriptstyle (\mathsf{loop}_k\, g_2)^{-1}} \\ \star & \xrightarrow[(\mathsf{loop}_k\, g_1)^{-1}]{} & \star \end{array}$$

which is easy to construct using the functoriality of loop_k . For $n \geq 2$, we define it by

$$-_k \,\mathsf{north} := \mathsf{north}$$
$$-_k \,\mathsf{south} := \mathsf{north}$$
$$\mathsf{ap}_{-_k}(\mathsf{merid}\, g) := \sigma_n(g)^{-1}$$

For the cancellation laws, we will need the following lemma. We remark that it is only is well-typed in the case when $n = 1$ is fixed or when n is on the form $(2 + m)$. The fact that it is well-typed exploits the fact that $0_k = 0_k +_k 0_k$ holds definitionally modulo the case distinction on n.

Proposition 16. *Let $n \geq 1$. Given $p, q : \Omega(K(G, n))$, we have $\mathsf{ap}^2_{+_k}(p, q) = p \cdot q$.*

Proof. We have

$$\mathsf{ap}^2_{+_k}(p, q) = \mathsf{ap}_{x \mapsto x +_k 0_k}(p) \cdot \mathsf{ap}_{y \mapsto 0_k +_k y}(q) = p \cdot q$$

The final step consists in applying the right- and left-unit laws to both components. This is justified since they agree at 0_k. □

We write $x -_k y$ for $x +_k (-_k y)$ and have that the directly defined $-_k$ indeed is inverse to $+_k$ by the following proposition:

Proposition 17. *For $x : K(G, n)$, we have $x -_k x = 0_k = (-_k x) +_k x$.*

Proof. By commutativity of $+_k$, it it suffices to show that $x -_k x = 0_k$. When $n = 0$, this comes from the group structure on G. When $n = 1$, we note that the goal type is a set, which means that it suffices to show the statement for $x : L(G)$. We have $\star -_k \star = 0_k$ by refl. For loop_k , we need to show that

$$\mathsf{ap}^2_{x,y \mapsto x -_k y}(\mathsf{loop}_k\, g, \mathsf{loop}_k\, g) = \mathsf{refl}$$

By Proposition 16, the LHS above is equal to $\mathsf{loop}_k\, g \cdot (\mathsf{loop}_k\, g)^{-1}$ which is equal to refl. The case when $n \geq 2$, is entirely analogous, again using Proposition 16 for the path constructors. □

And there we have it: the 3-tuple $(K(G, n), +_k, 0_k)$ forms, for any n, a commutative and associative H-space with inverse given by $-_k$. We will see in Section 4 that this trivially gives an induced group structure on cohomology groups, but let us first take a closer look at σ_n and also examine the multiplicative structure on $K(G, n)$.

3.2 $K(G, n)$ ***vs.*** $\Omega K(G, n + 1)$

We will soon see that $\sigma_n : K(G, n) \to \Omega K(G, n + 1)$ in fact is an equivalence. It will be crucial when computing cohomology groups of spaces. However, we do not need it yet: Proposition 16 already provides us with a strong link between the structures on $K(G, n)$ and $\Omega K(G, n + 1)$. For instance, we may already now deduce the commutativity of $\Omega K(G, n + 1)$ from the commutativity of $+_k$:

Proposition 18. *For $x : K(G, n)$ and $p, q : x = x$, we have $p \cdot q = q \cdot p$.*

Proof. When $n = 0$, we are automatically done, since G is a set. For $n \geq 1$, it suffices, by $(n - 2)$-type elimination, to show the statement when x is 0_k. Let $p, q : \Omega K(G, n)$. We have

$$p \cdot q = \mathsf{ap}^2_{+_k}(p, q) = \mathsf{ap}^2_{+_k}(q, p) = q \cdot p$$

where the second equality comes from the commutativity of $+_k$. □

The usefulness of the above proof (as opposed to one using an equivalence $K(G,n) \simeq \Omega K(G,n+1)$ as in (Brunerie, 2016)) is that it is rather direct and is easily shown to be homotopically trivial when p and q are refl.

On a similar note, we may also provide a direct proof of the following proposition.

Proposition 19. *Let $n \geq 1$ and $p : \Omega K(G,n)$, we have $\mathsf{ap}_{-_k}(p) = p^{-1}$.*

Proof. We give the proof for $n = 1$ as the case when $n > 1$ is entirely analogous. Let us define a fibration $\mathsf{Code}_x : (p : \star = x) \to \mathsf{hProp}$ for $x : K(G,1)$. As hProp is a set, it suffices to define $\mathsf{Code}_x(p)$ for $x : L(G)$. We do this by induction on x and define

$$\mathsf{Code}_\star(p) := (\mathsf{ap}_{-_k}(p) = p^{-1})$$

which is a proposition since $K(G,1)$ is 0-connected. We need to check that this definition respects the action of Code on $\mathsf{loop}_k\, g$ for $g : G$. This amounts to showing that, for any $p : \Omega K(G,1)$, we have an equivalence

$$(\mathsf{ap}_{-_k}(p) = p^{-1}) \simeq (\mathsf{ap}_{-_k}(p \cdot \mathsf{loop}_k\, g) = (p \cdot \mathsf{loop}_k\, g)^{-1})$$

We rewrite the terms on the RHS as follows:

$$\begin{aligned} \mathsf{ap}_{-_k}(p \cdot \mathsf{loop}_k\, g) &= \mathsf{ap}_{-_k}(p) \cdot \mathsf{ap}_{-_k}(\mathsf{loop}_k\, g) \\ &= \mathsf{ap}_{-_k}(p) \cdot (\mathsf{loop}_k\, g)^{-1} \end{aligned}$$

and

$$\begin{aligned} (p \cdot \mathsf{loop}_k\, g)^{-1} &= (\mathsf{loop}_k\, g)^{-1} \cdot p^{-1} \\ &= p^{-1} \cdot (\mathsf{loop}_k\, g)^{-1} \end{aligned}$$

Thus, we only need to construct an equivalence.

$$(\mathsf{ap}_{-_k}(p) = p^{-1}) \simeq (\mathsf{ap}_{-_k}(p) \cdot (\mathsf{loop}_k\, g)^{-1} = p^{-1} \cdot (\mathsf{loop}_k\, g)^{-1})$$

This equivalence is given by $\mathsf{ap}_{q \mapsto q \cdot (\mathsf{loop}_k\, g)^{-1}}$ and thereby the construction of Code is complete.

We now note that we have a section $(p : \star = x) \to \mathsf{Code}_x(p)$ for any $x : K(G,1)$ since, by path induction, it suffices to note that refl is an element of $\mathsf{Code}_\star(\mathsf{refl})$. Hence, in particular, we have an element of $\mathsf{Code}_\star(p)$ for all $p : \Omega K(G,1)$, which is what we wanted to show. □

The following proposition is also crucial and its proof can be found in (Licata and Finster, 2014, Lemma 4.4).

Proposition 20. *For $x, y : K(G,n)$, we have $\sigma_n(x +_k y) = \sigma_n(x) \cdot \sigma_n(y)$.*

Because σ_n is pointed, Proposition 20 immediately gives us the following corollary (which is proved just like in group theory).

Corollary 21. *For $x : K(G,n)$, we have $\sigma_n(-_k x) = \sigma_n(x)^{-1}$.*

We are now ready for the main result of the section, namely that $\Omega K(G,n+1) \simeq K(G,n)$. The original proof in HoTT is due to Licata and Finster (2014, Theorem 4.3) who used encode-decode proofs in the cases $n = 0$ and $n = 1$, and the Freudenthal suspension theorem (formalised in HoTT by Hou (Favonia) et al. (2016)) in the case $n \geq 2$. Here we use an identical proof in the case $n = 1$. However, it is possible to avoid the Freudenthal suspension theorem completely by simply noting that the proof of Licata and Finster when $n = 1$ actually can be used for all $n \geq 1$. We primarily wish to avoid the Freudenthal suspension theorem as circumventing it makes the computer formalisation more computationally viable.

Theorem 22. *The map* $\sigma_n : K(G,n) \to \Omega K(G,n+1)$ *is an equivalence.*

Proof. For $n = 0$, we refer to the proof in (Licata and Finster, 2014). For $n \geq 1$, we choose to provide an encode-decode proof similar to their $n = 1$ case. We define $\mathsf{Code} : K(G,n+1) \to n\text{-}\mathsf{Type}$ by

$$\begin{aligned}
\mathsf{Code}(|\,\mathsf{north}\,|) &:= K(G,n)\\
\mathsf{Code}(|\,\mathsf{south}\,|) &:= K(G,n)\\
\mathsf{ap}_{\mathsf{Code}\circ|-|}(\mathsf{merid}\ a) &:= \mathsf{ua}(x \mapsto |\,a\,|_k + x)
\end{aligned}$$

Here ua is the part of univalence which turns an equivalence into a path. Clearly the map to which it is applied is an equivalence. For any $t : K(G,n+1)$, we have a map $\mathsf{encode}_t : 0_k = t \to \mathsf{Code}(t)$ defined by $\mathsf{encode}_t(p) := \mathsf{transport}^{\mathsf{Code}}(p, 0_k)$ (this is equivalent to path induction on p). We define $\mathsf{decode}_t : \mathsf{Code}(t) \to 0_k = t$ by truncation elimination and $K(G,n)$-induction on t. For the point constructors, we define it by

$$\begin{aligned}
\mathsf{decode}_{|\,\mathsf{north}\,|}(x) &= \sigma_n(x)\\
\mathsf{decode}_{|\,\mathsf{south}\,|}(x) &= \sigma_n(x) \cdot \mathsf{ap}_{|-|}(\mathsf{merid}\ 0_k)
\end{aligned}$$

For the path constructor, we need to, after some simple rewriting of the goal, provide a path

$$\sigma_n(x -_k |\,a\,|) = \sigma_n(x) \cdot \sigma_n(|\,a\,|)^{-1}$$

for $x : K(G,n)$ and $a : \widetilde{K}(G,n)$ which is easily done using Proposition 20 and Corollary 21. We can now show that, for any $t : K(G,n+1)$ and $p : 0_k = t$, we have $\mathsf{encode}_t(\mathsf{decode}_t(p)) = p$ by path induction on p. In particular, we have $\mathsf{decode}_{0_k}(\mathsf{encode}_{0_k}(p)) = p$ for all $p : \Omega K(G,n+1)$. We can also show that $\mathsf{encode}_{0_k}(\mathsf{decode}_{0_k}(x)) = x$ for all $x : K(G,n)$ by simply unfolding the definitions of decode_{0_k} and encode_{0_k}. Indeed, after applying truncation elimination on x, we have:

$$\begin{aligned}
\mathsf{encode}_{0_k}(\mathsf{decode}_{0_k}(|\,a\,|)) &:= \mathsf{encode}_{0_k}(\sigma_n(|\,a\,|))\\
&:= \mathsf{transport}^{\mathsf{Code}}(\sigma_n(|\,a\,|), 0_k)\\
&= \mathsf{transport}^{\mathsf{Code}\circ|-|}(\mathsf{merid}\ (0_k)^{-1}, \mathsf{transport}^{\mathsf{Code}\circ|-|}(\mathsf{merid}\ a, 0_k))\\
&= \mathsf{transport}^{\mathsf{Code}\circ|-|}(\mathsf{merid}\ (0_k)^{-1}, |\,a\,| + 0_k)\\
&= (-_k 0_k) +_k (|\,a\,| + 0_k)\\
&= |\,a\,|
\end{aligned}$$

and we are done. □

Note that this implies that we have $\Omega^n(K(G,n)) \simeq G$. By Proposition 20, this equivalence is an isomorphism of groups. Together with Proposition 10, this shows that $K(G,n)$ indeed is an Eilenberg–MacLane space.

3.3 The cup product with group coefficients

There is an additional structure definable already on Eilenberg–MacLane spaces: the cup product. The first construction in HoTT is due to Brunerie (2016), who defined it for integral cohomology – that is, he defined $\smile_k : K(\mathbb{Z},n) \times K(\mathbb{Z},m) \to K(\mathbb{Z},n+m)$. This definition was later extended to $\smile_k : K(G_1,n) \times K(G_2,m) \to K(G_1 \otimes G_2, n+m)$ by Baumann (2018). The issue with the definitions of these cup products is that they rely heavily on the properties of smash products. As smash products have turned out to be rather difficult to reason about in HoTT (Brunerie, 2018), not all cup product axioms have been formally verified using these definitions. For this reason, a new definition of the cup product on $\mathbb{Z}$-cohomology was given by Brunerie et al. (2022) which allowed for a complete proof of the graded commutative ring laws. Our goal in this section is to generalise this definition and get a map $\smile_k : K(G_1,n) \times K(G_2,m) \to K(G_1 \otimes G_2, n+m)$ satisfying the expected laws.

Before defining the cup product, we need to define tensor products of abelian groups. In HoTT, we can do this very directly using the obvious HIT:

Definition 23. *Given abelian groups G_1 and G_2 we define their tensor product $G_1 \otimes G_2$ to be the HIT generated by*

- *points $g_1 \otimes g_2 : G_1 \otimes G_2$, for each $g_1 : G_1$ and $g_2 : G_2$*
- *points $x \oplus y$ for each pair of points $x, y : G_1 \otimes G_2$*
- *for any $x, y, z : G_1 \otimes G_2$, paths*

$$x \oplus y = y \oplus x$$
$$x \oplus (y \oplus z) = (x \oplus y) \oplus z$$
$$(0_{G_1} \otimes 0_{G_1}) \oplus x = x$$

- *for any $x : G_1$ and $y, z : G_2$, paths $x \otimes (y +_{G_2} z) = (x \otimes y) \oplus (x \otimes z)$*
- *for any $x, y : G_1$ and $z : G_2$, paths $(x +_{G_1} y) \otimes z = (x \otimes z) \oplus (y \otimes z)$*
- *a constructor* $\mathsf{is\text{-}set} : \mathsf{isSet}\,(G_1 \otimes G_2)$

One easily verifies that this construction has the expected universal property. Consequently, a map out of $G_1 \otimes G_2$ into a set is well-defined if and only if it is structure preserving. When mapping into a proposition, it suffices to construct the map over canonical elements $g_1 \otimes g_2$ and show that the proposition respects $\oplus$.

Let us return to the problem at hand: defining the cup product. We note first that it is very easy to define a preliminary cup product

$$\smile'_k : \widetilde{K}(G_1, n) \times K(G_2, m) \to K(G_1 \otimes G_2, n + m) \tag{3}$$

When $n = 0$, we define $g \smile'_k (-)$ by the functorial action of $K(-, m)$ on the group homomorphism given by $g \otimes (-) : G_2 \to G_1 \otimes G_2$. When $n = 1$, we inductively define:

$$\star \smile'_k y := 0_k$$
$$\mathsf{ap}_{x \mapsto x \smile'_k y}(\mathsf{loop}_k\, g) := \sigma_m(g \smile'_k y)$$

The fact that this respects the sq constructor is easy to check. For $n \geq 2$, the idea is the same:

$$\mathsf{north} \smile'_k y = 0_k$$
$$\mathsf{south} \smile'_k y = 0_k$$
$$\mathsf{ap}_{x \mapsto x \smile'_k y}(\mathsf{merid}\ a) = \sigma_{(n-1)+m}(a \smile'_k y)$$

We would now like to be able to lift $\smile'_k$ to a full cup product $\smile_k : K(G_1, n) \times K(G_2, m) \to K(G_1 \otimes G_2, n + m)$:

$$\begin{array}{ccc} \widetilde{K}(G_1, n) \times K(G_2, m) & \xrightarrow{\smile'_k} & K(G_1 \otimes G_2, n + m) \\ \downarrow & \nearrow & \\ K(G_1, n) \times (K(G_2, m) & & \end{array}$$

or, after currying:

$$\begin{array}{ccc} \widetilde{K}(G_1, n) & \xrightarrow{\smile'_k} & (K(G_2, m) \to K(G_1 \otimes G_2, n + m)) \\ \downarrow & \nearrow & \\ K(G_1, n) & & \end{array}$$

However, we run into an obstruction: for this lift to exist, we require the function type $K(G_2, m) \to K(G_1 \otimes G_2, n+m)$ to be an n-type, which it is not unless $m = 0$. Fortunately, a very trivial observation makes the lift possible: we want our cup product to satisfy $x \smile_k 0_k = 0_k$. In other words, we may see the cup product as a map into a pointed function type:

$$\smile_k : K(G_1, n) \to (K(G_2, m) \to_\star K(G_1 \otimes G_2, n+m))$$

In this case, we do get a lift

$$\begin{array}{ccc} \widetilde{K}(G_1, n) & \xrightarrow{\smile'_k} & (K(G_2, m) \to_\star K(G_1 \otimes G_2, n+m)) \\ \downarrow & \nearrow_{\smile_k} & \\ K(G_1, n) & & \end{array}$$

and thereby the cup product is defined. The existence of the lift is motivated by the following result which follows from a more general result (Buchholtz et al., 2018, Corollary 1), but has a very simple direct proof when the spaces involved are Eilenberg–MacLane spaces.

Proposition 24. *The space $K(G_1, n) \to_\star K(G_2, n+m)$ is m-truncated.*

Proof. It is enough to show that $\Omega^{m+1}(K(G_1, n) \to_\star K(G_2, n+m))$ is contractible. We have

$$\begin{aligned} \Omega^{m+1}(K(G_1, n) \to_\star K(G_2, n+m)) &\simeq (K(G_1, n) \to_\star \Omega^{m+1} K(G_2, n+m)) \\ &\simeq (K(G_1, n) \to_\star K(G_2, n-1)) \end{aligned}$$

The fact that $(K(G_1, n) \to_\star K(G_2, n-1))$ is contractible follows easily from the the $(n-2)$- type elimination rule for $K(G_1, n)$ in Figure 2c. □

This concludes the construction of $\smile_k$. Let us now prove the various ring laws governing it. We first note that since $\smile'_k$ is pointed in both arguments, this also applies to $\smile_k$. Thus, we have verified the first law:

Proposition 25. *For $x : K(G, n)$, we have $x \smile_k 0_k = 0_k$ and $0_k \smile_k x = 0_k$.*

We now set out to prove the remaining laws. This turns out to be rather difficult to do directly: we want to prove these laws using $K(G, n)$-elimination via Proposition 24, but this forces us to prove equalities of pointed maps, rather than just maps. For instance, for left-distributivity we have to prove that for any two points $x, y : K(G_1, n)$ we have that the two maps

$$\begin{aligned} z &\mapsto (x +_k y) \smile_k z \\ z &\mapsto (x \smile_k z) +_k (y \smile_k z) \end{aligned}$$

are equal as *pointed* maps living in $K(G_2, m) \to_\star K(G_1 \otimes G_2, n+m)$. This is far more involved than proving equality of the underlying maps: a proof of equality of pointed maps $(f, p), (g, q) : A \to_\star B$ consists not only of a homotopy $h : (a : A) \to f\, a = g\, a$, but also a coherence $p =^{h(\star_A)}_{\mathsf{refl}} q$. Fortunately, there is a trick to automatically infer the coherence. We attribute this result to Evan Cavallo.[d]

Lemma 26 (Evan's trick)**.** *Given two pointed functions $(f, p), (g, q) : A \to_\star B$ with B homogeneous. If $f = g$, then there is a path of pointed functions $(f, p) = (g, q)$.*

[d]The original form of the lemma was conjectured only for Eilenberg–MacLane spaces and appears as (Brunerie et al., 2022, Lemma 14). Its proof, and generalisation to arbitrary homogeneous types, is attributed to Cavallo, whose original formalisation can be found in (Cavallo, 2022). The result has later been generalised by Buchholtz et al. (2023, Lemma 2.7)

Proof. By assumption, we have a homotopy $h : (x : A) \to f(x) = g(x)$. We construct $r : \star_B = \star_B$ by

$$r := p^{-1} \cdot h(\star_A) \cdot q$$

and define $P : (B, \star_B) =_{\mathcal{U}_\star} (B, \star_B)$ by

$$P := \mathsf{ap}_{(B,-)}(r)$$

We get $P = \mathsf{refl}$ as an easy consequence of the homogeneity of B. Hence, instead of proving that $(f, p) = (g, q)$, it is enough to prove that $\mathsf{transport}^{A \to_\star (-)}(P, (f, p)) = (g, q)$. The transport only acts on p and q, so the identity of the first components holds by h. For the second components, we are reduced to proving that $p \cdot r = h(\star_A) \cdot q$. This is true immediately by the construction of r. □

Proposition 27 (Left-distributivity of $\smile_k$)**.** *Let $x, y : K(G_1, n)$ and $z : K(G_2, m)$. We have*

$$(x +_k y) \smile_k z = x \smile_k z +_k y \smile_k z$$

Proof. When $n = 0$, the result follows by a straightforward induction on m. When $n \geq 1$, we generalise and, for $x, y : K(G_1, n)$, consider the two functions

$$z \mapsto (x +_k y) \smile_k z$$
$$z \mapsto (x \smile_k z) +_k (y \smile_k z)$$

We claim that these are equal as *pointed* functions living in $K(G_2, m) \to_\star K(G_1 \otimes G_2, n + m)$. By Proposition 24, all path types over $K(G_2, m) \to_\star K(G_1 \otimes G_2, n + m)$ are $(n - 1)$-types. This allows us to apply truncation elimination and Proposition 12. In combination with Lemma 26, it suffices to construct, for each $z : K(G_2, m)$:

$$l(y) : (0_k +_k |\, y \,|) \smile_k z = (0_k \smile_k z) +_k (|\, y \,| \smile_k z)$$
$$r(x) : (|\, x \,| +_k 0_k) \smile_k z = (|\, x \,| \smile_k z) +_k (0_k \smile_k z)$$
$$q : l(0_k) = r(0_k)$$

which is direct using the unit laws for $+_k$ and $\smile_k$. The obvious construction makes q hold by refl. □

Right-distributivity follows by an almost identical proof, so we omit it.

Proposition 28 (Right-distributivity of $\smile_k$)**.** *Let $x : K(G_1, n)$ and $y, z : K(G_2, m)$. We have*

$$x \smile_k (y +_k z) = x \smile_k y +_k x \smile_k z$$

Before moving on to associativity, we will need the following lemma. As pointed out by Brunerie et al. (2022), this lemma also appears in (Brunerie, 2016, Proposition 6.1.1) where it is used for the construction of the Gysin sequence. Its proof relies on the pentagon identity for smash products which, at the time, was open in HoTT. In our setting, which like that of Brunerie et al. (2022) uses pointed maps instead of smash products, it holds by construction.

Lemma 29. *For $x : K(G_1, n)$ and $y : K(G_2, m)$, we have*

$$\mathsf{ap}_{x \mapsto x \smile_k y}(\sigma_n(x)) = \sigma_{n+m}(x \smile_k y)$$

Proof. When $n = 0$, the statement holds by definition. Let us consider the case $n \geq 1$. The maps we are comparing are of type $K(G_1, n) \to K(G_2, m) \to_\star \Omega(K(G_1 \otimes G_2, (n + 1) + m))$. The type

$$K(G_2, m) \to_\star \underbrace{\Omega(K(G_1 \otimes G_2, (n + 1) + m))}_{\simeq K(G_1 \otimes G_2, n+m)}$$

is an n-type by Proposition 24, and hence we may use $K(G,n)$-elimination on x. Hence, we want to construct an equality of pointed functions

$$(y \mapsto \mathsf{ap}_{x \mapsto x \smile_k y}(\sigma_n(a))) = (y \mapsto \sigma_{n+m}(a \smile'_k y))$$

for $a : \widetilde{K}(G,n)$. By Lemma 26, it suffices to show this equality for underlying functions. This follows trivially:

$$\begin{aligned}
\mathsf{ap}_{x \mapsto x \smile_k y}(\sigma_n(a)) &:= \mathsf{ap}_{x \mapsto |x| \smile_k y}(\mathsf{merid}\, a \cdot (\mathsf{merid}\, 0_k)^{-1}) \\
&= \mathsf{ap}_{x \mapsto |x| \smile_k y}(\mathsf{merid}\, a) \cdot (\mathsf{ap}_{x \mapsto |x| \smile_k y}(\mathsf{merid}\, 0_k))^{-1} \\
&:= \sigma_{n+m}(a \smile'_k y) \cdot \sigma_{n+m}(0_k \smile'_k y)^{-1} \\
&= \sigma_{n+m}(a \smile'_k y)
\end{aligned}$$

□

With Lemma 29, associativity is straightforward to prove.

Proposition 30. *The cup product is associative. That is, the following diagram commutes:*

$$\begin{array}{ccc}
K(G_1,n) \times K(G_2,m) \times K(G_3,k) & \xrightarrow{x,y,z \mapsto x \smile_k (y \smile_k z)} & K(G_1 \otimes (G_2 \otimes G_3), n + (m+k)) \\
& \searrow^{x,y,z \mapsto (x \smile_k y) \smile_k z} & \uparrow \wr \\
& & K((G_1 \otimes G_2) \otimes G_3, (n+m)+k)
\end{array}$$

Proof. We induct on n. Let us first consider the inductive step and save the base case for last. To this end, let $n \geq 1$. We may consider the two functions we are comparing as doubly pointed functions of type

$$K(G_1,n) \to (K(G_2,m) \to_\star (K(G_3,k) \to_\star K(G_1 \otimes (G_2 \otimes G_3), n+m+k)))$$

Applying an appropriate variant of Proposition 24, the codomain of the above function type is an n-type. When x is a point constructor of $K(G_1,n)$, the diagram in the statement commutes by refl. For the higher constructor (merid or loop_k, depending on the value of n), we are done if we can show, for $x : K(G_1, n-1)$, that

$$\mathsf{ap}_{x \mapsto x \smile_k (y \smile_k z)}(\sigma_{n-1}(x)) = \mathsf{ap}_{x \mapsto \alpha_{n+m+k}((x \smile_k y) \smile_k z)}(\sigma_{n-1}(x))$$

where $\alpha : (G_1 \otimes G_2) \otimes G_3 \simeq G_1 \otimes (G_2 \otimes G_3)$ and, recall, $\alpha_p : K((G_1 \otimes G_2) \otimes G_3, p) \to K(G_1 \otimes (G_2 \otimes G_3), p)$ is the functorial action of $K(-,p)$ on α. We have

$$\begin{aligned}
\mathsf{ap}_{x \mapsto x \smile_k (y \smile_k z)}(\sigma_{n-1}(x)) &= \sigma_{(n-1)+(m+k)}(x \smile_k (y \smile_k z)) && \text{(Lemma 29)} \\
&= \sigma_{(n-1)+(m+k)}(\alpha_{(n-1)+(m+k)}((x \smile_k y) \smile_k z)) && \text{(Ind. hyp.)} \\
&= \mathsf{ap}_{\alpha_{n+m+k}}(\sigma_{((n-1)+m)+k}((x \smile_k y) \smile_k z)) && \text{(Lemma 11)} \\
&= \mathsf{ap}_{\alpha_{n+m+k}}(\mathsf{ap}_{x \mapsto x \smile_k z}(\sigma_{((n-1)+m)}(x \smile_k y))) && \text{(Lemma 29)} \\
&= \mathsf{ap}_{\alpha_{n+m+k}}(\mathsf{ap}_{x \mapsto x \smile_k z}(\mathsf{ap}_{x \mapsto x \smile_k y}(\sigma_{(n-1)}(x)))) && \text{(Lemma 29)} \\
&= \mathsf{ap}_{x \mapsto \alpha_{n+m+k}((x \smile_k y) \smile_k z)}(\sigma_{n-1}(x))
\end{aligned}$$

which concludes the inductive step.

Let us return to the base case, namely, the case $n = 0$. In this case, we fix $g_1 : G_1$ and instead consider the functions $y, z \mapsto g_1 \smile_k (y \smile_k z)$ and $y, z \mapsto (g_1 \smile_k y) \smile_k z$ to be of type

$$K(G_2,m) \to K(G_3,k) \to_\star K(G_1 \otimes (G_2 \otimes G_3), m+k).$$

We induct on m. The inductive step follows in the same way as the inductive step above. For the base case, we fix y to be $g_2 : G_2$ and are left to prove $g_1 \smile_k (g_2 \smile_k z) = (g_1 \smile_k g_2) \smile_k z$ for $z : K(G_3, k)$. Again, this is follows by induction on k. The base case is given by associativity of $G_1 \otimes (G_2 \otimes G_3)$ and the inductive step follows in exactly the same way as before. □

Let us verify graded commutativity. Although our argument is a direct generalisation of that of Brunerie et al. (2022, Proposition 18), we remark that a similar albeit somewhat more high-level argument appears in (Wärn, 2023, Section 4.3).

Proposition 31. *The cup product is graded commutative. That is, the following diagram commutes:*

$$\begin{array}{ccc} K(G_1, n) \times K(G_2, m) & \xrightarrow{\smile_k} & K(G_1 \otimes G_2, n+m) \\ & \searrow^{x,y \mapsto (-1)^{nm}(y \smile_k x)} & \uparrow \wr \\ & & K(G_2 \otimes G_1, m+n) \end{array}$$

Proof. We may, without loss of generality, assume that $n \geq m$. We induct on the quantity $n + m$. The base-case when $n = m = 0$ is trivial. Let us consider the two cases $n = m = 1$ and $n, m \geq 2$ and omit the two cases when $n \geq 2$ and $m < 2$ as they follow in an entirely analogous manner. In what follows, let $c_p : K(G_2 \otimes G_1, p) \to K(G_1 \otimes G_2, p)$ denote the commutator.

We start with the case $n = m = 1$. In this case, we would like to be able to apply the set-elimination rule for $K(-, 1)$. As before, fix $x : K(G_1, 1)$. We strengthen the goal by asking for $x \smile_k (-)$ and $y \mapsto (-1)^{nm}(y \smile_k x)$ to be equal as pointed functions living in $K(G_2, 1) \to_\star K(G_1 \otimes G_2, 2)$. The type of such functions is a 1-type, and hence any path type over it is a set. Thus, we may apply set-elimination and assume that $x : L(G_1)$. We may now repeat the argument for $y : K(G_2, n)$: we would like $(-) \smile_k y$ and $x \mapsto (-1)^{nm}(y \smile_k x)$ to be equal as pointed functions living in $L(G_1) \to_\star K(G_1 \otimes G_2, 2)$. By an very similar argument to that of Proposition 24, this pointed function type is again a 1-type, which justifies set-elimination being applied to y. Combining this with Lemma 26, it is enough to show that

$$x \smile_k y = -c_2(y \smile_k x)$$

for $x : L(G_1)$ and $y : L(G_2)$. We do this by $L(G_1)$ induction. When either x or y is a point constructor, the identity holds by refl. The remaining case amounts to (by evaluation of c_2) the following identity

$$\mathsf{ap}_{x \mapsto \mathsf{ap}_{x \smile_k (-)}(\mathsf{loop}_k\, h)}(\mathsf{loop}_k\, g) = \mathsf{flip}(\mathsf{ap}_{x \mapsto \mathsf{ap}_{x \smile_k (-)}(\mathsf{loop}_k\, h)}(\mathsf{loop}_k\, g)^{-1})$$

which holds by Lemma 6.

We give a rough sketch of the case $n, m \geq 2$. We may apply $K(G_1, n)$-elimination again, by the same argument as in the previous case. The crucial step this time is verifying

$$\mathsf{ap}_{x \mapsto \mathsf{ap}_{x \smile_k (-)}(\mathsf{merid}\ h)}(\mathsf{merid}\ g) = \mathsf{ap}_{x \mapsto \mathsf{ap}_{(-1)^{nm}(x \smile_k (-))}(\mathsf{merid}\ h)}(\mathsf{merid}\ g) \tag{4}$$

We note that the above identity and the identities that follow only are well-typed up to some coherences which we brush under the rug for the sake of readability. We notice that (4) follows if we can show that for $x : K(G_1, n-1)$, $y : K(G_2, m-1)$, we have

$$\mathsf{ap}_{x \mapsto \mathsf{ap}_{x \smile_k (-)}(\sigma_{m-1}(y))}(\sigma_{n-1}(x)) = \mathsf{ap}_{x \mapsto \mathsf{ap}_{(-1)^{nm}(x \smile_k (-))}(\sigma_{m-1}(y))}(\sigma_{n-1}(x)) \tag{5}$$

Let us, for further readability, omit the equivalence c_p in the following equations. Some of the following equalities hold up to multiplication by $(-1)^p$ for some $p : \mathbb{N}$. This will be indicated in

the right-hand column.

$$\begin{aligned}
\mathsf{ap}_{a\mapsto \mathsf{ap}_{a\smile_k(-)}(\sigma_{m-1}(y))}(\sigma_{n-1}(x)) &= \mathsf{ap}_{a\mapsto \mathsf{ap}_{(-)\smile_k a}(\sigma_{n-1}(x))}(\sigma_{m-1}(y)) && (-1)\\
&= \mathsf{ap}_{\sigma_{(n-1)+m}}(\mathsf{ap}_{x\smile_k(-)}(\sigma_{m-1}(y)))\\
&= \mathsf{ap}_{\sigma_{m+(n-1)}}(\mathsf{ap}_{(-)\smile_k x}(\sigma_{m-1}(y))) && m(n-1)\\
&= \mathsf{ap}_{\sigma_{(m-1)+(n-1)+1}}(\sigma_{(m-1)+(n-1)}(y\smile_k x))\\
&= \mathsf{ap}_{\sigma_{(n-1)+(m-1)+1}}(\sigma_{(n-1)+(m-1)}(x\smile_k y)) && (n-1)(m-1)\\
&= \mathsf{ap}_{\sigma_{n+(m-1)}}(\mathsf{ap}_{(-)\smile_k y}(\sigma_{n-1}(x)))\\
&= \mathsf{ap}_{\sigma_{m+(n-1)}}(\mathsf{ap}_{y\smile_k(-)}(\sigma_{n-1}(x))) && (m-1)n\\
&= \mathsf{ap}_{a\mapsto \mathsf{ap}_{(-)\smile a}(\sigma_{m-1}(y))}(\sigma_{n-1}(x))
\end{aligned}$$

Since $(-1)+n(m-1)+(n-1)(m-1)+(n-1)m =_{\mathbb{Z}/2\mathbb{Z}} nm$, the above identity holds up to multiplication by $(-1)^{nm}$, which gives (5). □

3.4 The cup product with ring coefficients

We now consider the special case of $\smile_k : K(G_1,n)\times K(G_2,m)\to K(G_1\otimes G_2, n+m)$ where $G_1 = G_2 = R$ for a (not necessarily commutative) ring R. Let $\mathsf{mult}_n : K(R\otimes R,n)\to K(R,n)$ be the map induced by the multiplication $R\otimes R\to R$. Now, consider the composition

$$K(R,n)\times K(R,m) \xrightarrow{\smile_k} K(R\otimes R, n+m) \xrightarrow{\mathsf{mult}_{n+m}} K(R,n+m)$$

Let us, with some overloading of notation, use $\smile_k : K(R,n)\times K(R,m)\to K(R,n)$ to denote also the composition above. Note that the above tensor product is a tensor product of underlying abelian groups and not of rings, although this is of no consequence for the construction. Using the results from Section 3.3, we easily arrive at the following results. Their proofs all follow the same strategy and we will only sketch the proof of Proposition 33.

Proposition 32. *The cup product $\smile_k$ with ring coefficients is associative, left- and right-distributive and has 0_k as annihilating element.*

Proposition 33. *The cup product $\smile_k$ with coefficients in a commutative ring is graded-commutative.*

Proof. Let c_p denote the commutator $K(G\otimes H,p)\to K(H\otimes G,p)$. Consider the following diagram.

$$\begin{array}{ccccc}
K(R,n)\times K(R,m) & \xrightarrow{\smile_k} & K(R\otimes R, n+m) & \xrightarrow{\mathsf{mult}_{n+m}} & K(R,n+m)\\
\downarrow & & \downarrow{\scriptstyle c\circ(-1)^{nm}} & & \downarrow{\scriptstyle (-1)^{nm}}\\
K(R,m)\times K(R,n) & \xrightarrow{\smile_k} & K(R\otimes R, m+n) & \xrightarrow{\mathsf{mult}_{m+n}} & K(R,m+n)
\end{array}$$

The statement is that the outer square commutes. By Proposition 31 the left-square commutes. It is easy to verify, using commutativity of R, that $\mathsf{mult}_{n+m}\circ c = \mathsf{mult}_{m+n}$. Hence, the right-square commutes, and we are done. □

Finally, we note that we now have access to a neutral element $1_R : K(R,0)$ which is a left and right unit for $\smile_k$:

Proposition 34. *For $x : K(R, n)$, we have $x \smile_k 1_R = x = 1_R \smile_k x$.*

Proof. By a straightforward induction on n, using the recursive definition of the cup product. □

4. Cohomology

For the construction of cohomology groups, we rely on Brown (1962) representability: given $n : \mathbb{N}$, an abelian group G and a type X, we define the *nth cohomology group of X with coefficients in G* by[e]

$$H^n(X, G) := \|X \to K(G, n)\|_0$$

This construction is clearly contravariantly functorial by pre-composition. The group structure on $K(G, n)$ naturally carries over to $H^n(X, G)$ by point-wise application of $+_k$ and $-_k$. The neutral element is, in both cases, the constant map $\_ \mapsto 0_k$. In what follows, we will denote addition, inversion and the neutral element in $H^n(X, G)$ by $+_h$, $-_h$ and 0_h respectively. Explicitly, they are defined by

$$\begin{aligned} |\, f \,| +_h |\, g \,| &:= |\, x \mapsto f(x) +_k g(x) \,| \\ -_h |\, f \,| &:= |\, x \mapsto -_k f(x) \,| \\ 0_h &:= |\, \_ \mapsto 0_k \,| \end{aligned}$$

In the same way, our construction of the cup product also carries over to cohomology groups. In particular, the cup product with ring coefficients in Section 3.4 gives us a cup product on cohomology

$$\smile : H^n(X, R) \times H^m(X, R) \to H^{n+m}(X, R)$$

satisfying the usual graded ring axioms and, when R is commutative, graded commutativity. In addition, it induces a multiplication on $H^*(X, R) = \oplus_{n:\mathbb{N}} H^n(X, R)$, turning it into a graded-commutative (assuming R is commutative) ring known as the *cohomology ring* of X with coefficients in R. This was formalised in `Cubical Agda` by Lamiaux et al. (2023). This gives an even more fine-grained invariant than just the cohomology groups $H^n(X, G)$, as it can help to distinguish spaces for which all cohomology groups agree. As a classic example which was also considered by Lamiaux et al. (2023), consider Figure 3 which depicts a torus and the 'Mickey Mouse space' consisting of a sphere glued together with two circles.

Figure 3: $\mathbb{T}^2$ and $\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1$.

Let X be either of the two spaces in the figure. One can prove that $H^0(X, \mathbb{Z}) \cong \mathbb{Z}$ as they are both connected, $H^1(X, \mathbb{Z}) \cong \mathbb{Z} \times \mathbb{Z}$ as they each have two 2-dimensional holes (the circle in the middle of the torus and the one going around the interior, and the 'ears' of Mickey), and $H^2(X, \mathbb{Z}) \cong \mathbb{Z}$ as

[e] Here, one could more generally have chosen a dependent version by letting $H^n(X, G_{(-)}) := \|(x : X) \to K(G_x, n)\|_0$ for any family of abelian groups G_x over X as is done by, for example, van Doorn (2018, Definition 5.4.2). We choose the non-dependent version for ease of presentation, although everything which follows can be read with the dependent version in mind.

they have one 3-dimensional hole each (the interior of the torus and Mickey's head). Furthermore, their higher cohomology groups are all trivial as they do not have any further higher dimensional cells. The cohomology groups are hence insufficient to tell the spaces apart. However, one can show that the cup product on $\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1$ is trivial, while for $\mathbb{T}^2$ it is not. So $H^*(\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1, \mathbb{Z}) \not\simeq H^*(\mathbb{T}^2, \mathbb{Z})$, which again means that the spaces cannot be equivalent (see Section 6.1 for details on how this can be proved by computer computation in `Cubical Agda`).

4.1 Reduced cohomology

When X is pointed, we may also define *reduced* cohomology groups by simply requiring all functions to be pointed:

$$\widetilde{H}^n(X, G) := \|X \to_\star K(G, n)\|_0$$

Just like before, the additive structure on $K(G, n)$ carries over directly to $\widetilde{H}(X, G)$. As we will soon see, this definition is equivalent to the non-reduced version when $n \geq 1$ but has the advantage of vanishing for connected spaces when $n = 0$.

Proposition 35. *The first projection (sending a pointed map to its underlying function) induces an isomorphism $\widetilde{H}^n(X, G) \cong H^n(X, G)$ for $n \geq 1$.*

Proof. Let F denote the function in question. The fact that it is a homomorphism is trivial. We may construct its inverse $F^{-1} : H^n(X, G) \to \widetilde{H}^n(X, G)$ explicitly by

$$F^{-1}\,|\,f\,| := |\,x \mapsto f(x) -_k f(\star_X)\,|$$

In order to see that $F(F^{-1}|\,f\,|) = |\,f\,|$, we note that this is a proposition. By connectedness of $K(G, 1)$, we may assume we have a path $f(\star_X) = 0_k$. Hence

$$F(F^{-1}\,|\,f\,|) := |\,x \mapsto f(x) -_k f(\star_X)\,| = |\,f\,| \tag{6}$$

For the other direction, we need to check that

$$F^{-1}(F\,|\,(f, p)\,|) = |\,(f, p)\,|$$

By Lemma 26, we only need to verify that the first projections agree, and we are reduced to the second equality in (6), which we have already verified. □

Proposition 36. $\widetilde{H}^0(X, G) \oplus G \cong H^0(X, G)$.

Proof. We define $\phi : (X \to_\star G) \times G \to (X \to G)$ by

$$\phi(f, g) := x \mapsto f(x) +_G g$$

and $\psi : (X \to G) \to (X \to_\star G) \times G$ by

$$\psi(f) := ((x \mapsto f(x) -_G f(\star_X)), f(\star_X))$$

The fact that these maps cancel out follows immediately from the group laws on G. This induces, after applying the appropriate set truncations, an equivalence $H^0(X, G) \simeq \widetilde{H}^0(X, G) \oplus G$. The fact that it is an isomorphism is also trivial using the group laws on G. □

4.2 Eilenberg–Steenrod axioms for cohomology

In order to verify that our cohomology theory is sound, let us verify the *Eilenberg–Steenrod Axioms* (Eilenberg and Steenrod, 1952) for the reduced version of cohomology $\widetilde{H}^n(X, G)$. These

axioms have been studied previously in HoTT by Shulman (2013); Cavallo (2015); Buchholtz and Hou (Favonia) (2018); van Doorn (2018). To state the **Exactness** axiom, we need to introduce cofibres (also known as *mapping cones*).

Definition 37. *Given a function $f : X \to Y$, the cofibre of f, denoted* cofib f*, is pushout of the span $\mathbb{1} \leftarrow X \xrightarrow{f} Y$. We will use the name* cfcod *for the inclusion* $\mathsf{inr} : Y \to \mathsf{cofib}\ f$.

$$\begin{array}{ccc} X & \xrightarrow{\quad f \quad} & Y \\ \downarrow & \ulcorner & \downarrow{\scriptstyle \mathsf{cfcod}} \\ \mathbb{1} & \longrightarrow & \mathsf{cofib}\ f \end{array}$$

We will also need *wedge sums*.

Definition 38. *Given a family of pointed types X_i over an indexing type I, the wedge sum of X_i, denoted $\bigvee_i X_i$, is the cofibre of the inclusion $j : I \to \Sigma_{i:I} X_i$ defined by $j(i) = (i, \star_{X_i})$.*

From a constructive point of view, we will primarily be interested in wedge sums indexed by a type satisfying the set-level axiom of choice (see, for instance, (Cavallo, 2015, Section 3.2.3), (van Doorn, 2018, Section 5.4), and (Buchholtz and Hou (Favonia), 2018, Section 6) for further discussions of this).

Definition 39. *A type I is said to satisfy the set-level axiom of choice (AC_0) if for any family of types X_i over I, the canonical map*

$$\|(i : I) \to X_i\|_0 \to ((i : I) \to \|X_i\|_0)$$

is an equivalence.

In particular, any finite set I satisfies AC_0. We may now state the Eilenberg–Steenrod axioms for reduced cohomology.

Definition 40. *A $\mathbb{Z}$-indexed family of contravariant functors $E^n : \mathcal{U}_\star \to \mathsf{AbGroup}$ is said to be an ordinary, reduced cohomology theory if the following axioms hold.*

- ***Suspension:*** *there is an isomorphism $E^n(X) \cong E^{n+1}(\Sigma X)$ natural in X.*
- ***Exactness:*** *given a function $f : X \to_\star Y$, there is an exact sequence*

$$E^n(\mathsf{cofib}\ f) \xrightarrow{E^n\,\mathsf{cfcod}} E^n(Y) \xrightarrow{E^n f} E^n(X)$$

- ***Dimension:*** *for $n : \mathbb{Z}$ with $n \neq 0$, we have $E^n(\mathbb{S}^0) \cong \mathbb{1}$.*
- ***Additivity:*** *for $n : \mathbb{Z}$ and a family of pointed types X_i over a type I satisfying AC_0, the map $E^n(\bigvee_i X_i) \to \Pi_{i:I}(E^n(X_i))$ induced by the functorial action of E^n on the inclusion $X_i \to \bigvee_{i:I} X_i$ is an isomorphism of groups.*

The following theorem was originally proved in Book HoTT by Cavallo (2015). We spell out some of the details for our setup here. In what follows, let $\widetilde{H}^n(X, G) := \mathbb{1}$ for $n < 0$.

Theorem 41. *Given an abelian group G, the reduced cohomology theory $\widetilde{H}^n(-, G)$ satisfies the Eilenberg–Steenrod Axioms.*

Proof. We verify that $\widetilde{H}^n(-, G)$ satisfies the four axioms.

- **Suspension:** this axiom is trivially satisfied when $n < -1$. When $n = -1$, we need to verify that $\widetilde{H}^0(\Sigma X, G)$ is trivial, which follows immediately by connectedness of ΣX. For $n \geq 0$, the suspension axiom follows immediately from the fact that Σ and Ω are adjoint. We have

$$\begin{aligned}\widetilde{H}^{n+1}(\Sigma X, G) &= \|\Sigma X \to_\star K(G, n+1)\|_0 \\ &\simeq \|X \to_\star \Omega\, K(G, n+1)\|_0 \\ &\simeq \|X \to_\star K(G, n)\|_0 \\ &= \widetilde{H}^n(X, G)\end{aligned}$$

  It is easy to see that the above equivalence of types is a homomorphism and is natural in X. This concludes the proof of the suspension axiom.

- **Exactness:** Let $f : X \to Y$. We want to show that the sequence

$$\widetilde{H}^n(\mathsf{cofib}\, f, G) \xrightarrow{\mathsf{cfcod}^*} \widetilde{H}^n(Y, G) \xrightarrow{f^*} \widetilde{H}^n(X, G)$$

  is exact.[f] In other words, we want to show that, given an element $|\, g \,| : \widetilde{H}^n(Y, G)$, we have $|\, g \,| \in \ker(f^*)$ iff $g \in \mathsf{im}(\mathsf{cfcod}^*)$. For the left-to-right implication, the assumption $|\, g \,| \in \ker(f^*)$ gives us a homotopy[g] $p : (x : X) \to 0_k = g(f(x))$. Using this, we construct a function $g_{\mathsf{cf}} : \mathsf{cofib}\, f \to K(G, n)$ by

$$\begin{aligned}g_{\mathsf{cf}}(\mathsf{inl}\, \star) &= 0_k \\ g_{\mathsf{cf}}(\mathsf{inr}\, y) &= g(y) \\ \mathsf{ap}_{g_{\mathsf{cf}}}(\mathsf{push}\, x) &= p(x)\end{aligned}$$

  The fact that $\mathsf{cfcod}^*(|\, g_{\mathsf{cf}} \,|) = |\, g \,|$ holds by construction, and hence $|\, g \,| \in \mathsf{im}(\mathsf{cfcod}^*)$.

  For the other direction, we simply want to show that the composition $f^* \circ \mathsf{cfcod}^*$ is null-homotopic. This holds by construction, since $\mathsf{cfcod} \circ f : X \to \mathsf{cofib}\, f$ is null-homotopic by definition of $\mathsf{cofib}\, f$.

- **Dimension:** when $n < 0$, the statement is trivial. When $n > 0$, we have

$$\widetilde{H}^n(\mathbb{S}^0, G) = \|\mathbb{S}^0 \to_\star K(G, n)\|_0 \simeq \|K(G, n)\|_0$$

  Since $K(G, n)$ is $(n-1)$-connected, $\|K(G, n)\|_0$ vanishes for $n > 0$.

[f] Recall that the action on maps by $\widetilde{H}^n(-, G)$ is defined by precomposition. That is, $\widetilde{H}^n(f, G) :\equiv f^*$.

[g] This homotopy only exists up to propositional truncation, but since we are proving a proposition, this is not an obstacle.

- **Additivity:** Again, the statement is trivial when $n<0$. For the non-trivial case, the result follows from a simple rewriting of $\widetilde{H}^n(\bigvee_{i:I} X_i, G)$ using the elimination principle for $\bigvee_{i:I} X_i$:

$$\begin{aligned}
\widetilde{H}^n\left(\bigvee_{i:I} X_i, G\right) &:= \left\| \bigvee_{i:I} X_i \to_\star K(G,n) \right\|_0 \\
&\simeq \left\| \sum_{f:(\sum_{i:I} X_i)\to K(G,n)} \sum_{x_0:K(G,n)} (((i:I) \to f(i, \star_{X_i}) = x_0) \times (x_0 = 0_k)) \right\|_0 \\
&\simeq \left\| \sum_{f:(\sum_{i:I} X_i)\to K(G,n)} \sum_{(x_0,p):\sum_{x:K(G,n)}(x=0_k)} ((i:I) \to f(i, \star_{X_i}) = x_0) \right\|_0 \\
&\simeq \left\| \sum_{f:(\sum_{i:I} X_i)\to K(G,n)} ((i:I) \to f(i, \star_{X_i}) = 0_k) \right\|_0 \\
&\simeq \| (i:I) \to X_i \to_\star K(G,n) \|_0 \\
&\simeq (i:I) \to \| X_i \to_\star K(G,n) \|_0 \qquad (\mathsf{AC}_0) \\
&= \Pi_{i:I} \widetilde{H}^n(X_i, G)
\end{aligned}$$

  where the step from the third to fourth line is simply a deletion of the (contractible) singleton type $\sum_{x:K(G,n)}(x = 0_k)$. The fact that this composition of equivalences indeed gives the usual homomorphism $\widetilde{H}^n(\bigvee_{i:I} X_i, G) \to \Pi_{i:I} \widetilde{H}^n(X_i, G)$ is immediate by construction. □

Just like in traditional algebraic topology, many standard results and constructions follow directly from the axioms – see, for instance, (Cavallo, 2015; Buchholtz and Hou (Favonia), 2018) for concrete examples, including the Mayer–Vietoris sequence (Cavallo, 2015, Section 4.5).

4.3 The Mayer–Vietoris sequence

The Mayer–Vietoris sequence is surprisingly easy to construct in HoTT. It was first developed in HoTT by Cavallo (2015) for axiomatic cohomology theories and then for the specific $\mathbb{Z}$-cohomology theory of Brunerie (2016) who constructed the sequence directly. This direct construction in no way depends on the use of $\mathbb{Z}$-coefficients and the exact same proof translates to our theory. In this short section, we will briefly recall this construction. All notations are borrowed from Brunerie (2016).

Consider three types X, Y, Z with functions $f : X \to Y$ and $g : X \to Z$. Let D denote the pushout of the span $Y \xleftarrow{f} X \xrightarrow{g} Z$. There is a natural map $\widetilde{d} : (X \to K(G,n)) \to (D \to K(G, n+1))$ given by

$$\begin{aligned}
\widetilde{d}(\gamma)(\mathsf{inl}\, y) &:= 0_k \\
\widetilde{d}(\gamma)(\mathsf{inr}\, z) &:= 0_k \\
\mathsf{ap}_{\widetilde{d}(\gamma)}(\mathsf{push}\, x) &:= \sigma_n(\gamma(x))
\end{aligned}$$

This lifts to a map $d : H^n(X, G) \to H^{n+1}(D, G)$ by $d(|\gamma|) = |\widetilde{d}(\gamma)|$. There is also a natural map $\Delta : H^n(Y) \times H^n(Z) \to H^n(X, G)$ given by

$$\Delta(\alpha, \beta) := f^*(\alpha) -_h g^*(\beta)$$

Finally, we have a map $i : H^n(D, G) \to H^n(Y, G) \times H^n(Z, G)$ given by

$$i(\delta) := (\mathsf{inl}^*(\delta), \mathsf{inr}^*(\delta))$$

Theorem 42 (Mayer–Vietoris sequence)**.** *The families of functions i, Δ and d give rise to a long exact sequence on the form:*

$$
\begin{array}{ccccc}
H^0(D,G) & \xrightarrow{\ i\ } & H^0(Y,G)\times H^0(Z,G) & \xrightarrow{\ \Delta\ } & H^0(X,G) \\
 & \overset{d}{\swarrow} & & & \\
H^1(D,G) & \xrightarrow{\ i\ } & H^1(Y,G)\times H^1(Z,G) & \xrightarrow{\ \Delta\ } & H^1(X,G) \\
 & \overset{d}{\swarrow} & & & \\
H^2(D,G) & \xrightarrow{\ i\ } & \ldots & &
\end{array}
$$

Proof. Straightforward generalisation of Brunerie (2016, Proposition 5.2.2). □

This is useful to compute cohomology of spaces defined as pushouts, such as $\mathbb{C}P^2$ (see Section 5.4). Let us now consider another useful long exact sequence developed in HoTT by Brunerie (2016) known as the *Gysin sequence*.

4.4 The Thom isomorphism and the Gysin sequence

Brunerie (2016) also introduced, for the first time in HoTT, the Thom isomorphism and the Gysin sequence. While Brunerie's original constructions concerned integral cohomology, we may easily generalise them to cohomology with coefficients in an arbitrary commutative ring R. In this short section, we will give a quick review of Brunerie's proof/construction – here generalised to arbitrary coefficients (which has no major effect on the proofs).

Let $\alpha_n : \mathbb{S}^n \to_\star K(R,n)$ denote the image of $1_r : R$ under the equivalence

$$R \simeq \Omega^n K(R,n) \simeq (\mathbb{S}^n \to_\star K(R,n))$$

This induces a family of equivalences $g^i : K(R,i) \to (\mathbb{S}^n \to_\star K(R,i+n))$ defined by

$$g^i(x) := y \mapsto x \smile_k \alpha_n(y)$$

Now, let B be a 0-connected, pointed type and suppose we have fibration $Q : B \to \mathsf{Type}_\star$ with $\psi : Q(\star_B) \simeq_\star \mathbb{S}^n$ and a B-indexed family of functions $c_b : Q(b) \to_\star K(R,n)$ with $c_{\star_B} = \alpha_n \circ \psi$. We remark that such a family automatically exists when B is further assumed to be 1-connected. We may use this to define a B-relative version of g^i:

$$g^i_b : K(R,i) \to (Q(b) \to_\star K(R,i+n))$$
$$g^i_b(x) := y \mapsto x \smile_k c_b(y)$$

Showing that this map is an equivalence is straightforward: for connectedness reasons, we only need to do so when b is $\star_B$, in which case we have $g^i_{\star_B}(x) = g^i(x) \circ \psi$ which we already know defines an equivalence.

The fact that g^i_b is an equivalence means, in particular, that the following map is one too.

$$\varphi^i : (B \to K(R,i)) \to ((b : B) \to Q(b) \to_\star K(R,i+n))$$
$$\varphi^i(f) := (b, q \mapsto f(b) \smile c_b(q))$$

We instantiate the above with some concrete spaces. Let $P : B \to \mathrm{Type}$ be a fibration with $P(\star_B) \simeq \mathbb{S}^{n-1}$ and let $Q(b)$ be the suspension of $P(b)$ – in other words, let $Q(b) := \Sigma(P(b))$. Suppose also that we have a family $c_b : (Q(b) \to_\star K(R,n))$ as above (again, this is automatic if B is 1-connected). Let $E := \Sigma_{b:B} P(b)$ be the total space of P and $\widetilde{E}$ be the associated *Thom space* or, in other words, the cofibre of the projection map $E \to B$. It is easy to see that $((b : B) \to Q(b) \to_\star$

$K(R,k)) \simeq (\widetilde{E} \to_\star K(R,k))$. Combining this with φ^i, we get an isomorphism

$$\phi^i : H^i(B,R) \cong \widetilde{H}^{i+n}(\widetilde{E},R)$$

Finally, we may consider the long exact sequence related to the pushout defining $\widetilde{E}$ (the first row in the following diagram). Substituting along ϕ^{i-n} yields the Gysin sequence (the second row in the diagram).

$$\begin{array}{ccccccccc}
\ldots & \longrightarrow & H^{i-1}(E,R) & \longrightarrow & \widetilde{H}^i(\widetilde{E},R) & \longrightarrow & H^i(B,R) & \longrightarrow & H^i(E,R) & \longrightarrow & \ldots \\
 & & \| & & {\scriptstyle \phi^{i-n}}\uparrow & & \| & & \| & & \\
\ldots & \longrightarrow & H^{i-1}(E,R) & \dashrightarrow & H^{i-n}(B,R) & \xrightarrow[(-)\smile e]{} & H^i(B,R) & \longrightarrow & H^i(E,R) & \longrightarrow & \ldots
\end{array}$$

Here the dashed arrow is defined as the composition $H^{i-1}(E,R) \to \widetilde{H}^i(\widetilde{E},R) \xrightarrow{(\phi^{i-n})^{-1}} H^{i-n}(B,R)$.

Above, the class $e : H^n(B,R)$ is defined by $e := |\, b \mapsto c_b(\mathsf{south}) \,|$. The commutativity of the centre square follows immediately by construction of ϕ^{i-n}. For the sake of completeness, let us state the existence of the Gysin sequence as a theorem.

Theorem 43 (The Gysin Sequence)**.** *Let B be 0-connected and $P : B \to \mathsf{Type}$ be a fibration equipped with an equivalence $\psi : P(\star_B) \simeq \mathbb{S}^{n-1}$. Assume that the following condition is satisfied:*

(∗) There is a B-indexed family $c_b : \Sigma(P(b)) \to_\star K(R,n)$ s.t. $c_{\star_B}$ is the image of $1_R : R$ under the composition of isomorphisms

$$R \cong H^n(\mathbb{S}^n,R) \overset{(\Sigma(\psi))^*}{\cong} H^n(\Sigma(P(\star_B)),R)$$

Let E denote the total space of P and define $e : H^n(B,R)$ by $e := |\, b \mapsto c_b(\mathsf{south}) \,|$. We get a long exact sequence

$$\ldots \longrightarrow H^{i-1}(E,R) \longrightarrow H^{i-n}(B,R) \xrightarrow{(-)\smile e} H^i(B,R) \longrightarrow H^i(E,R) \longrightarrow \ldots$$

Moreover, condition (∗) is always satisfied when B is 1-connected.

The Gysin sequence was for instance crucially used by Brunerie (2016) to compute the integral cohomology *ring* of $\mathbb{C}P^2$ (see also Section 5.4). We will also use this sequence to compute the cohomology ring of the infinite real projective space in Section 5.5.

5. Computations of cohomology groups and rings

In this section we will compute some cohomology groups and rings of some common spaces. All of these groups/rings will be very familiar to the traditional mathematician. Our intent is to showcase how these computations are carried out in HoTT. What is particularly interesting here is that the computations can often be done very directly, without relying on more general machinery (for example, the Mayer–Vietoris sequence, spectral sequences, and so on). We have two reasons for preferring direct proofs whenever possible. Firstly, we think that the existence of such proofs in many cases showcases the strengths of the synthetic approach to cohomology theory in HoTT. Many proofs become incredibly short and rely solely on spelling out the computation rule for the space in question. In particular, we can carry out these proofs without any homological algebra. Secondly, by carefully choosing the direct proofs, we often produce proof terms which

are far simpler than those produced by more advanced machinery. This means that we can use the isomorphisms we construct here to make concrete computations (that is, normalisation of closed terms) in a constructive proof assistant such as `Cubical Agda`. We will discuss this more in Section 6.

Let us start by computing the cohomology groups of some special types.

Proposition 44. $H^n(\mathbb{1}, G) \cong \begin{cases} G & n = 0 \\ \mathbb{1} & n \geq 1 \end{cases}$

Proof. We have $H^n(\mathbb{1}) := \|\mathbb{1} \to K(G,n)\|_0 \simeq \|K(G,n)\|_0$. This is $\|G\|_0 \simeq G$ when $n = 0$ and contractible when $n > 0$ since $K(G,n)$ is $(n-1)$-connected. □

We also include note the following easy lemma

Lemma 45. *For any type X, we have $H^n(X, G) \cong H^n(\|X\|_n, G)$.*

Proof. Using that $K(G,n)$ is n-truncated, we get, by the universal property of n-truncation, that

$$H^n(X,G) = \|X \to K(G,n)\|_0 \simeq \|\|X\|_n \to K(G,n)\|_0 = H^n(\|X\|_n, G)$$

The fact that this equivalence is a homomorphism is trivial. □

Using this we can easily compute the zeroth cohomology group of a 0-connected type (and hence, any n-connected type with $n \geq 0$).

Proposition 46. *For any 0-connected type X, we have $H^0(X, G) \cong G$.*

Proof. By Lemma 45, we have $H^0(X,G) = H^0(\|X\|_0, G)$. But $\|X\|_0 \simeq \mathbb{1}$ since X is 0-connected. Thus, by Proposition 44, the zeroth cohomology group is isomorphic to G. □

5.1 Spheres

We now have what we need to compute the cohomology of the n-sphere. Let us start with $n = 1$.

Proposition 47. $H^1(\mathbb{S}^1, G) \cong G$.

Proof. A function $\mathbb{S}^1 \to K(G,1)$ is uniquely determined by its action on the canonical base point and the canonical loop. Hence, it corresponds to choices of a point $x : K(G,1)$ and a loop $x = x$. In other words, $(\mathbb{S}^1 \to K(G,1)) \simeq \Sigma_{x:K(G,1)}(x = x)$. But $K(G,1)$ is homogeneous, so $(x = x) \simeq \Omega K(G,1)$. Thus, we have

$$\begin{aligned} H^1(\mathbb{S}^1, G) &= \|\mathbb{S}^1 \to K(G,1)\|_0 \\ &\simeq \|\Sigma_{x:K(G,1)}(x = x)\|_0 \\ &\simeq \|K(G,1) \times \Omega K(G,1)\|_0 \\ &\simeq \|K(G,1)\|_0 \times G \end{aligned}$$

Now, $K(G,1)$ is 0-connected, so $\|K(G,1)\|_0$ vanishes. Thus, we have $H^1(\mathbb{S}^1, G) \simeq G$.

We need to verify that this isomorphism is a homomorphism. It is easier to show that its inverse is. Let us call it ϕ. By definition, $\phi(g) = |f_g|$ where $f_g : \mathbb{S}^1 \to K(G,1)$ is the map sending base to $\star$ and loop to $\mathsf{loop}_k\, g$. Thus, we only need to verify that for $g_1, g_2 : G$ and $x : \mathbb{S}^1$, we have $f_{g_1}(x) +_k f_{g_2}(x) = f_{g_1+g_2}(x)$. We do this by induction on x. When x is base, the goal holds by refl. For the

action on loop, we need to show that

$$\mathsf{ap}^2_{+_k}(\mathsf{loop}_k\, g_1, \mathsf{loop}_k\, g_2) = \mathsf{loop}_k\, (g_1 + g_2)$$

which holds by Proposition 16 and functoriality of loop_k . □

It is easy to generalise this to get all non-trivial cohomology groups of $\mathbb{S}^n$ for $n \geq 1$.

Proposition 48. *For $n \geq 1$, we have $H^n(\mathbb{S}^n, G) \cong G$.*

Proof. The case $n = 1$ is Proposition 47. For $n > 1$, we simply apply the suspension isomorphism inductively. □

The vanishing groups can also be handled directly.

Proposition 49. *For $n \neq m$ and $n \geq 1$ have $H^n(\mathbb{S}^m, G) \cong \mathbb{1}$.*

Proof. Let us first consider the case $n > m$. We induct on m. When $m = 0$, we have

$$H^n(\mathbb{S}^0) = \|\mathbb{S}^0 \to K(G,n)\|_0 \simeq \|K(G,n) \times K(G,n)\|_0$$

which vanishes since $K(G,n)$ is $(n-1)$-connected and $n > m = 0$. For larger m, we know by suspension that $H^n(\mathbb{S}^m) \cong H^{n-1}(\mathbb{S}^{m-1})$ which vanishes by the induction hypothesis.

When $n < m$ we have

$$H^n(\mathbb{S}^m, G) \simeq H^n(\|\mathbb{S}^m\|_n, G)$$

which vanishes because $\mathbb{S}^m$ is $n \leq (m-1)$-connected and $n \geq 1$. □

Knowing these groups, the cohomology ring of the spheres is easily computed.

Proposition 50. *Given any ring R and $n \geq 1$, we have $H^*(\mathbb{S}^n, R) \cong R[x]/(x^2)$.*

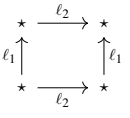
Proof. The only non-trivial cohomology groups are $H^0(\mathbb{S}^n, R) \cong H^n(\mathbb{S}^n, R) \cong R$. Thus, the cup product can only be non-trivial in dimensions $0 \times n$ and $n \times 0$. In these dimension, the cup product is simply left respectively right R-multiplication. This gives the desired isomorphism by letting constants in $H^*(\mathbb{S}^n, R)$ represent elements coming from $H^0(\mathbb{S}^n, R)$ and elements of degree one represent elements coming from $H^n(\mathbb{S}^n, R)$. □

5.2 The torus

Now that we know the cohomology of spheres, let us investigate something slightly more advanced: the torus.

Definition 51 (The torus)**.** *We define $\mathbb{T}^2$ as the HIT generated by the following constructors:*

- $\star : \mathbb{T}^2$
- $\ell_1, \ell_2 : \star = \star$
- $\mathsf{sq} : l_2 =^{l_1}_{l_1} l_2$ *– that is, a filler of the square:*



It is well-known that $\mathbb{T}^2 \simeq \mathbb{S}^1 \times \mathbb{S}^1$.[h]. We will see that this fact makes the computation of the cohomology groups of $\mathbb{T}^2$ rather direct. We follow the proof strategy of Brunerie et al. (2022, Propositions 20–21), which directly translates from integral cohomology to our setting.

Proposition 52.

$$H^n(\mathbb{T}^2, G) \cong \begin{cases} G & \text{if } n = 0 \text{ or } n = 2 \\ G \times G & \text{if } n = 1 \\ \mathbb{1} & \text{otherwise} \end{cases}$$

Proof. The case when $n = 0$ follows by Proposition 46 since $\mathbb{T}^2$ is 0-connected. For the case $n \geq 1$, let us inspect the underlying function space of $H^n(\mathbb{T}^2, G)$. We have

$$\begin{aligned} (\mathbb{T}^2 \to K(G,n)) &\simeq (\mathbb{S}^1 \times \mathbb{S}^1 \to K(G,n)) \\ &\simeq \mathbb{S}^1 \to (\mathbb{S}^1 \to K(G,n)) \\ &\simeq \sum_{f : \mathbb{S}^1 \to K(G,n)} (f = f) \\ &\simeq \sum_{f : \mathbb{S}^1 \to K(G,n)} ((x : \mathbb{S}^1) \to \Omega(K(G,n), f(x))) \\ &\simeq (\mathbb{S}^1 \to K(G,n)) \times (\mathbb{S}^1 \to \Omega(K(G,n))) \\ &\simeq (\mathbb{S}^1 \to K(G,n)) \times (\mathbb{S}^1 \to K(G,n-1)) \end{aligned}$$

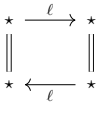
where the step from line 3 to 4 is function extensionality and the step from line 4 to 5 follows by homogeneity of $K(G,n)$. Under set truncation, the last type in this chain of equivalences is simply $H^n(S^1, G) \times H^{n-1}(S^1, G)$, which we know from Section 5.1 is simply $G \times G$ when $n = 1$, G when $n = 2$, and trivial for higher n. The fact that this map is a homomorphism follows by a similar argument to that of the corresponding statement in the proof of Proposition 47 . □

5.3 The real projective plane and the Klein bottle

Let us consider two somewhat more complex (but closely related) examples: the real projective plane and the Klein bottle. In HoTT, we define the real projective plane, $\mathbb{R}P^2$, and the Klein bottle, K^2, by HITs corresponding to their usual folding diagrams:

Definition 53 (The real project plane)**.** *We define* $\mathbb{R}P^2$ *as the HIT generated by the following constructors:*

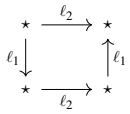
- $\star : \mathbb{R}P^2$
- $\ell : \star = \star$
- $\mathsf{sq} : \ell^{-1} = \ell$ *– that is, a filler of the square:*



[h] In plain HoTT, this was first fully proved by Sojakova (2016). In cubical type theory, this fact is a triviality (Mörtberg and Pujet, 2020, Section 3).

Definition 54 (The Klein bottle)**.** *We define K^2 as the HIT generated by the following constructors:*

- $\star : K^2$
- $\ell_1, \ell_2 : \star = \star$
- $\mathsf{sq} : l_2 \overset{l_1^{-1}}{=}_{l_1} l_2$ *– that is, a filler of the square:*



Let us first tackle the cohomology of $\mathbb{R}P^2$. It is an easy lemma that $\mathbb{R}P^2$ is 0-connected, and hence $H^0(\mathbb{R}P^2, G) \cong G$. In order to compute its higher cohomology groups, we will need to introduce some new constructions: for an abelian group G and an integer $n \geq 0$, let $G[n]$ denote the n-torsion subgroup of G or, in other words, the kernel of the map $n \cdot (-) : G \to G$. Let G/n denote the cokernel of the same homomorphism – that is, the group $G/\sim$ where $\sim$ is the relation $x^n = 0$. We denote the quotient map $G \to G/n$ by $[-]$.

Proposition 55. $H^1(\mathbb{R}P^2, G) \cong G[2]$.

Proof. The elimination principle for $\mathbb{R}P^2$ tells us that

$$(\mathbb{R}P^2 \to K(G,1)) \simeq \sum_{x:K(G,1)} \sum_{p:x=x} (p^{-1} = p) \simeq \sum_{x:K(G,1)} \sum_{p:x=x} (p \cdot p = \mathsf{refl})$$

Since $K(G,n)$ is homogeneous, this is equivalent to the type

$$K(G,1) \times \sum_{p:\Omega K(G,1)} (p \cdot p = \mathsf{refl})$$

which under the equivalence $\Omega K(G,1) \simeq G$ is equivalent to the type

$$K(G,1) \times \sum_{g:G} (g + g = 0_G)$$

or, in other words, $K(G,1) \times G[2]$. Thus, we get

$$\begin{aligned} H^1(\mathbb{R}P^2, G) = \|\mathbb{R}P^2 \to K(G,1)\|_0 &\simeq \|K(G,1) \times G[2]\|_0 \\ &\simeq \|K(G,1)\|_0 \times G[2] \\ &\simeq G[2] \end{aligned}$$

We verify that the inverse of this equivalence is a homomorphism. The inverse $\psi : G[2] \to H^1(\mathbb{R}P^2, G)$ is given by $\psi(g) = f_g$ where $f_g : \mathbb{R}P^2 \to K(G,1)$ sends $\star$ to 0_k and ℓ to $\mathsf{loop}_k\, g$. We do not need to worry about its action on sq: for truncation reasons this is not data but a property. We need to verify that $f_{g_1+g_2}(x) = f_{g_1}(x) +_k f_{g_2}(x)$ for $x : \mathbb{R}P^2$ and $g_1, g_2 : G$. When x is $\star$, this holds by refl. For the action on ℓ, we need to verify that

$$\mathsf{loop}_k\,(g_1 + g_2) = \mathsf{ap}^2_{+_k}(\mathsf{loop}_k\, g_1, \mathsf{loop}_k\, g_2)$$

which we already verified in the proof of Proposition 47. This concludes the proof. □

Proposition 56. $H^2(\mathbb{R}P^2, G) \cong G/2$.

Proof. Like in the proof of Proposition 55, we have

$$\begin{aligned}(\mathbb{R}P^2 \to K(G,2)) &\simeq \sum_{x:K(G,2)} \sum_{p:x=x} p \cdot p = \mathsf{refl} \\ &\simeq K(G,2) \times \sum_{p:\Omega K(G,2)} p \cdot p = \mathsf{refl} \\ &\simeq K(G,2) \times \sum_{x:K(G,1)} x +_k x = 0_k\end{aligned}$$

Under set truncation, the $K(G,2)$-component vanishes which leaves us with the type $\|\sum_{x:K(G,1)} x +_k x = 0_k\|_0$. Let us construct a map $\phi : \|\sum_{x:K(G,1)} x +_k x = 0_k\|_0 \to G/2$ by means of a curried map $\phi_x : x +_k x = 0_k \to G/2$ depending on $x : K(G,1)$. We may define it by the set-elimination rule for $K(G,1)$. When x is 0_k, what is desired is a map $\phi_\star : \Omega K(G,1) \to G/2$. This map may simply be defined by $\phi_\star := [-] \circ \sigma_0^{-1}$. We now need to describe the action on $\mathsf{loop}_k\, g$. This amounts to providing a path $[g] = [g' + g + g']$, which of course trivially exists in $G/2$. This defines $\phi(|\, x, p\,|) := \phi_x(p)$.

The inverse $\psi : G/2 \to \|\Sigma_{x:K(G,1)}(x +_k x = 0_k)\|_0$ is defined on canonical elements by $\psi[g] = (0_k, \mathsf{loop}_k\, g)$. In order to show that this is well-defined, we need to show that

$$|\,(0_k, \mathsf{refl})\,| = |\,(0_k, \mathsf{loop}_k\,(g +_G g))\,|$$

This is done component-wise. For first component, we need to provide a path $0_k = 0_k$. In fact, the right choice here it *not* refl but instead $\mathsf{loop}_k\, g$. Showing that the second components agree now amounts to providing a filler of the square

$$\begin{array}{ccc} 0_k & \xrightarrow{\mathsf{loop}_k\,(g+_G g)} & 0_k \\ {\scriptstyle (\mathsf{ap}^2_{+_k}(\mathsf{loop}_k\, g, \mathsf{loop}_k\, g))^{-1}}\uparrow & & \Big\| {\scriptstyle \mathsf{refl}} \\ 0_k & \xrightarrow[\mathsf{refl}]{} & 0_k \end{array}$$

which, in turn, is equivalent to proving that $\mathsf{loop}_k\,(g +_G g) = \mathsf{ap}^2_{+_k}(\mathsf{loop}_k\, g, \mathsf{loop}_k\, g)$, which we have already done in the proofs of Propositions 47 and 55.

Finally, let us prove that the maps cancel. Verifying that $\phi(\psi(x)) = x$ for $x : G/2$ is very direct: since the statement is a proposition, it suffices to to this when x is on the form $[g]$. In this case, we have $\phi(\psi[g]) := \phi_{0_k}(\mathsf{loop}_k\, g) := \sigma_0^{-1}(\sigma_0(g)) = g$.

For the other direction, the fact that we are proving a proposition means that it suffices to show the cancellation for elements of $\|\Sigma_{x:K(G,1)}(x +_k x = 0_k)\|_0$ on the form $|\,(0_k, p)\,|$ with $p : \Omega K(G,1)$. We get $\psi(\phi\,|\,0_k, p\,|) := \psi(\sigma_0^{-1}(p)) = |\,0_k, \sigma_0(\sigma_0^{-1}(p))\,| = |\,0_k, p\,|$. Thus, we have shown:

$$\begin{aligned}H^2(\mathbb{R}P^2, G) &\simeq \left\| K(G,2) \times \sum_{x:K(G,1)} x +_k x = 0_k \right\|_0 \\ &\simeq \left\| \sum_{x:K(G,1)} x +_k x = 0_k \right\|_0 \\ &\simeq G/2\end{aligned}$$

The argument for why this equivalence is a homomorphism is technical but similar to the argument given in previous proofs. □

Proposition 57. *$H^n(\mathbb{R}P^2, G)$ is trivial for $n > 2$.*

Proof. As we have seen in the two previous proofs:

$$H^n(\mathbb{R}P^2, G) \simeq \left\| K(G,n) \times \sum_{x:K(G,n-1)} x +_k x = 0_k \right\|_0$$
$$\simeq \left\| \sum_{x:K(G,n-1)} x +_k x = 0_k \right\|_0$$

We may move in the truncation one step further to get

$$\left\| \sum_{x:K(G,n-1)} \|x +_k x = 0_k\|_0 \right\|_0$$

Since $n > 2$, any path space over $K(G, n-1)$ is (at least) 0-connected. Thus, the truncation kills the space $(x +_k x = 0_k)$. Hence, the above type is simply $\|K(G, n-1)\|_0$, which also vanishes due to connectedness. □

This gives us all the cohomology groups of $\mathbb{R}P^2$. We will now see how this, in a rather direct way, gives us all the cohomology groups of K^2 as well. Trivially, $H^0(K^2, G) \cong G$ for connectedness reasons.

Proposition 58. $H^n(K^2, G) \cong \begin{cases} H^n(\mathbb{R}P^2, G) & n \neq 1 \\ G \times H^1(\mathbb{R}P^2, G) & n = 1 \end{cases}$

Proof. Consider the function space $(K^2 \to K(G,n))$:

$$(K^2 \to K(G,n)) \simeq \sum_{x:K(G,n)} \sum_{p,q:x=x} (p \cdot q \cdot p = q)$$

Again, we may use homogeneity of $K(G,n)$ to show that this is equivalent to the type

$$K(G,n) \times \sum_{p,q:\Omega K(G,n)} (p \cdot q \cdot p = q)$$

We now consider the path space $p \cdot q \cdot p = q$ for $p, q : \Omega K(G,n)$. By commutativity of $\Omega\, K(G,n)$, composing both sides by q^{-1} will cancel out q everywhere. Thus, the type is simply $p \cdot p = \mathsf{refl}$. Hence, we have

$$H^n(K^2, G) :\equiv \|K^2 \to K(G,n)\|_0$$
$$\simeq \left\| K(G,n) \times \Omega K(G,n) \times \sum_{p:\Omega K(G,n)} (p \cdot p \equiv \mathsf{refl}) \right\|_0$$

Excluding the $\Omega K(G,n)$ component, this is precisely the characterisation of $H^n(\mathbb{R}P^2, G)$. Thus, the above type is equivalent to

$$\|\Omega K(G,n)\|_0 \times H^n(\mathbb{R}P^2, G)$$

The factor $\|\Omega K(G,n)\|_0$ vanishes for $n > 1$. When $n = 1$, we have $\|\Omega K(G,1)\|_0 \simeq \|G\|_0 \simeq G$. Verifying that this is a homomorphism is straightforward but somewhat technical. □

Computing the integral cohomology ring of $\mathbb{R}P^2$ is straightforward. Naturally, the following proposition easily generalises to $H^*(\mathbb{R}P^2, R)$ for any 2-torsion free ring R.

Proposition 59. $H^*(\mathbb{R}P^2, \mathbb{Z}) \cong \mathbb{Z}[x]/(2x, x^2)$.

Proof. By Proposition 55, we have $H^1(\mathbb{R}P^2, \mathbb{Z}) \cong \mathbb{Z}[2]$. Since $\mathbb{Z}$ has no non-trivial torsion elements, this tells us that $H^1(\mathbb{R}P^2, \mathbb{Z})$ is trivial. Hence, we only have as non-trivial cohomology

groups the groups $H^0(\mathbb{R}P^2, \mathbb{Z}) \cong \mathbb{Z}$ and $H^2(\mathbb{R}P^2, \mathbb{Z}) \cong \mathbb{Z}/2\mathbb{Z}$. The proof now proceeds just like that of Proposition 50, representing elements from $H^0(\mathbb{R}P^2, \mathbb{Z})$ and elements from $H^2(\mathbb{R}P^2, \mathbb{Z})$ respectively by degree-0 and degree-1 elements in $H^*(\mathbb{R}P^2, \mathbb{Z})$. The $2x$-quotient comes from the 2-torsion in $H^2(\mathbb{R}P^2, \mathbb{Z})$. □

Let us now turn to the Klein bottle. For the coming results, the following construction will be crucial.

Definition 60. *There is a homotopy*

$$\mathsf{kill}\text{–}\Delta : (p : \Omega K(G, n)) \to \mathsf{ap}^2_{\smile_k}(p, p) = 0_k$$

defined by the composite path

$$\mathsf{ap}^2_{\smile_k}(p, p) \xrightarrow{\mathsf{ap}^2_{\smile_k}\text{-}\mathsf{funct}(p,p)} \mathsf{ap}_{(-)\smile_k 0_k}(p) \cdot \mathsf{ap}_{0_k \smile_k (-)}(p) \xrightarrow{h(p)} \mathsf{refl}$$

Here $h(p)$ is given by pointwise application of the right and left annihilation laws for $\smile_k$ using that they agree on 0_k.

We will rely on this construction to kill off certain cup products on $\mathbb{R}P^2$and K^2. For instance, it makes the following computation easy.

Proposition 61. $H^*(K^2, \mathbb{Z}) \cong \mathbb{Z}[x, y]/(2y, x^2, y^2, xy)$.

Proof. Using Proposition 58 and the preceding results, we know that the only non-trivial cohomology groups of K^2 are

$$H^0(K^2, \mathbb{Z}) \cong \mathbb{Z}$$
$$H^1(K^2, \mathbb{Z}) \cong \mathbb{Z} \times \mathbb{Z}[2] \cong \mathbb{Z}$$
$$H^2(K^2, \mathbb{Z}) \cong \mathbb{Z}/2\mathbb{Z}$$

Consider first the ring $\mathbb{Z}[x, y]$. Using the above isomorphisms, we may naively let degree-0 elements in $\mathbb{Z}[x, y]$ represent elements from $H^0(K^2, \mathbb{Z})$, degree-1 elements with x-term represent elements from $H^1(K^2, \mathbb{Z})$ and degree-1 elements with y term represent elements from $H^2(K^2, \mathbb{Z})$. Under this identification, we see that the relations described by the ideal $(2y, xy, y^2)$ are satisfied. Hence, we have a map $\mathbb{Z}[x, y]/(2y, xy, y^2) \to H^1(K^2, \mathbb{Z})$. In order to lift this to an isomorphism $\mathbb{Z}[x, y]/(2y, x^2, y^2, xy) \cong H^*(K^2, \mathbb{Z})$, we need to show that the cup product is trivial in dimension 1×1. It suffices to show that it vanishes on generators. We can easily define a map $\alpha : K^2 \to K(1, \mathbb{Z})$ such that $|\alpha|$ generates $H^1(K^2, \mathbb{Z})$. We define it by

$$\alpha(\star) := 0_k$$
$$\mathsf{ap}_\alpha(\ell_1) := \mathsf{refl}$$
$$\mathsf{ap}_\alpha(\ell_2) := \mathsf{loop}_k\, 1$$

where, $1 : \mathbb{Z}$ denotes the generator. The final step of the definition – the action of α on sq – corresponds to providing a proof that $\mathsf{loop}_k\, 1 = \mathsf{loop}_k\, 1$, which we do by refl. Let $x : K^2$. We show that $\alpha(x) \smile_k \alpha(x) = 0_k$ by induction on x. We prove $\alpha(\star) \smile_k \alpha(\star) = 0_k$ by refl and $\mathsf{ap}^2_{\smile_k}(\mathsf{ap}_\alpha(\ell_2), \mathsf{ap}_\alpha(\ell_2)) = \mathsf{refl}$ by refl. For $\mathsf{ap}^2_{\smile_k}(\mathsf{ap}_\alpha(\ell_1), \mathsf{ap}_\alpha(\ell_1))$, we need to provide a path $\mathsf{ap}^2_{\smile_k}(\mathsf{loop}_k\, 1, \mathsf{loop}_k\, 1) = \mathsf{refl}$, which we give by $\mathsf{kill}\text{–}\Delta(\mathsf{loop}_k\, 1)$. Finally, for the action of sq, we

need to provide a filler of the square

$$\begin{array}{ccc} \mathsf{ap}^2_{\smile_k}(\mathsf{loop}_k\,1, \mathsf{loop}_k\,1) & \xrightarrow{\mathsf{kill}-\Delta(\mathsf{loop}_k\,1)} & \mathsf{refl} \\ \Big\| \mathsf{refl} & & \Big\| \mathsf{refl} \\ \mathsf{ap}^2_{\smile_k}(\mathsf{loop}_k\,1, \mathsf{loop}_k\,1) & \xrightarrow[\mathsf{kill}-\Delta(\mathsf{loop}_k\,1)]{} & \mathsf{refl} \end{array}$$

which is trivial. Thus, we have shown, in particular, that $|\,\alpha\,|\smile|\,\alpha\,|$ is trivial, which means that $\smile$ vanishes everywhere in dimension 1×1. This concludes the proof. □

Let us now turn to the $\mathbb{Z}/2\mathbb{Z}$-cohomology of K^2 and $\mathbb{R}P^2$. When working in $\mathbb{Z}/2\mathbb{Z}$-coefficients, it turns out that we will need to investigate $\mathsf{kill}-\Delta$ further. In particular, it turns out that applying $\mathsf{kill}-\Delta$ may give rise to non-trivial homotopies. In particular, the twist in $\mathbb{R}P^2$and K^2 will force us to investigate how much $\mathsf{kill}-\Delta$ commutes with path inversion in the sense that there exists a loop $q:\Omega^2K(\mathbb{Z}/2\mathbb{Z},2n)$ solving the following square filling problem

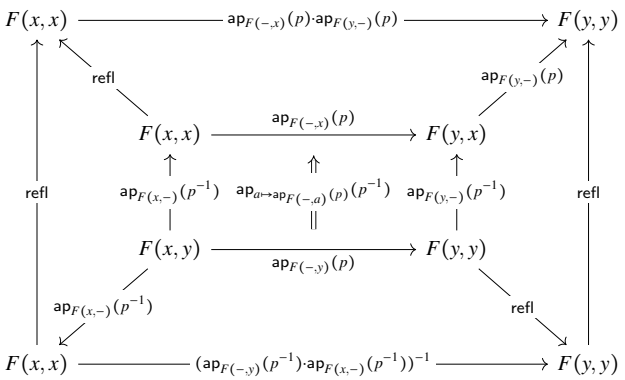
$$\begin{array}{ccc} \mathsf{ap}^2_{\smile_k}(p,p) & \xrightarrow{\mathsf{kill}-\Delta(p)} & \mathsf{refl} \\ \Big\downarrow & & \Big\downarrow q \\ (\mathsf{ap}^2_{\smile_k}(p^{-1},p^{-1}))^{-1} & \xrightarrow[\mathsf{ap}_{(-)^{-1}}(\mathsf{kill}-\Delta(p^{-1}))]{} & \mathsf{refl} \end{array}$$

We remark that the only case of interest to us is the case $n=1$, due to the vanishing of higher homotopy groups. We will need the following construction.

Definition 62. *Given two points $x,y:A$ and a binary function function $F:A\to A\to B$, we construct the homotopy*

$$\mathsf{Res}_F:(p:x=y)\to(\mathsf{ap}_{F(-,y)}(p^{-1})\cdot\mathsf{ap}_{F(x,-)}(p^{-1}))^{-1}=\mathsf{ap}_{F(-,x)}(p)\cdot\mathsf{ap}_{F(y,-)}(p)$$

by the outer square of the following composite square



where all the outermost squares are given by their obvious fillers.

The following lemma is immediate by path induction.

Lemma 63. *For any point $x : A$ and $F : A \to A \to B$, we have* $\mathsf{Res}_F(\mathsf{refl}_x) = \mathsf{refl}$.

The usefulness of Res is that it measures how much ap^2_F-funct fails commute with path inversion, as made clear by the following lemma.

Lemma 64. *Given a binary function $F : A \to A \to B$ and a path $p : x =_A y$, we have a filler*

$$\begin{array}{ccc} \mathsf{ap}^2_F(p,p) & \xrightarrow{\mathsf{ap}^2_F\text{-}\mathsf{funct}(p,p)} & \mathsf{ap}_{F(-,\star_A)}(p) \cdot \mathsf{ap}_{F(\star_A,-)}(p) \\ \uparrow & & \uparrow{\scriptstyle \mathsf{Res}_F(p)} \\ (\mathsf{ap}^2_F(p^{-1},p^{-1}))^{-1} & \xrightarrow[\mathsf{ap}_{(-)^{-1}}(\mathsf{ap}^2_F\text{-}\mathsf{funct}(p^{-1},p^{-1}))]{} & (\mathsf{ap}_{F(-,\star_A)}(p^{-1}) \cdot \mathsf{ap}_{F(\star_A,-)}(p^{-1}))^{-1} \end{array}$$

where the left-most path is the obvious coherence (defined by path induction in Book HoTT and simply by refl *in cubical type theory).*

Proof. Path induction on p. □

What is rather surprising is that $\mathsf{Res}_F(p)$ is not just a mere coherence path: it may be homotopically non-trivial when p is a loop. In particular, when p is $\mathsf{loop}_k\, 1 : \Omega K(G,1)$ and F is the cup product $\smile_k : K(\mathbb{Z}/2\mathbb{Z},1) \to K(\mathbb{Z}/2\mathbb{Z},1) \to K(\mathbb{Z}/2\mathbb{Z},2)$, the inner square in Definition 62 is, modulo cancellation laws, precisely the two-cell $\sigma^2(1) : \Omega^2 K(\mathbb{Z}/2\mathbb{Z},2)$ (here σ^2 is shorthand for $\mathsf{ap}_{\sigma_1} \circ \sigma_0 : K(\mathbb{Z}/2\mathbb{Z},0) \to \Omega^2 K(\mathbb{Z}/2\mathbb{Z},2)$). This follows directly by construction of the cup product. Hence we arrive at the following lemma, which is key to analysing the behaviour of the cup product on $\mathbb{R}P^2$ and K^2.

Lemma 65. *We have a filler of the following square*

$$\begin{array}{ccc} \mathsf{ap}^2_{\smile_k}(\mathsf{loop}_k\, 1, \mathsf{loop}_k\, 1) & \xrightarrow{\mathsf{kill}\text{-}\Delta(\mathsf{loop}_k\, 1)} & \mathsf{refl} \\ \downarrow & & \downarrow{\scriptstyle \sigma^2(1)} \\ (\mathsf{ap}^2_{\smile_k}((\mathsf{loop}_k\, 1)^{-1}, (\mathsf{loop}_k\, 1)^{-1}))^{-1} & \xrightarrow[\mathsf{ap}_{(-)^{-1}}(\mathsf{kill}\text{-}\Delta(\mathsf{loop}_k\, 1^{-1}))]{} & \mathsf{refl} \end{array}$$

We may now characterise the cup product on the cohomology groups of $\mathbb{R}P^2$ and K^2 with $\mathbb{Z}/2\mathbb{Z}$ coefficients. In fact, Lemma 65 is the only technical lemma we will need, and we do not need to rely on any other advanced theorems/constructions. Let us start with $\mathbb{R}P^2$.

Proposition 66. *The following diagram commutes*

$$\begin{array}{ccc} \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} & \xrightarrow{\mathsf{mult}} & \mathbb{Z}/2\mathbb{Z} \\ \downarrow{\scriptstyle \wr} & & \downarrow{\scriptstyle \wr} \\ H^1(\mathbb{R}P^2, \mathbb{Z}/2\mathbb{Z}) \times H^1(\mathbb{R}P^2, \mathbb{Z}/2\mathbb{Z}) & \xrightarrow{\smile} & H^2(\mathbb{R}P^2, \mathbb{Z}/2\mathbb{Z}) \end{array}$$

Proof. It is enough to show that $|\alpha| \smile |\alpha| = |\beta|$ for generators $\alpha : \mathbb{R}P^2 \to K(\mathbb{Z}/2\mathbb{Z}, 1)$ and $\beta : \mathbb{R}P^2 \to K(\mathbb{Z}/2\mathbb{Z}, 2)$. We define α by

$$\alpha(\star) := 0_k$$
$$\mathsf{ap}_\alpha(\ell) := \mathsf{loop}_k\, 1$$
$$\mathsf{ap}_{\mathsf{ap}_\alpha}(\mathsf{sq}) := Q(\mathsf{loop}_k\, 1)$$

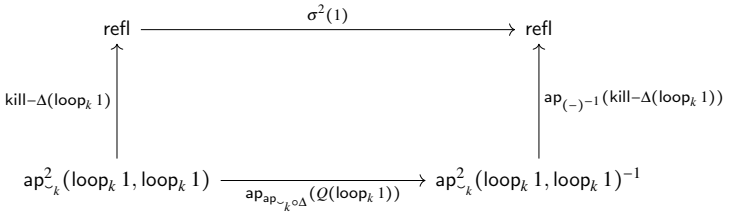
where $Q : (p : \Omega K(\mathbb{Z}/2\mathbb{Z}, 1) \to p = p^{-1}$ is given by the 2-torsion in $K(\mathbb{Z}/2\mathbb{Z}, n)$ (its explicit construction is irrelevant for our purposes). We define β by

$$\alpha(\star) := 0_k$$
$$\mathsf{ap}_\beta(\ell) := \mathsf{refl}$$
$$\mathsf{ap}_{\mathsf{ap}_\beta}(\mathsf{sq}) := \sigma^2(1)$$

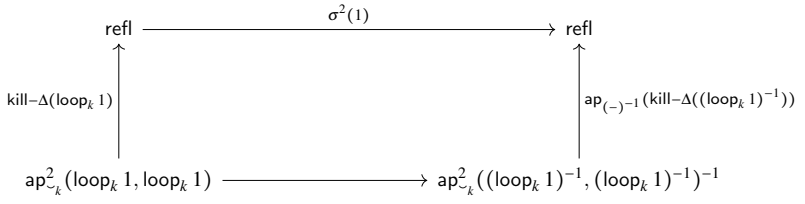
where $\sigma^2 : \mathbb{Z}/2\mathbb{Z} \xrightarrow{\sim} \Omega^2 K(\mathbb{Z}/2\mathbb{Z}, 2)$. The fact that α and β define generators on cohomology is straightforward by the constructions in the proofs of Propositions 55 and 56. We now provide a homotopy $h : (x : \mathbb{R}P^2) \to \alpha(x) \smile_k \alpha(x) = \beta(x)$. We define

$$h(\star) := \mathsf{refl}$$
$$\mathsf{ap}_h(\ell) := \mathsf{kill}{-}\Delta(\mathsf{loop}_k\, 1)$$

For the action of h on sq, we need to provide a filler of the following square of paths.



The bottom path acts on the right-hand path by moving in a path inversion around $\mathsf{loop}_k\, 1$. Hence, our new square filling problem is



But this is precisely Lemma 65, and we are done. □

As an immediate corollary, we get the cohomology ring of $\mathbb{R}P^2$ with $\mathbb{Z}/2\mathbb{Z}$-coefficients.

Proposition 67. $H^*(\mathbb{R}P^2, \mathbb{Z}/2\mathbb{Z}) \cong \mathbb{Z}/2\mathbb{Z}[x]/(x^3)$.

Proof. We simply let elements in $H^n(\mathbb{R}P^2, \mathbb{Z}/2\mathbb{Z})$ correspond to degree-n elements in $H^*(\mathbb{R}P^2, \mathbb{Z}/2\mathbb{Z})$. By Proposition 66, we know that the multiplicative structure on $H^*(\mathbb{R}P^2, \mathbb{Z}/2\mathbb{Z})$ is regular $\mathbb{Z}/2\mathbb{Z}$-multiplication, which proves statement. □

Let us now turn to the Klein bottle. Although the cohomology ring structure is different, we may still make use of the above. We start by considering its multiplicative structure. First, we construct the generators $\alpha, \beta : K^2 \to K(\mathbb{Z}/2\mathbb{Z}, 1)$ by

$$\begin{aligned} \alpha(\star) &:= 0_k & \beta(\star) &:= 0_k \\ \mathsf{ap}_\alpha(\ell_1) &:= \mathsf{loop}_k\, 1 & \mathsf{ap}_\beta(\ell_1) &:= \mathsf{refl} \\ \mathsf{ap}_\alpha(\ell_2) &:= \mathsf{refl} & \mathsf{ap}_\beta(\ell_2) &:= \mathsf{loop}_k\, 1 \\ \mathsf{ap}_{\mathsf{ap}_\alpha}(\mathsf{sq}) &:= \mathsf{flip}(\mathsf{refl}_{\mathsf{loop}_k\, 1}) & \mathsf{ap}_{\mathsf{ap}_\beta}(\mathsf{sq}) &:= \mathsf{refl}_{\mathsf{loop}_k\, 1} \end{aligned}$$

It is immediate by construction that the isomorphism $H^1(K^2, \mathbb{Z}/2\mathbb{Z}) \cong \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ takes $|\,\alpha\,|$ to $(1,0)$ and $|\,\beta\,|$ to $(0,1)$. The single generator of $H^2(K^2, \mathbb{Z}/2\mathbb{Z})$ is given by $\gamma : K^2 \to K(\mathbb{Z}/2\mathbb{Z}, 2)$:

$$\begin{aligned} \gamma(\star) &:= 0_k \\ \mathsf{ap}_\gamma(\ell_1) &:= \mathsf{refl} \\ \mathsf{ap}_\gamma(\ell_2) &:= \mathsf{refl} \\ \mathsf{ap}_{\mathsf{ap}_\gamma}(\mathsf{sq}) &:= \sigma^2(1) \end{aligned}$$

It again follows by construction that the isomorphism $H^2(K^2, \mathbb{Z}/2\mathbb{Z}) \cong \mathbb{Z}/2\mathbb{Z}$ takes $|\,\gamma\,|$ to 1.

Proposition 68. *We have the following relations in* $H^*(K^2, \mathbb{Z}/2\mathbb{Z})$

$$|\,\alpha\,|^2 = \gamma \qquad |\,\alpha\,| \smile |\,\beta\,| = \gamma \qquad |\,\beta\,|^2 = 0_h$$

Proof. Let us deal with the identities from right to left, in increasing difficulty. Showing that $\beta^2 = 0_h$ is easy. First, we construct a homotopy $f : (x : K^2) \to \beta(x) \smile_k \beta(x) = 0_k$ by induction on x.

$$\begin{aligned} f(\star) &:= \mathsf{refl} \\ \mathsf{ap}_f(\ell_1) &:= \mathsf{refl} \\ \mathsf{ap}_f(\ell_2) &:= \mathsf{kill}{-}\Delta(\mathsf{loop}_k\, 1) \end{aligned}$$

For the action on sq, we are reduced to simply providing a path $\mathsf{kill}{-}\Delta(\mathsf{loop}_k\, 1) = \mathsf{kill}{-}\Delta(\mathsf{loop}_k\, 1)$, which we can by refl. This gives us, in particular, that $|\,\beta\,|^2 = 0_h$.

Let us now provide a homotopy $g : (x : K^2) \to \alpha(x) \smile_k \beta(x) = \gamma(x)$. We again proceed by K^2-induction.

$$\begin{aligned} g(\star) &:= \mathsf{refl} \\ \mathsf{ap}_g(\ell_1) &:= \mathsf{refl} \\ \mathsf{ap}_g(\ell_2) &:= \mathsf{refl} \end{aligned}$$

The reason that refl works for the action on ℓ_1 and ℓ_2 is because $\mathsf{ap}_{0_k \smile_k (-)}(\mathsf{loop}_k\, 1)$ and $\mathsf{ap}_{(-) \smile_k 0_k}(\mathsf{loop}_k\, 1)$ by definition reduce to refl_{0_k}. The final step amount to showing that

$$\mathsf{ap}_{x \mapsto \mathsf{ap}_{y \mapsto \alpha(x) \smile_k \beta(y)}(\mathsf{loop}_k\, 1)}(\mathsf{loop}_k\, 1) = \sigma^2(1)$$

which is easy to see by unfolding the definition of $\smile_k$. Thus, we have $|\,\alpha\,| \smile |\,\beta\,| = |\,\gamma\,|$.

Finally, let us verify that $|\,\alpha\,|^2 = |\,\gamma\,|$. We construct a homotopy $h : (x : K^2) \to \alpha(x) \smile_k \alpha(x) = \gamma(x)$ by

$$\begin{aligned} h(\star) &:= \mathsf{refl} \\ \mathsf{ap}_h(\ell_1) &:= \mathsf{kill}{-}\Delta(\mathsf{loop}_k\, 1) \\ \mathsf{ap}_h(\ell_2) &:= \mathsf{refl} \end{aligned}$$

For the action on sq, we end up having to fill another square (or rather: another degenerate cube). Fortunately for us, this filling problem can easily be reduced to that of filling the first square appearing in the proof of Proposition 66, which we have already filled. Thus $|\alpha|^2 = |\gamma|$. □

Corollary 69. $H^*(K^2, \mathbb{Z}/2\mathbb{Z}) \cong \mathbb{Z}/2\mathbb{Z}[x,y]/(x^3, y^2, xy + x^2)$

Proof. By Proposition 68, we obtain the isomorphism by mapping the generators $1 : H^0(K^2, \mathbb{Z}/2\mathbb{Z})$, $\alpha, \beta : H^1(K^2, \mathbb{Z}/2\mathbb{Z})$ and $\gamma : H^2(K^2, \mathbb{Z}/2\mathbb{Z})$ to 1, x, y and x^2 respectively. □

5.4 The complex projective plane

Sometimes, providing direct cohomology computations is rather difficult. One example of a space whose cohomology is difficult to compute directly, but is straightforward to compute using the Gysin sequence is $\mathbb{C}P^2$ – the cofibre of the Hopf map $h : \mathbb{S}^3 \to \mathbb{S}^2$ – the details of which will not be needed here (see (Ljungström and Mörtberg, 2023, Definition 5) for a self-contained definition or (HoTT Book, Section 8.5) for the original definition in HoTT). This was first done in HoTT by Brunerie (2016) who applied the Gysin sequence to the fibre sequence

$$\mathbb{S}^1 \to \mathbb{S}^5 \to \mathbb{C}P^2 \tag{7}$$

It follows from the Mayer–Vietoris sequence that, for any group G, the cohomology of $\mathbb{C}P^2$ is concentrated in degrees 0, 2 and 4:

$$H^n(\mathbb{C}P^2, G) \cong \begin{cases} G & \text{if } n \in \{0, 2, 4\} \\ \mathbb{1} & \text{otherwise} \end{cases}$$

The space $\mathbb{C}P^2$ is 1-connected and so we may apply the Gysin sequence to (7) with $n = 2$. Unfolding the sequence for $i = 2$ and $i = 4$ tells us that $e : H^2(\mathbb{C}P^2, \mathbb{Z})$ and $e^2 : H^4(\mathbb{C}P^2, \mathbb{Z})$ are generators, thus giving us $H^*(\mathbb{C}P^2, \mathbb{Z}) \cong \mathbb{Z}[x]/(x^3)$. This was formalised in full detail by Lamiaux et al. (2023).

5.5 Infinite real projective space

To further showcase the usefulness of the Gysin sequence, let us compute $H^*(\mathbb{R}P^\infty, \mathbb{Z}/2\mathbb{Z})$. This computation is particularly interesting since it is an example of an application of the Gysin sequence to a space which is *not* 1-connected. For it to work, we hence need to construct the family c_b in $(*)$ explicitly. As $\mathbb{R}P^\infty$ is well-known to be an Eilenberg–MacLane space we may conveniently define it by $\mathbb{R}P^\infty := K(\mathbb{Z}/2\mathbb{Z}, 1)$. We remark that this differs from the definition given in (Buchholtz and Rijke, 2017, Definition IV.1), but the argument we provide here should be easy to translate to fit their definition.

Hence, we are to compute $H^*(K(\mathbb{Z}/2\mathbb{Z}, 1), \mathbb{Z}/2\mathbb{Z})$. Using the equivalence $\psi : \Omega K(\mathbb{Z}/2\mathbb{Z}, 1) \simeq_\star \mathbb{S}^0$, we construct the fibre sequence

$$\mathbb{S}^0 \to \Sigma_{x:K(\mathbb{Z}/2\mathbb{Z},1)}(0_k = x) \to K(\mathbb{Z}/2\mathbb{Z}, 1)$$

This fits the Gysin sequence with $n = 1$ and the fibration $P : K(\mathbb{Z}/2\mathbb{Z}, 1) \to \mathrm{Type}$ being defined by $P(x) := (0_k = x)$. In order to apply the sequence, we need to construct, for each $x : K(\mathbb{Z}/2\mathbb{Z}, 1)$, a map

$$c_x : \Sigma(0_k = x) \to_\star K(\mathbb{Z}/2\mathbb{Z}, 1)$$

Let us initially construct c_x is the case when $x := 0_k$. In this case, the domain of c_x is simply $\Sigma(\Omega(K(\mathbb{Z}/2\mathbb{Z}, 1)))$ and hence the counit of the adjunction $\Sigma \dashv \Omega$ gives us our map:

$$\varepsilon : \Sigma(\Omega(K(\mathbb{Z}/2\mathbb{Z}, 1))) \to_\star K(\mathbb{Z}/2\mathbb{Z}, 1)$$

We would like to define $c_{0_k} := \varepsilon$, but it is not entirely obvious why this definition would define c_x for *all* $x : K(\mathbb{Z}/2\mathbb{Z}, 1)$ – the type $\Sigma(0_k = 0_k) \to_\star K(\mathbb{Z}/2\mathbb{Z}, 1)$ is not a proposition but a set, and hence this is not automatic. The trick is to add some structure to the type of c_x by instead considering the following type:

$$\sum_{e:\Sigma(0_k=x)\to_\star K(\mathbb{Z}/2\mathbb{Z},1)} (e \neq \mathsf{const}) \tag{8}$$

Lemma 70. *The type in (8) is contractible. Furthermore, ε is non-constant and thus is the centre of contraction when x is 0_k.*

Proof. Since we are proving a proposition, it suffices to do so when x is 0_k. We get

$$\begin{aligned}(\Sigma(0_k = 0_k) \to_\star K(\mathbb{Z}/2\mathbb{Z}, 1)) &\simeq (\Omega(K(\mathbb{Z}/2\mathbb{Z}, 1)) \to_\star \Omega(K(\mathbb{Z}/2\mathbb{Z}, 1))) \\ &\simeq (\mathbb{S}^0 \to_\star \mathbb{S}^0) \\ &\simeq \mathsf{Hom}(\mathbb{Z}/2\mathbb{Z}, \mathbb{Z}/2\mathbb{Z})\end{aligned}$$

By construction, this equivalence sends ε to the identity on $\mathbb{Z}/2\mathbb{Z}$. The predicate $(-) \neq \mathsf{const}$ is easily seen to be preserved by this chain of equivalences, and hence we get

$$\sum_{e:\Sigma(0_k=0_k)\to_\star K(\mathbb{Z}/2\mathbb{Z},1)} (e \neq \mathsf{const}) \simeq \sum_{\phi:\mathsf{Hom}(\mathbb{Z}/2\mathbb{Z},\mathbb{Z}/2\mathbb{Z})} (\phi \neq \mathsf{const})$$

Since the identity is the unique non-constant map in $\mathsf{Hom}(\mathbb{Z}/2\mathbb{Z}, \mathbb{Z}/2\mathbb{Z})$, we are done. □

Hence, by Lemma 70, we construct our family $c_x : \Sigma(0_k = x) \to_\star K(\mathbb{Z}/2\mathbb{Z}, 1)$ by choosing the unique element of (8). This gives us our class $e : H^1(K(\mathbb{Z}/2\mathbb{Z}, 1), \mathbb{Z}/2\mathbb{Z})$. In fact, with this rather explicit construction, it is easy to show that $e = |\,\mathsf{id}\,|$, but this additional knowledge will not be needed here. We may now instantiate the Gysin sequence. We get, for $i \geq 1$, exact sequences

$$H^{i-1}(E, \mathbb{Z}/2\mathbb{Z}) \longrightarrow H^{i-1}(\mathbb{R}P^\infty, \mathbb{Z}/2\mathbb{Z}) \xrightarrow{(-)\smile e} H^i(\mathbb{R}P^\infty, \mathbb{Z}/2\mathbb{Z}) \longrightarrow H^i(E, \mathbb{Z}/2\mathbb{Z})$$

where, recall, $\mathbb{R}P^\infty := K(\mathbb{Z}/2\mathbb{Z}, 1)$ and $E := \sum_{x:K(\mathbb{Z}/2\mathbb{Z},1)}(0_k = x)$. Since E is contractible, we get that the $(-) \smile e : H^{i-1}(\mathbb{R}P^\infty, \mathbb{Z}/2\mathbb{Z}) \to H^i(\mathbb{R}P^\infty, \mathbb{Z}/2\mathbb{Z})$ is an isomorphism. Since $\mathbb{R}P^\infty$ is connected, we have $H^0(\mathbb{R}P^\infty, \mathbb{Z}/2\mathbb{Z}) \cong \mathbb{Z}/2\mathbb{Z}$, and hence we get a chain of isomorphisms

$$\mathbb{Z}/2\mathbb{Z} \cong H^0(\mathbb{R}P^\infty, \mathbb{Z}/2\mathbb{Z}) \cong H^1(\mathbb{R}P^\infty, \mathbb{Z}/2\mathbb{Z}) \cong \cdots \cong H^n(\mathbb{R}P^\infty, \mathbb{Z}/2\mathbb{Z})$$

whose composition is simply the map $(-) \smile e^n : \mathbb{Z}/2\mathbb{Z} \to H^n(\mathbb{R}P^\infty, \mathbb{Z}/2\mathbb{Z})$. Hence, we have computed $H^*(\mathbb{R}P^\infty, \mathbb{Z}/2\mathbb{Z})$.

Proposition 71. $H^*(\mathbb{R}P^\infty, \mathbb{Z}/2\mathbb{Z}) \cong \mathbb{Z}/2\mathbb{Z}[x]$.

6. Computer computations and synthetic cohomology theory

In this section we will discuss various computations which can be performed using the constructions in the paper when formalised in a system like `Cubical Agda`, where univalence and HITs have computational content. This section is the only in the paper which cannot be directly formalised in Book HoTT where univalence and HITs are added axiomatically and lack computational content.

6.1 Proofs by computation in Cubical Agda

One big advantage of formalising synthetic mathematics in a system like `Cubical Agda` compared to Book HoTT is that many simple routine results hold definitionally. This sometimes vastly simplifies proofs. Even more ambitiously, one can hope that entire results can be proved directly by automatic computation. A classic conjecture of this form is the computability of the 'Brunerie number' (Brunerie, 2016) – that is, the number $n : \mathbb{Z}$ such that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/n\mathbb{Z}$. The number n can, in theory, be computed by running the function

$$\mathbb{1} \xrightarrow{(1,1)} \mathbb{Z} \times \mathbb{Z} \xrightarrow{\cong} \pi_2(\mathbb{S}^2) \times \pi_2(\mathbb{S}^2) \xrightarrow{[-,-]} \pi_3(\mathbb{S}^2) \xrightarrow{\cong} \mathbb{Z}$$

where $[-,-]$ denotes the Whitehead product. Brunerie dedicates the second half of the thesis to manually proving that the absolute value of the number returned by this function is 2. To this end, he has to introduce cohomology, the Hopf invariant and several other complex constructions. If this number could instead be computed on a computer, these manual computations would not be required. Unfortunately, we still have not been able to carry out this computation in `Cubical Agda`, or in any other proof assistant where univalence and HITs have computational content.

However, while the original Brunerie number is still out of reach, a much simplified version of the number was successfully computed by the authors in (Ljungström and Mörtberg, 2023). We refer the interested reader to that paper for details. Computations of related numbers have also been reported by Jack (2023). However, an even simpler example of the same type of problem can be found when trying to show that the wedge sum of two circles and a sphere is not homotopy equivalent to the torus. This is typically done by distinguishing their cohomology rings and the problem boils down to showing that the cup product vanishes in degrees $(1,1)$ for $\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1$ but not for $\mathbb{T}^2$. This was done computationally by Brunerie et al. (2022, Section 6) by running the following function in `Cubical Agda`

$$\mathbb{1} \xrightarrow{(1,0),(0,1)} (\mathbb{Z} \times \mathbb{Z}) \times (\mathbb{Z} \times \mathbb{Z}) \xrightarrow{\cong} H^1(X,\mathbb{Z}) \times H^1(X,\mathbb{Z}) \xrightarrow{\smile} H^2(X,\mathbb{Z}) \xrightarrow{\cong} \mathbb{Z}$$

for X the two spaces in question. Simply by noting that the output was 0 for one space and 1 for the other, one can deduce that $\mathbb{S}^2 \vee \mathbb{S}^1 \vee \mathbb{S}^1 \not\simeq \mathbb{T}^2$.

Another class of results which we have successfully proved this way is that of statements concerning the fact that some cohomology group is generated by some particular element $g : H^n(X,G)$. By first showing that $H^n(X,G)$ is isomorphic to a simple well-known group like $\mathbb{Z}$ by some isomorphism ϕ, it suffices to check that $\phi(g) = \pm 1$ for g to generate $H^n(X,G)$. However, the applicability of this method is also sometimes limited by computations being infeasible. Already in (Brunerie, 2016), another such number appears: namely, the output of the following function

$$\mathbb{1} \xrightarrow{(1,1)} \mathbb{Z} \times \mathbb{Z} \xrightarrow{\cong} H^2(\mathbb{C}P^2,\mathbb{Z}) \times H^2(\mathbb{C}P^2,\mathbb{Z}) \xrightarrow{\smile} H^4(\mathbb{C}P^2,\mathbb{Z}) \xrightarrow{\cong} \mathbb{Z} \tag{9}$$

Successfully running it should give an output of absolute value 1 and would provide a proof that the Hopf invariant $\pi_3(\mathbb{S}^2) \to \mathbb{Z}$ is an isomorphism. This computation was attempted in `Cubical Agda` by Brunerie et al. (2022) without luck. The definition of this number is arguably less complex than that of the Brunerie number, so the fact that `Cubical Agda` is stuck on it is an indication that a careful benchmarking of similar examples should be done. Further similar numbers approximating the original Brunerie number can be found in (Ljungström and Mörtberg, 2023) and we have, in fact, also in this paper implicitly defined other similar numbers. Consider, for instance the function

$$\mathbb{1} \xrightarrow{|\alpha|,|\alpha|} H^1(\mathbb{R}P^2,\mathbb{Z}/2\mathbb{Z}) \times H^1(\mathbb{R}P^2,\mathbb{Z}/2\mathbb{Z}) \xrightarrow{\smile} H^2(\mathbb{R}P^2,\mathbb{Z}/2\mathbb{Z}) \xrightarrow{\cong} \mathbb{Z}/2\mathbb{Z}$$

where $\alpha : \mathbb{R}P^2 \to K(\mathbb{Z}/2\mathbb{Z},1)$ is defined as in Proposition 66. Computing this, we should get 1 as output. This would completely remove the need for Lemma 65 in the proof of Proposition 66. Similar computations could also be used to verify Proposition 68. We judge these computations to

be more feasible than that of (9) since the degrees of the cohomology groups involved are lower, although they have thus far not been successfully run in `Cubical Agda`.

6.2 Benchmarks

In order to find potential bottlenecks and get a better idea of which computations succeed and which fail, we have run several benchmarks in `Cubical Agda`. These all consider the underlying maps $\phi : H^n(X, G) \to H$ maps of isomorphisms defined for concrete groups G and H, and spaces/-types X. Using these maps, we have tried computing $\phi(h)$ for concrete elements $h : H^n(X, G)$. As H is often a simpler group like $\mathbb{Z}$, $\mathbb{Z} \times \mathbb{Z}$ or $\mathbb{Z}/2\mathbb{Z}$, these values are all easily understood. We typically pick h as a generator of $H^n(X, G)$, or as some more complicated element constructed from generators. Whenever $H^n(X, G)$ is a cyclic group, we write $g^{(n)}$ for its generator and when there are multiple generators, we simply write $g_i^{(n)}$ for the ith generator. These generators are all easily constructed and we refer the interested reader to the `Agda` code for the precise definitions.

The various benchmarks and choices are all given in Table 1. The first four columns gives n, X, G, and H, s.t. $H^n(X, G) \cong H$. Both the G and H columns often have two rows to indicate the two choices of G which we have considered. We have focused on $\mathbb{Z}$ and $\mathbb{Z}/2\mathbb{Z}$ coefficients as these are the most interesting from a computational point of view. The fifth column contains the values of h which we have considered, separated by spaces. So, for $H^1(\mathbb{S}^1, G)$ for example, we considered 3 values of h for each of the two coefficient groups under consideration. The results can be found in the final column. Its entries are aligned with the corresponding values of h which we have tried.

As we expect similar goals like these to appear in future formalisations, the tests were run on a regular laptop with 1.60GHz Intel processor and 16GB of RAM. The successful computation were marked with ✓and the failed computations, marked with ✗, were manually terminated after a few minutes. Details and exact timings can be found at `www.github.com/aljungstrom/SyntheticCohomologyTests/blob/master/CohomologyComputations.agda`.

As expected, more tests work for lower dimensions, but for $\mathbb{R}P^2$, and the more complex K^2 and $\mathbb{R}P^\infty$, all tests fail even in dimension 1. This is not as surprising as it may seem. For $\mathbb{R}P^2$ and K^2, ϕ attempts to compute the winding number of a loop in $\Omega K(\mathbb{Z}, 1)$ which is constructed in terms of the complex proof that $\sigma^{-1} : \Sigma K(\mathbb{Z}, 2) \to K(\mathbb{Z}, 1)$ is a morphism of H-spaces. Another observation is that the choice of coefficients does not seem to make much of a difference. Somewhat more surprising, however, is the fact that several computations which terminated successfully in (Brunerie et al., 2022) fail to terminate here. It is not uncommon that generality comes at the cost of efficiency, but it is puzzling that a minor change of the definition of $K(\mathbb{Z}, n)$ (from the definition using n-truncated n-spheres to the one used here) would make such a difference.

7. More related and future work

In this paper, we have described a synthetic approach to cohomology theory in HoTT, with the possibility of doing direct computations in `Cubical Agda`. This works builds on lots of prior work which we have discussed throughout the paper, but there is also additional related work on which this work is not directly based. In this final section we will discuss this work, as well as possible future directions.

First of all, there is some related prior work already formalised in `Cubical Agda`. Qian (2019) formalised $K(G, 1)$ as a HIT, following Licata and Finster (2014), and proved that it satisfies $\pi_1(K(G, 1)) \cong G$. Alfieri (2019) and Harington (2020) formalised $K(G, 1)$ as the classifying space BG using G-torsors. Using this, $H^1(\mathbb{S}^1, \mathbb{Z}) \cong \mathbb{Z}$ was proved – however, computing using the maps in this definition proved to be infeasible.

Certified computations of homology groups using proof assistants have been considered prior to HoTT/UF. For instance, the `Coq` system (The `Coq` Development Team, 2021) has been used

n	X	G	H	h	Results
1	$\mathbb{S}^1$	$\mathbb{Z}$	$\mathbb{Z}$	$g^{(1)}$ $g^{(1)} +_h g^{(1)}$ $-_h g^{(1)}$	✓ ✓ ✓
		$\mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}$	$g^{(1)}$ $g^{(1)} +_h g^{(1)}$ $-_h g^{(1)}$	✓ ✓ ✓
2	$\mathbb{S}^2$	$\mathbb{Z}$	$\mathbb{Z}$	$g^{(2)}$ $g^{(2)} +_h g^{(2)}$ $-_h g^{(2)}$	✓ ✗ ✗
		$\mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}$	$g^{(2)}$ $g^{(2)} +_h g^{(2)}$ $-_h g^{(2)}$	✓ ✗ ✗
3	$\mathbb{S}^3$	$\mathbb{Z}$	$\mathbb{Z}$	$g^{(3)}$ $g^{(3)} +_h g^{(3)}$ $-_h g^{(3)}$	✗ ✗ ✗
		$\mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}$	$g^{(3)}$ $g^{(3)} +_h g^{(3)}$ $-_h g^{(3)}$	✗ ✗ ✗
1	$\mathbb{T}^2$	$\mathbb{Z}$	$\mathbb{Z}^2$	$g_1^{(1)} +_h g_2^{(1)}$ $g_1^{(1)} -_h g_2^{(1)}$	✓ ✓
		$\mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}^2$	$g_1^{(1)} +_h g_2^{(1)}$ $g_1^{(1)} -_h g_2^{(1)}$	✓ ✓
2	$\mathbb{T}^2$	$\mathbb{Z}$	$\mathbb{Z}$	$g^{(2)}$ $g_1^{(1)} \smile g_2^{(1)}$ $\left(g_1^{(1)} +_h g_1^{(1)}\right) \smile g_2^{(1)}$	✗ ✗ ✗
		$\mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}$	$g^{(2)}$ $g_1^{(1)} \smile g_2^{(1)}$ $\left(g_1^{(1)} +_h g_1^{(1)}\right) \smile g_2^{(1)}$	✗ ✗ ✗
1	$\bigvee_{i=2,1,1} \mathbb{S}^i$	$\mathbb{Z}$	$\mathbb{Z}^2$	$g_1^{(1)} +_h g_2^{(1)}$ $g_1^{(1)} -_h g_2^{(1)}$	✓ ✓
		$\mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}^2$	$g_1^{(1)} +_h g_2^{(1)}$ $g_1^{(1)} -_h g_2^{(1)}$	✓ ✓
2	$\bigvee_{i=2,1,1} \mathbb{S}^i$	$\mathbb{Z}$	$\mathbb{Z}$	$g^{(2)}$ $g_1^{(1)} \smile g_2^{(1)}$ $\left(g_1^{(1)} +_h g_1^{(1)}\right) \smile g_2^{(1)}$	✓ ✓ ✓
		$\mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}$	$g^{(2)}$ $g_1^{(1)} \smile g_2^{(1)}$ $\left(g_1^{(1)} +_h g_1^{(1)}\right) \smile g_2^{(1)}$	✓ ✓ ✓
1	$\mathbb{R}P^2$	$\mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}$	$g^{(1)}$ $g^{(1)} +_h g^{(1)}$ $-_h g^{(1)}$	✗ ✗ ✗
2	$\mathbb{R}P^2$	$\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}$	$g^{(2)}$ $g^{(2)} +_h g^{(2)}$ $-_h g^{(2)}$	✗ ✗ ✗
		$\mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}$	$g^{(2)}$ $g_1^{(1)} \smile g_1^{(1)}$	✗ ✗
1	K^2	$\mathbb{Z}$	$\mathbb{Z}$	$g^{(1)}$ $g^{(1)} +_h g^{(1)}$ $-_h g^{(1)}$	✗ ✗ ✗
		$\mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}^2$	$g_1^{(1)} +_h g_2^{(1)}$ $g_1^{(1)} -_h g_2^{(1)}$	✗ ✗
2	K^2	$\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}$	$g^{(2)}$ $g_1^{(1)} \smile g_1^{(1)}$	✗ ✗
		$\mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}$	$g^{(2)}$ $g_1^{(1)} \smile g_2^{(1)}$ $\left(g_1^{(1)} +_h g_1^{(1)}\right) \smile g_2^{(1)}$	✗ ✗ ✗
1	$\mathbb{R}P^\infty$	$\mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z}$	$g^{(1)}$ $g^{(1)} +_h g^{(1)}$ $-_h g^{(1)}$	✗ ✗ ✗

Table 1. : Benchmarks

to compute homology (Heras et al., 2012) and persistent homology (Heras et al., 2013) with coefficients in a field. This was later extended to homology with $\mathbb{Z}$-coefficients by Cano et al. (2016). The approach in these papers was entirely algebraic and spaces were represented as simplicial complexes. Other formalisations of various classical (co)homology theories can be found in `Lean`'s `mathlib` (mathlib Community, 2020), but this work is also not synthetic. However, an earlier HoTT formalisation of synthetic (co)homology theory in `Lean` 2 can be found in (The Spectral Sequence Project, 2018) which was developed as part of (van Doorn, 2018).

An interesting approach to increase the efficiency of synthetic cohomology computations would be to develop classical computational approaches to cohomology synthetically. This was done for cellular cohomology by Buchholtz and Hou (Favonia) (2018) who showed that a cohomology theory akin to the one considered in this paper can also be computed using synthetic cellular cohomology for spaces which are CW complexes. In ongoing work with Loïc Pujet, we are formalising cellular (co)homology in `Cubical Agda` with the aim of reducing (co)homology computations to linear algebra, similar to the certified computations of homology groups in `Coq`

mentioned above (which relied on certified Gaussian elimination and Smith normal form computations, as is customary in traditional computational topology). One envisioned application of this is in the formalisation of the recent synthetic proof of Serre's classification theorem for homotopy groups of spheres by Barton and Campion (Barton, 2022).

A synthetic approach to homology in HoTT/UF was developed informally by Graham (2018) using stable homotopy groups. This was later extended with a proof of the Hurewicz theorem by Christensen and Scoccola (2023). There has also been some recent work on synthetic definitions of other classical tools in homological algebra, in particular Ext groups (Christensen and Flaten, 2023). Various results related to synthetic homology theory, including a formalisation of the Eilenberg–Steenrod axioms for homology were formalised as part of (The Spectral Sequence Project, 2018). It would be interesting to also formalise this in `Cubical Agda` and try to compute also homology groups synthetically.

The definition of $\smile$ in the setting of $\mathbb{Z}$-cohomology in HoTT/UF is due to Brunerie (2016, Chapter 5.1). This definition, however, relies on the smash product which has proved very complex to reason about formally (Brunerie, 2018). Despite this, Baumann (2018) generalised this to $H^n(X, G)$ and managed to formalise graded commutativity in `HoTT-Agda`. Baumann's formal proof of this property is ~ 5000 LOC while our formalisation is just ~ 1200 LOC. This indicates that it would be infeasible to formalise other algebraic properties of $H^*(X, R)$ with this definition. Associativity seems particularly infeasible, but with our definition the formal proof is only ~ 600 LOC. In fact, recent work by Ljungström (2024) on the symmetric monoidal structure of the smash product fills in the gaps in (Brunerie, 2016), thus making Brunerie's arguments, in theory, formalisable. Nevertheless, the definition we have presented here still appears to us to be more convenient to work with.

References

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

Barton, R. 2022. Finite presentability of homotopy groups of spheres. Talk at the Seminar on Homotopy Type Theory at CMU, presenting joint work with Tim Campion.

Baumann, T. 2018. The cup product on cohomology groups in Homotopy Type Theory. Master's thesis, University of Augsburg.

Brown, E. H. 1962. Cohomology theories. *Annals of Mathematics*, 75(3):467–484.

Brunerie, G. 2016. *On the homotopy groups of spheres in homotopy type theory*. PhD thesis, Université Nice Sophia Antipolis.

Brunerie, G. 2018. Computer-generated proofs for the monoidal structure of the smash product. *Homotopy Type Theory Electronic Seminar Talks*.

Brunerie, G., Ljungström, A., and Mörtberg, A. 2022. Synthetic Integral Cohomology in Cubical Agda. In **Manea, F. and Simpson, A.**, editors, *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*, volume 216 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 11:1–11:19, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 10.4230/LIPIcs.CSL.2022.11.

Buchholtz, U., Christensen, J. D., Flaten, J. G. T., and Rijke, E. 2023. Central H-spaces and banded types. arXiv: 2301.02636.

Buchholtz, U. and Hou (Favonia), K.-B. 2018. Cellular Cohomology in Homotopy Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pp. 521–529, New York, NY, USA. Association for Computing Machinery. 10.1145/3209108.3209188.

Buchholtz, U. and Rijke, E. 2017. The Real Projective Spaces in Homotopy Type Theory. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '17. IEEE Press.

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

Cano, G., Cohen, C., Dénès, M., Mörtberg, A., and Siles, V. 2016. Formalized Linear Algebra over Elementary Divisor Rings in Coq. *Logical Methods in Computer Science*, 12(2). 10.2168/LMCS-12(2:7)2016.

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

Cavallo, E. 2022. Formalization of Evan's Trick. www.github.com/agda/cubical/blob/f8d6fcf3bc24ce3915b3960ad3b716fd0

82e8b9d/Cubical/Foundations/Pointed/Homogeneous.agda.

Christensen, J. D. and Flaten, J. G. T. 2023. Ext groups in Homotopy Type Theory. arXiv: 2305.09639.

Christensen, J. D. and Scoccola, L. 2023. The Hurewicz theorem in Homotopy Type Theory. *Algebraic & Geometric Topology*, 23:2107–2140.

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

Graham, R. 2018. Synthetic Homology in Homotopy Type Theory. arXiv: 1706.01540.

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

Hatcher, A. 2002. *Algebraic Topology*. Cambridge University Press.

Heras, J., **Coquand, T.**, **Mörtberg, A.**, **and Siles, V.** 2013. Computing Persistent Homology Within Coq/SSReflect. *ACM Transactions on Computational Logic*, 14(4):1–26. 10.1145/2528929.

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

Hou (Favonia), K.-B., **Finster, E.**, **Licata, D. R.**, **and Lumsdaine, P. L.** 2016. A Mechanization of the Blakers–Massey Connectivity Theorem in Homotopy Type Theory. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '16, pp. 565–574, New York, NY, USA. ACM. 10.1145/2933575.2934545.

Jack, T. 2023. $\pi_4(\mathbb{S}^3) \not\equiv 1$ and another Brunerie number in CCHM. The Second International Conference on Homotopy Type Theory (HoTT 2023).

Lamiaux, T., **Ljungström, A.**, **and Mörtberg, A.** 2023. Computing Cohomology Rings in Cubical Agda. In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2023, pp. 239–252, New York, NY, USA. Association for Computing Machinery. 10.1145/3573105.3575677.

Licata, D. R. and Brunerie, G. 2014. A Cubical Type Theory. Talk at Oxford Homotopy Type Theory Workshop.

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

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

Ljungström, A. 2024. Symmetric monoidal smash products in homotopy type theory. *Mathematical Structures in Computer Science*, 34(9):985–1007. 10.1017/S0960129524000318.

Ljungström, A. and Mörtberg, A. 2023. Formalizing $\pi_4(S^3) \cong \mathbb{Z}/2\mathbb{Z}$ and Computing a Brunerie Number in Cubical Agda. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. 10.1109/lics56636.2023.10175833.

mathlib Community, T. 2020. The Lean Mathematical Library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, pp. 367–381, New York, NY, USA. Association for Computing Machinery. 10.1145/3372885.3373824.

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

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

Shulman, M. 2013. Cohomology. Post on the Homotopy Type Theory blog: www.homotopytypetheory.org/2013/07/24/.

Shulman, M. 2019. All $(\infty, 1)$-toposes have strict univalent universes. arXiv: 1904.07004.

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

The Coq Development Team 2021. The Coq Proof Assistant. www.coq.inria.fr.

The Spectral Sequence Project 2018. www.github.com/cmu-phil/Spectral.

The Univalent Foundations Program 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Self-published, Institute for Advanced Study.

van Doorn, F. 2018. *On the Formalization of Higher Inductive Types and Synthetic Homotopy Theory*. PhD thesis, Carnegie Mellon University.

Vezzosi, A., **Mörtberg, A.**, **and Abel, A.** 2021. Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types. *Journal of Functional Programming*, 31:e8. 10.1017/S0956796821000034.

Wärn, D. 2023. Eilenberg–Maclane spaces and stabilisation in homotopy type theory. *Journal of Homotopy and Related Structures*, 18(2):357–368. 10.1007/s40062-023-00330-5.

Paper II

FORMALISING AND COMPUTING THE FOURTH HOMOTOPY GROUP OF THE 3-SPHERE IN CUBICAL AGDA

AXEL LJUNGSTRÖM [a] AND ANDERS MÖRTBERG [a]

Department of Mathematics, Stockholm University, Stockholm, Sweden
e-mail address: axel.ljungstrom@math.su.se, anders.mortberg@math.su.se

Abstract. Brunerie's 2016 PhD thesis contains the first synthetic proof in Homotopy Type Theory (HoTT) of the classical result that the fourth homotopy group of the 3-sphere is $\mathbb{Z}/2\mathbb{Z}$. The proof is one of the most impressive pieces of synthetic homotopy theory to date and uses a lot of advanced classical algebraic topology rephrased synthetically. Furthermore, the proof is fully constructive and the main result can be reduced to the question of whether a particular "Brunerie number" β can be normalised to ± 2. The question of whether Brunerie's proof could be formalised in a proof assistant, either by computing this number or by formalising the pen-and-paper proof, has since remained open. In this paper, we present a complete formalisation in `Cubical Agda`. We do this by modifying Brunerie's proof so that a key technical result, whose proof Brunerie only sketched in his thesis, can be avoided. We also present a formalisation of a new and much simpler proof that β is ± 2. This formalisation provides us with a sequence of simpler Brunerie numbers, one of which normalises very quickly to -2 in `Cubical Agda`, resulting in a fully formalised computer-assisted proof that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$.

1. Introduction

Homotopy theory originated in algebraic topology, but is by now a central tool in many branches of modern mathematics, such as algebraic geometry and category theory. One of the central notions of study in homotopy theory is that of the *homotopy groups* of a space X, denoted $\pi_n(X)$. These groups constitute a topological invariant, making them a powerful tool for establishing whether two given spaces can or cannot be homotopy equivalent. The first two such groups of a space are easy to understand: $\pi_0(X)$ characterises the connected components of X and $\pi_1(X)$ is the fundamental group, i.e. the group of equivalence classes consisting of the loops contained in X up to homotopy. This idea generalises to higher values

Key words and phrases: Homotopy type theory, Synthetic homotopy theory, Formalisation of mathematics, Constructive mathematics.

This paper is an extended version of "Formalizing $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$ and Computing a Brunerie Number in Cubical Agda" published in the post-proceedings of Logic in Computer Science 2023 [LM23]. Some details about what has been added can be found in the **Outline** paragraph below.

This paper is based upon research supported by the Swedish Research Council (SRC, Vetenskapsrådet) under Grant No. 2019-04545. The research has also received funding from the Knut and Alice Wallenberg Foundation through the Foundation's program for mathematics.

Preprint submitted to
Logical Methods in Computer Science

of n, for which $\pi_n(X)$ consists of n-dimensional loops up to homotopy. For many spaces, these groups tend to become increasingly esoteric and difficult to compute for large n. This is true also for seemingly tame spaces like spheres, for which $\pi_n(\mathbb{S}^m)$ in general is highly irregular when $n > m \geq 2$.[1] This paper concerns the first computer formalisation of the classical result that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$, a result which is particularly interesting because it gives the whole first stable stem of homotopy groups of spheres, i.e. $\pi_{n+1}(\mathbb{S}^n)$ for $n \geq 3$. The fact that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$ was proved already in the 1930's by Pontryagin using cobordism theory, but we instead follow the synthetic approach to homotopy theory developed in Homotopy Type Theory (HoTT) and popularised by the HoTT Book [Uni13]. In this new approach to homotopy theory, spaces are represented directly as (higher inductive) types and homotopy groups are computed using Voevodsky's univalence axiom [Voe10a]. This gives a logical approach to homotopy theory, suitable for computer formalisation in proof assistants based on type theory, while also making it possible to interpret results in any suitably structured $(\infty, 1)$-topos [Shu19].

The basis for our formalisation is the 2016 PhD thesis of Brunerie [Bru16a] which contains the first synthetic proof in HoTT that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$. The proof is one of the most impressive pieces of synthetic homotopy theory to date and uses advanced machinery from classical algebraic topology developed synthetically, including the symmetric monoidal structure of smash products, (integral) cohomology rings, the Mayer-Vietoris and Gysin sequences, the Hopf invariant, Whitehead products, etc. The formalisation of Brunerie's proof has since remained open, primarily due to the highly technical nature of some of the details. In this paper, we will present such a formalisation in `Cubical Agda` [VMA21], a *cubical* extension of the `Agda` proof assistant [Agd24] with native support for computational univalence and higher inductive types (HITs).

In addition to being a very impressive proof in synthetic homotopy theory, Brunerie's proof is particularly interesting as it is fully constructive. The proof consists of two parts, with the first one culminating in Chapter 3 with the definition of a number $\beta : \mathbb{Z}$ such that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/\beta\mathbb{Z}$. Since then, this β has been commonly referred to as the *Brunerie number*. Brunerie writes the following about it:

> *This result is quite remarkable in that even though it is a constructive proof, it is not at all obvious how to actually compute this [β]. At the time of writing, we still haven't managed to extract its value from its definition.* [Bru16a, Page 85]

In fact, [Bru16a, Appendix B] contains a complete and concise definition of β as the image of 1 under a sequence of 12 maps:

$$\begin{array}{c}
\mathbb{Z} \xrightarrow{n\mapsto \mathsf{loop}^n} \Omega(\mathbb{S}^1) \xrightarrow{\Omega\varphi_{\mathbb{S}^1}} \Omega^2(\mathbb{S}^2) \xrightarrow{\Omega^2\varphi_{\mathbb{S}^2}} \Omega^3(\mathbb{S}^3) \\
\xleftarrow{\Omega^3 e} \\
\Omega^3(\mathbb{S}^1 * \mathbb{S}^1) \xrightarrow{\Omega^3\alpha} \Omega^3(\mathbb{S}^2) \xrightarrow{h} \Omega^3(\mathbb{S}^1 * \mathbb{S}^1) \xrightarrow{\Omega^3(e^{-1})} \Omega^3(\mathbb{S}^3) \\
\xleftarrow{e_3} \\
\Omega^2\|\mathbb{S}^2\|_2 \xrightarrow{\Omega\kappa_{2,\mathbb{S}^2}} \Omega\|\Omega(\mathbb{S}^2)\|_1 \xrightarrow{\kappa_{1,\Omega\mathbb{S}^2}} \|\Omega^2(\mathbb{S}^2)\|_0 \xrightarrow{e_2} \Omega(\mathbb{S}^1) \xrightarrow{e_1} \mathbb{Z}
\end{array}$$

[1] See [Bru16a, Figure 2.1] for a table of $\pi_n(\mathbb{S}^m)$ for small n and m.

By implementing this number in a proof assistant with computational support for univalence and HITs, one should be able to normalise it using a computer to establish that $\beta = \pm 2$ and hence that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$. In 2016, by the time Brunerie was finishing his thesis, there were some experimental proof assistants based on the cubical type theory of [CCHM18], but these were too slow to perform such a complex computation. So, instead of relying on normalisation, Brunerie spends the second part of the thesis (Chapters 4–6) to prove, using a lot of the advanced machinery mentioned above, that $|\beta|$ is propositionally equal to 2. However, if one were instead able to compute the number automatically in a proof assistant, this equality would hold definitionally—effectively reducing the complexity and length of the proof by an order of magnitude.

The intriguing possibility of a computer assisted formal proof made many people interested and countless attempts to normalise Brunerie's β have been made using increasingly powerful computers. However, to date, no one has succeeded and it is still unclear whether it is normalisable in a reasonable amount of time. In light of this, it is natural to wonder whether it is possible to simplify Brunerie's number in order to be able to compute it. For example, Brunerie's original definition only involves 1-HITs, as the status of higher HITs was still quite understudied at the time. With a better understanding of higher HITs developed in [LS20, CHM18, CH19], one quickly sees that the first 3 maps can be combined into one sending 1 to the 3-cell of $\mathbb{S}^3$ defined as a 3-HIT and not as an iterated suspension as in Brunerie's thesis. Unfortunately, simple optimisations like this do not seem to reduce the complexity of the computation enough and all attempts to run it have thus far failed.

After several unsuccessful attempts at optimising the computation, we instead decided to formalise the second half of Brunerie's thesis. However, this is by no means straightforward. The first issue appears already in the beginning of Chapter 4, a chapter concerning smash products of spheres. The main result of the section is Proposition 4.1.2, which says that the smash product is a 1-coherent symmetric monoidal product on pointed types. However, the proof of this result is just a sketch and Brunerie writes the following about it:

> *The following result is the main result of this section even though we essentially admit it.* [Bru16a, Page 90]

Unfortunately, this result is then used to construct integral cohomology rings, $H^*(X)$, whose cup product, $\smile$, appears in the definition of the so-called Hopf invariant which is crucially used to prove that $|\beta|$ is 2. While one might be convinced that Brunerie's informal proof sketch is correct, it is not obvious how one convinces a proof assistant of this. A complete formalisation would either have to fill in the holes in the sketch or find an alternative construction which avoids Proposition 4.1.2. In fact, Brunerie tried very hard to fill these holes using `Agda` metaprogramming [Bru18]. However, he never managed to typecheck his computer generated proof of the pentagon identity. Hence, this approach also seems infeasible with current proof assistant technology.

Luckily, Brunerie, Ljungström and Mörtberg [BLM22] recently gave an alternative synthetic definition of the cup product on $H^*(X)$ which completely avoids smash products. This has allowed us to completely skip the problematic Chapter 4 and, in particular, Proposition 4.1.2, while still following the proofs in Chapters 5 and 6. Having a strategy for a formal proof, we were then able to embark on able to embark on the ambitious project of formalising Brunerie's proof. Even though we do not need any theory about smash products, there was still a lot left to formalise and our final formalisation closely follows Brunerie's

proof, except for various smaller simplifications and adjustments which we discuss in the paper.

In addition to this, we have also formalised a new proof by Ljungström [Lju22] which completely circumvents Chapters 4–6. This major simplification builds on manually calculating the image of the element $\eta : \pi_3(\mathbb{S}^2)$, corresponding to β under the isomorphism $\pi_3(\mathbb{S}^2) \cong \mathbb{Z}$, by dividing this isomorphism into several maps, tracing η in each step. In particular, the new proof is completely elementary and does not rely on advanced tools such as cohomology. The elements that one obtains while tracing η are all new "Brunerie numbers" that should normalise to ± 2. In fact, one of these normalises, in just under 4 seconds on a regular laptop, to -2 in `Cubical Agda` at the time of writing. So, despite still not being able to compute the original β, this work can be seen as an alternative solution to Brunerie's conjecture about obtaining a computational proof that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$ which relies on simplifying the Brunerie number until it becomes effectively computable.

Outline. The paper closely follows the structure of Brunerie's proof. In section 2, we discuss key results from HoTT that we will need and their formalisation in `Cubical Agda`. section 3, which roughly corresponds to Chapter 2 of Brunerie's thesis, contains some first results on homotopy groups of spheres—e.g. the computation of $\pi_n(\mathbb{S}^m)$ for $n \leq m$. We then give Brunerie's definition of β and prove that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/\beta\mathbb{Z}$, the formalisation of which involves the James construction and Whitehead products. The remainder of the paper is then devoted to the formalisation of the different proofs that $\beta = \pm 2$. We first discuss the formalisation of Chapters 4–6 of Brunerie's proof in section 5. This involves a lot of technical machinery like cohomology, the Hopf invariant, etc. We then, in section 6, turn our attention to the new elementary proof that $\beta = \pm 2$ and the new Brunerie number which quickly normalises to -2 in `Cubical Agda`. Here, we also present some result concerning joins of spheres and the vanishing of Whitehead products. We conclude in section 7 with a discussion and comparison of the different formal proofs, as well as some directions for future work.

Compared to the previous publication on which the current paper is based, [LM23], the main differences are the following.

- Many proofs which were omitted because of page constraints in [LM23] have been added or extended throughout the paper. In particular, the proofs in [LM23, Section VI] have been substantially expanded with many details added in subsection 6.3.
- In section 6, many results from [LM23, Section VI] have also been generalised, e.g. the alternative definition of homotopy groups in terms of joins of spheres, π_n^*, is now studied in general and not just for $n = 3$.
- As part of the expansion and generalisation of [LM23, Section VI] in section 6, a new subsection 6.1 on joins and smash products of spheres, a new subsection 6.2 on homotopy groups in terms of joins and a new subsection 6.4 on the possibility of a stand-alone proof of Brunerie's theorem have been added.

Formalisation. All results in the paper have been formalised in `Cubical Agda` and are part of the `agda/cubical` library, available at `https://github.com/agda/cubical/`. The code in the paper is mainly literal `Agda` code taken verbatim from the library, but we have taken some liberties when typesetting, e.g. shortening notations and omitting some universe levels. A `Cubical Agda` summary file linking the formalisation and paper can be found at: `https://github.com/agda/cubical/blob/master/Cubical/Papers/Pi4S3-JournalVersion.agda`

The development typechecks with Agda's `--safe` flag, which ensures that there are no admitted goals or postulates.

2. Homotopy Type Theory in Cubical Agda

In this section, we concisely summarise the key HoTT concepts needed for the proofs and their formalisation in `Cubical Agda`. This roughly corresponds to [Bru16a, Chapter 1]. For a more in-depth introduction, see the HoTT Book [Uni13] which also serves as a reference for the formal language "Book HoTT". In this paper, we will present many things with cubical notations, but almost all of the results also hold with minor changes in Book HoTT where paths are represented using Martin-Löf's inductive Id-types [ML75] instead of cubical path types. In section 7 we discuss in more detail which proofs crucially rely on cubical features.

All of the results presented in this section were already part of the `agda/cubical` library before we began our formalisation and, while useful as a resource for our notations, experts on HoTT and `Cubical Agda` can safely skim this section.

2.1. Elementary HoTT notions and `Cubical Agda` notations.

We write $(x : A) \to B\,x$ for dependent function types and denote the identity function by $\mathsf{id}_A : A \to A$. We write $\Sigma_{x:A}(B\,x)$ for the dependent pair type and fst and snd for its projection maps. In what follows, we mean by a *pointed type* a dependent pair $(A, \star_A)$ consisting of a type A and a fixed basepoint $\star_A : A$. For ease of notation, we will often omit the basepoint and simply write A for the pointed type $(A, \star_A)$. Given two pointed types A and B, the type of *pointed functions* $A \to_\star B$ consists of pairs $(f, \star_f)$ where $f : A \to B$ and $\star_f : f\,\star_A \equiv \star_B$ witnesses basepoint preservation. Again, we simply write $f : A \to_\star B$ and take $\star_f$ implicit.

HoTT supports inductive types, i.e. types inductively generated by their constructors/points. We write Bool for the type of booleans and $\mathbb{1}$ for the unit/singleton type with a single point $\star_1$. A defining feature of HoTT, as opposed to plain Martin-Löf type theory [ML84], is the existence of *higher inductive types* (HITs). This is a generalisation of inductive types where we are not only allowed to specify the generating points of the type in question, but also identifications between these points (and possibly identifications of these identifications, and so on). This is useful for defining quotient types, but also for defining spaces when working in the *types-as-spaces* interpretation of HoTT (see e.g. [Uni13, Table 1] and [AW09]). `Cubical Agda` natively supports HITs and a type representing the circle can be defined as follows:

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

Here, base ≡ base denotes the type of identifications of base with itself. This is interpreted as the type of *paths* from base to itself when viewing $\mathbb{S}^1$ as a space. Hence, the above HIT captures precisely the representation of the circle as a cell complex with one 0-cell (base) and one 1-cell (loop). We always take $\mathbb{S}^1$ to be pointed by base. In order to discuss the induction principle for $\mathbb{S}^1$, we need to discuss paths in more detail. Cubically, paths correspond to functions out of the unit interval, just like in traditional topology. In `Cubical Agda`, there

is a primitive interval type[2] I with endpoints $\mathsf{i0}$ and $\mathsf{i1}$. A path of type $x \equiv y$ between two points $x, y : A$ is a function $p : \mathsf{I} \to A$ such that $p\,\mathsf{i0} = x$ and $p\,\mathsf{i1} = y$ judgmentally. For instance, refl, the constant path at a point x, is defined by:

$$\mathsf{refl} : (x : A) \to x \equiv x$$
$$\mathsf{refl}\ x = \lambda\ i \to x$$

Note that we use "=" for definitional/judgmental equality and "$\equiv$" for `Cubical Agda`'s path-equality. This can be contrasted with the HoTT Book [Uni13] which uses the opposite convention where "=" is propositional/typal equality and "$\equiv$" definitional/judgmental equality.

This type of notational conventions is not the only difference between `Cubical Agda` and Book HoTT. Many proofs that are complicated in Book HoTT become remarkably direct using the direct treatment of equality using path types. For instance, function extensionality and its inverse $\mathsf{funExt^{-}}$ are one-liners that just flip the arguments:

$$\mathsf{funExt} : ((x : A) \to f\ x \equiv g\ x) \to f \equiv g$$
$$\mathsf{funExt}\ p\ i\ x = p\ x\ i$$

$$\mathsf{funExt^{-}} : f \equiv g \to ((x : A) \to f\ x \equiv g\ x)$$
$$\mathsf{funExt^{-}}\ p\ x\ i = p\ i\ x$$

In Book HoTT, however, funExt is typically proved as a consequence of the univalence axiom using a rather ingenious proof [Lic14] while its inverse follows from path induction. Another elementary example of a proof involving $\_\equiv\_$ is cong (called ap in Book HoTT), which applies a function to a path:

$$\mathsf{cong} : (f : A \to B)\ (p : x \equiv y) \to f\ x \equiv f\ y$$
$$\mathsf{cong}\ f\ p\ i = f\ (p\ i)$$

Although the treatment of paths in `Cubical Agda` differs somewhat from Book HoTT, we may still prove *path induction*: for any dependent type $B : (y : A)\,(p : x \equiv y) \to \mathsf{Type}$, all dependent functions $f : (y : A)\,(p : x \equiv y) \to B\,x\,p$ are uniquely determined by $f\,x\,(\mathsf{refl}\,x)$. In Book HoTT, this can be used, among other things, to define the notion of a *dependent* path, which formalises the situation when two points $a : A$ and $b : B$ are equal up to a path $p : A \equiv B$. In `Cubical Agda`, however, the type of dependent paths is primitive:

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

In fact, $\_\equiv\_$ is just the special case of PathP where the line of paths is constant:

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

We are now ready to describe the induction principle of $\mathbb{S}^1$. A dependent function $f : (x : \mathbb{S}^1) \to B\,x$ is determined by a point $b : B\,\mathsf{base}$ and a loop $\ell : \mathsf{PathP}(\lambda i \to B(\mathsf{loop}\,i))\,b\,b$. In `Cubical Agda`, this would be written using pattern matching, as in the left-most definition below, which is introduced side-by-side with the way it would commonly be written in informal HoTT (as in Brunerie's thesis):

[2]For technical reasons, this is actually just a "pre-type" in `Cubical Agda`.

```
f base = b
f (loop i) = ℓ i
```

$$f(\mathsf{base}) = b$$
$$\mathsf{ap}_f(\mathsf{loop}) = \ell$$

2.2. **More higher inductive types.** Let us now introduce the remaining HITs used in [Bru16a]. These come equipped with induction principles analogous to that of $\mathbb{S}^1$. To define higher spheres, we need suspensions:

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

We always take suspensions to be pointed by north. We may now define the n-sphere, for $n \geq 1$, by $\mathbb{S}^n = \mathsf{Susp}^{n-1}\mathbb{S}^1$ where Susp^{n-1} denotes $(n-1)$-fold suspension. We also define $\mathbb{S}^{-1} = \bot$ (the empty type) and $\mathbb{S}^0 = \mathsf{Bool}$. We remark that we could equivalently have defined $\mathbb{S}^1$ as the suspension of $\mathbb{S}^0$ as is done in [Bru16a]. Our reason for not doing so is that certain functions using $\mathbb{S}^1$ appear to compute better with the base/loop definition. Furthermore, this is the definition used in already existing code in the `agda/cubical` library.

We may also capture the (homotopy) pushout of a span $B \xleftarrow{f} A \xrightarrow{g} C$ by 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)
```

Diagrammatically this corresponds to:

$$\begin{array}{ccc} A & \xrightarrow{g} & C \\ {\scriptstyle f}\downarrow & \ulcorner & \downarrow{\scriptstyle \mathsf{inr}} \\ B & \xrightarrow[\mathsf{inl}]{} & \mathsf{Pushout}\ f\ g \end{array}$$

We use pushouts to define the wedge sum of two pointed types, denoted $A \vee B$, the join of two types, denoted $A * B$, and the cofibre of a map $f : A \to B$, denoted cofib f:

$$\begin{array}{ccc} \mathbb{1} & \longrightarrow & B \\ \downarrow & \ulcorner & \downarrow \\ A & \longrightarrow & A \vee B \end{array} \qquad \begin{array}{ccc} A \times B & \xrightarrow{\mathsf{snd}} & B \\ {\scriptstyle \mathsf{fst}}\downarrow & \ulcorner & \downarrow \\ A & \longrightarrow & A * B \end{array} \qquad \begin{array}{ccc} A & \xrightarrow{f} & B \\ \downarrow & \ulcorner & \downarrow \\ \mathbb{1} & \longrightarrow & \mathsf{cofib}\ f \end{array}$$

Two particularly important functions out of wedge sums are

```
∇ : A ∨ A → A
∇ (inl x) = x
∇ (inr x) = x
∇ (push ⋆₁ i) = ⋆_A
```

and

```
i∨ : A ∨ B → A × B
i∨ (inl a) = (a , ⋆_B)
```

$i^\vee\ (\mathsf{inr}\ b) = (\star_A\ ,\ b)$
$i^\vee\ (\mathsf{push}\ \star_1\ i) = (\star_A\ ,\ \star_B)$

2.3. **Truncation levels and n-truncations.** An important concept in HoTT is that of Voevodsky's *h-levels* [Voe10b], which gives rise to the notion of an *n-type*. Since types in HoTT are interpreted as spaces (or rather, as homotopy types), they are not only determined by their points but also by which higher paths they may contain. We say that a type A is an n-type if all $(n+1)$-dimensional structure of A is trivial. Formally, this is captured by an inductive definition. We say that A is a (-2)-type if it is contractible, i.e. consisting of a single point, as captured by $\mathsf{isContr}\ A = \Sigma_{a_0:A}((a : A) \to a_0 \equiv a)$. We inductively say that A is an $(n+1)$-type if for any $x, y : A$, the type $x \equiv y$ is an n-type. We call (-1)-types *propositions* and 0-types *sets*.

We can turn any type A into an n-type by *n-truncation*, denoted $\| A \|_n$. For instance, the (-1)-truncation may be directly defined using the following HIT:

```
data ‖_‖₋₁ (A : Type) : Type where
  |_| : A → ‖ A ‖₋₁
  squash : (x y : ‖ A ‖₋₁) → x ≡ y
```

We often use direct definitions like this of (-1)- and 0-truncation in our formalisation, and similar constructions work for any fixed value of n, but not when n is arbitrary. For higher n we rely on the hub-and-spoke construction [Uni13, Section 7.3].

```
data ‖_‖ (A : Type) (n : ℕ₋₁) : Type where
  |_| : A → ‖ A ‖ n
  hub : (f : 𝕊 n → ‖ A ‖ n) → ‖ A ‖ n
  spoke : (f : 𝕊 n → ‖ A ‖ n) (x : 𝕊 n) → hub f ≡ f x
```

One caveat with truncations is that a map $f : A \to B$ does *not*, in general, induce a map $f : \| A \|_n \to B$. This is, however, the case when B is an n-type. In particular, f always induces a function $\| f \|_n : \| A \|_n \to \| B \|_n$.

2.4. **Univalence, loop spaces, and H-spaces.** In order to introduce Voevodsky's univalence principle [Voe10a], we need to define the (homotopy) fibre of a function. Given a function $f : A \to B$ and a point $b : B$, we define the fibre of f over b by $\mathsf{fib}\ f\ b = \Sigma_{x:A}(f\ a \equiv b)$. We say that $f : A \to B$ is an equivalence, written $f : A \simeq B$, if $\mathsf{fib}\ f\ b$ is contractible for all $b : B$. In order to prove that a function $f : A \to B$ is an equivalence, it suffices to provide an inverse $f^- : B \to A$ and two paths $f \circ f^- \equiv \mathsf{id}_B$ and $f^- \circ f \equiv \mathsf{id}_A$. If f is also pointed, we write $f : A \simeq_\star B$.

Univalence states that the canonical map $A \equiv B \to A \simeq B$, defined by path induction, is an equivalence. In particular, we get a map $\mathsf{ua} : A \simeq B \to A \equiv B$ promoting equivalences to paths. This provides us with a useful method for transferring proofs between equivalent types which extends to *structured* types and is then referred to as the *structure identity principle* [Uni13, Section 9.8].

Transferring proofs is, however, not the only use case of univalence in HoTT. It can also be used to characterise *loop spaces* of HITs. This is often done using the *encode-decode method* [Uni13, Section 8.1.4], a type theoretic analogue of proofs by contractibilty of total spaces of fibrations. In HoTT, we define the loop space of a pointed type A, by

$\Omega\, A = (\star_A \equiv \star_A)$. This is again pointed by $\mathsf{refl}\, \star_A$, so we may iterate this definition to get the nth loop space of A, denoted $\Omega^n\, A$. Loop spaces belong to a particularly important class of types called *H-spaces*. These consist of a pointed type B equipped with a unital magma structure

$$\begin{aligned} \mu &: B \times B \to B \\ \mu_l &: (b : B) \to \mu(\star_B, b) \equiv b \\ \mu_r &: (b : B) \to \mu(b, \star_B) \equiv b \end{aligned}$$

satisfying $\mu_l\, \star_B \equiv \mu_r\, \star_B$. Another particularly important H-space for our purposes is $\mathbb{S}^1$, for which we will use $+$ to denote its binary operation. $\mathbb{S}^1$ also comes equipped with a notion of inversion which we will denote by $-$. In fact, $\mathbb{S}^1$ is a commutative and associative H-space.

3. First results on homotopy groups of spheres

In this section, we cover [Bru16a, Chapter 2], which introduces some elementary results on the homotopy groups of spheres. All of these results can also be found in the HoTT Book [Uni13]. Before even stating them, we need homotopy groups:

Definition 3.1 (Homotopy groups)**.** For $n : \mathbb{N}$, we define the nth *homotopy group* of a pointed type A by:

$$\pi_n(A) = \| \mathbb{S}^n \to_\star A \|_0$$

The name homotopy *group* should be taken with a grain of salt: it, in general, only has a group structure when $n \geq 1$ (abelian when $n \geq 2$). The structure may be defined, much like in [BHF18, Section 5], by considering the equivalence $(\mathbb{S}^n \to_\star A) \simeq (\mathbb{S}^{n-1} \to_\star \Omega\, A)$, where the latter type has a multiplication given by pointwise path composition. An alternative definition of $\pi_n(A)$ is via loop spaces. There is an equivalence $\omega_n : \Omega^n\, A \simeq (\mathbb{S}^n \to_\star A)$ and, hence, we could equivalently have defined $\pi_n(A)$ by setting $\pi_n(A) = \| \Omega^n\, A \|_0$. This makes the group structure on $\pi_n(A)$ more transparent: it is simply path composition. This is the definition used in the HoTT Book [Uni13]. Brunerie uses both definitions in his thesis and often passes between the two without comment.

An elementary but crucial result for the computation of homotopy groups is the existence of the *long exact sequence of homotopy groups*. Its proof is usually phrased using the loop space definition of homotopy groups as in e.g. [Uni13, Theorem 8.4.6]. For ease of notation, let us write $\mathsf{fib}\, f$ for the fibre of a pointed function $f : A \to_\star B$ over the basepoint of B.

Proposition 3.2 (LES of homotopy groups)**.** *For any pointed map $f : A \to_\star B$, there is a long exact sequence*

$$\begin{array}{ccccccc} & & \cdots & \longrightarrow & \pi_{n+1}(B) & & \\ & \swarrow & & & & & \\ \pi_n(\mathsf{fib}\, f) & \longrightarrow & \pi_n(A) & \longrightarrow & \pi_n(B) & & \\ & \swarrow & & & & & \\ \pi_{n-1}(\mathsf{fib}\, f) & \longrightarrow & \cdots & & & & \end{array}$$

where the horizontal maps are induced by the functorial action of π_n on $\mathsf{fst} : \mathsf{fib}\, f \to A$ and $f : A \to B$.

Above, we have implicitly taken the kernel and image of a group homomorphism $\phi : G \to H$ to be defined by

$$\begin{aligned} \mathsf{ker}\,\phi &= \mathsf{fib}\,\phi\,0_H \\ \mathsf{im}\,\phi &= \Sigma_{h:H} \| \Sigma_{g:G}(\phi(g) \equiv h) \|_{-1} \end{aligned}$$

When analysing loop spaces and homotopy groups of suspensions, the following function is of great importance. It will be used in many constructions to come.

Definition 3.3 (The suspension map)**.** Given a pointed type A, there is a canonical map $\sigma : A \to \Omega\,(\mathsf{Susp}\,A)$ given by

$$\sigma\,x = \mathsf{merid}\,x \cdot (\mathsf{merid}\,\star_A)^{-1}$$

This induces a homomorphism on homotopy groups by post-composition:

$$\pi_n(A) \xrightarrow{\sigma_*} \pi_n(\Omega\,(\mathsf{Susp}\,A)) \xrightarrow{\cong} \pi_{n+1}(\mathsf{Susp}\,A)$$

We will often, with some abuse of notation, simply write σ_* for this composition. We also define $\sigma_n : \| A \|_n \to \Omega \| \mathsf{Susp}\,A \|_{n+1}$ by

$$\sigma_n |\,x\,| = \mathsf{cong}\,|\_|\,(\sigma\,x)$$

We will soon see the suspension map in action, but first we need the following elementary result.

Proposition 3.4 (Join of spheres)**.** $\mathbb{S}^n * \mathbb{S}^m \simeq \mathbb{S}^{n+m+1}$.

In fact, as we will see in section 6, there is more to say about this equivalence. We make a forward reference to Proposition 6.4 and the preceding discussion for a detailed account of its construction.

In particular, Proposition 3.4 gives us an equivalence $\mathbb{S}^1 * \mathbb{S}^1 \simeq \mathbb{S}^3$. Using this fact, we define the following map, which will play a crucial role in the analysis of $\pi_4(\mathbb{S}^3)$.

Definition 3.5 (Hopf map)**.** We define $\mathsf{hopf} : \mathbb{S}^3 \to \mathbb{S}^2$ by the composition $\mathbb{S}^3 \xrightarrow{\sim} \mathbb{S}^1 * \mathbb{S}^1 \xrightarrow{\mathsf{h}} \mathbb{S}^2$ where h is given by

$$\begin{aligned} &\mathsf{h} : \mathbb{S}^1 * \mathbb{S}^1 \to \mathbb{S}^2 \\ &\mathsf{h}\,(\mathsf{inl}\,x) = \mathsf{north} \\ &\mathsf{h}\,(\mathsf{inr}\,y) = \mathsf{north} \\ &\mathsf{h}\,(\mathsf{push}\,(x\ ,\ y)\ i) = \sigma\,(y\ \text{-}\ x)\ i \end{aligned}$$

where $-$ is defined using the H-space and inversion structure on $\mathbb{S}^1$.

It turns out that the following is true [Uni13, Theorem 8.5.1].

Proposition 3.6 (The fibre of the Hopf map)**.** *The fibre of* hopf *is equivalent to* $\mathbb{S}^1$*, i.e.* $\mathsf{fib}\,\mathsf{hopf} \simeq \mathbb{S}^1$.

Proposition 3.6 gives us a fibration sequence $\mathbb{S}^1 \to \mathbb{S}^3 \to \mathbb{S}^2$ which, in particular, will allow us to connect homotopy groups of $\mathbb{S}^2$ with those of $\mathbb{S}^3$ and $\mathbb{S}^1$. For this, we need to introduce the notion of *connectedness*. We say that a type A is n-connected if $\| A \|_n$ is contractible. Similarly, we say that a function $f : A \to B$ is n-connected if all of its fibres are n-connected. This means, in particular, that the induced function $\| f \|_n : \| A \|_n \to \| B \|_n$ is an equivalence. The following is an immediate consequence of the definition of n-truncations.

Lemma 3.7 (Connectedness of spheres)**.** *For $n \geq -1$, $\mathbb{S}^n$ is $(n-1)$-connected.*

Using Lemma 3.7, we can easily prove the following:

Proposition 3.8 ([Bru16a, Proposition 2.4.1])**.** *For $n < m$, the group $\pi_n(\mathbb{S}^m)$ is trivial.*

For the sake of completeness, let us take the liberty of mentioning some results from [Bru16a, Chapter 3] already here, since they also concern low-dimensional homotopy groups of spheres. A crucial result is the following theorem [Uni13, Theorem 8.6.4]:

Theorem 3.9 (Freudenthal suspension theorem)**.** *Given an n-connected and pointed type A, the map $\sigma : A \to \Omega\,(\mathsf{Susp}\,A)$ is $2n$-connected.*

On can easily deduce from Theorem 3.9 that, in particular, $\sigma_n : \| A \|_n \to \| \Omega\,(\mathsf{Susp}\,A) \|_n$ is an equivalence. This allows us to prove the following result:

Corollary 3.10. *For $n \geq 1$, we have $\pi_n(\mathbb{S}^n) \cong \mathbb{Z}$. Furthermore, $\pi_n(\mathbb{S}^n)$ is generated by $i_n = |\,\mathsf{id}_{\mathbb{S}^n}\,|$.*

Proof. The synthetic proof of the classical result that $\pi_1(\mathbb{S}^1) \cong \mathbb{Z}$ is due to Licata and Shulman [LS13]. The fact that $\pi_2(\mathbb{S}^2) \cong \pi_1(\mathbb{S}^1)$ is given by the LES associated to the Hopf fibration combined with Proposition 3.8. The fact that $\pi_{n+1}(\mathbb{S}^{n+1}) \cong \pi_n(\mathbb{S}^n)$ is an immediate consequence of Theorem 3.9. The second statement follows by induction on n, using that suspension is functorial and thereby preserves the identity map. □

We have now analysed all homotopy groups $\pi_n(\mathbb{S}^m)$ with $n \leq m$. This yields the following:

Proposition 3.11. *Post-composition by hopf induces an isomorphism $\pi_3(\mathbb{S}^3) \cong \pi_3(\mathbb{S}^2)$.*

Proof. By Proposition 3.2 and Proposition 3.6, we get an exact sequence

$$\pi_3(\mathbb{S}^1) \to \pi_3(\mathbb{S}^3) \xrightarrow{\mathsf{hopf}_*} \pi_3(\mathbb{S}^2) \to \pi_2(\mathbb{S}^1)$$

as $\pi_n(\mathbb{S}^1)$ vanishes for $n > 1$, hopf_* is an isomorphism. □

Corollary 3.12. *There is an isomorphism $\psi : \pi_3(\mathbb{S}^2) \cong \mathbb{Z}$. Furthermore, $\pi_3(\mathbb{S}^2)$ is generated by hopf.*

Proof. By Corollary 3.10 we know that $\pi_3(\mathbb{S}^3)$ is generated by the identity function on $\mathbb{S}^3$. We know that the isomorphism $\pi_3(\mathbb{S}^3) \cong \pi_3(\mathbb{S}^2)$ is given by post-composition by hopf and thus the generator of $\pi_3(\mathbb{S}^3)$ is mapped to hopf. □

3.1. **Formalisation of Brunerie's Chapter 2.** Most of these results have already been added to `agda/cubical` by Mörtberg & Pujet [MP20], Ljungström [Lju20], and Brunerie, Ljungström & Mörtberg [BLM22]. The Freudenthal suspension theorem was formalised in `Cubical Agda` by Cavallo [Cav20], using a direct cubical proof following [Uni13, Theorem 8.6.4]. Corollary 3.10 was given a direct proof, following the computation of cohomology groups of spheres in [BLM22].

There were some technical difficulties related to the equivalence $\omega_n : \Omega^n\,A \simeq (\mathbb{S}^n \to_\star A)$, which is used to show that the two different definitions of homotopy groups are equivalent. In several proofs, it is more natural to work on the left-hand-side of ω_n. At the same time, working on the right-hand-side often makes constructing elements easier (compare, for instance, an explicit description of the generator of $i_3 : \pi_3(\mathbb{S}^3)$ described as a 3-loop in

$\mathbb{S}^3$ to the very compact definition $i_3 = |\,\mathsf{id}_{\mathbb{S}^3}\,|$). This means that we often have to translate between the two definitions. One particularly important example is the LES of homotopy groups associated to a function $A \to_\star B$. On each level, the maps are given as follows:

$$\Omega^n\,(\mathsf{fib}\,f) \xrightarrow{\Omega^n\,\mathsf{fst}} \Omega^n\,A \xrightarrow{\Omega^n\,f} \Omega^n\,B$$

This is then transported to the definition of homotopy groups as maps from spheres via ω_n. For the proof of e.g. Corollary 3.12, we need to know that the maps in the sequence are given as follows:

$$\pi_n(\mathsf{fib}\,f) \xrightarrow{\mathsf{fst}_*} \pi_n(A) \xrightarrow{f_*} \pi_n(B)$$

What we need is then more than just an equivalence $\omega_n : \Omega^n\,A \simeq (\mathbb{S}^n \to_\star A)$ – we need to show that this equivalence is functorial. This is implicitly assumed in Brunerie's thesis, but, in `Cubical Agda`, we need to make it precise. Formalising this fact is not entirely trivial. First, we need a tractable definition of the equivalence in question. It can be described inductively with base case $\omega_1 : \Omega\,A \to (\mathbb{S}^1 \to_\star A)$ given by:

$$\begin{aligned}\omega_1\,p\,\mathsf{base} &= \star_A\\ \omega_1\,p\,(\mathsf{loop}\,i) &= p\,i\end{aligned}$$

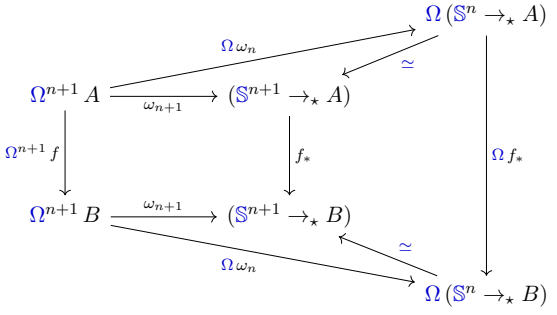
which we take to be pointed by refl. It is easy to verify that this is an equivalence. We define ω_{n+1} by the composition:

$$\Omega^{n+1}\,A = \Omega\,(\Omega^n\,A) \xrightarrow{\Omega\,\omega_n} \Omega\,(\mathbb{S}^n \to_\star A) \xrightarrow{\mathsf{funExt}^-_\star} (\mathbb{S}^n \to_\star \Omega\,A) \to (\mathbb{S}^{n+1} \to_\star A)$$

where the last arrow comes from the adjunction $\mathsf{Susp} \dashv \Omega$. This is a composition of equivalences, and hence an equivalence. We then need to verify that the following commutes

$$\begin{array}{ccc}
\Omega^n\,A & \xrightarrow{\omega_n} & (\mathbb{S}^n \to_\star A)\\
{\scriptstyle \Omega^n\,f}\downarrow & & \downarrow{\scriptstyle f_*}\\
\Omega^n\,B & \xrightarrow[\omega_n]{} & (\mathbb{S}^n \to_\star B)
\end{array}$$

This can be proved inductively. The base case is easy and the inductive step is given by the following diagram



where the commutativity of the outer square comes from the base case paired with the inductive hypothesis, the triangles from the definition of ω_{n+1} and the right-most square from a straightforward argument.

4. The Brunerie number

Here we give an overview of the first half of Brunerie's proof. This corresponds to [Bru16a, Chapter 3] and culminates in the isomorphism $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/\beta\mathbb{Z}$ for an at this point unknown "Brunerie number" $\beta : \mathbb{Z}$. We also discuss the formalisation of this part of the proof and various simplifications found during the formalisation.

4.1. The James construction.

To define β, Brunerie uses the *James construction* [Jam55], which he introduced in HoTT and partially formalised in [Bru19].

Proposition 4.1 (James construction)**.** *For a $(k \geq 0)$-connected pointed type A, there are types $\mathsf{J}_n\, A$ with inclusions*

$$\mathsf{J}_0\, A \xhookrightarrow{j_0} \mathsf{J}_1\, A \xhookrightarrow{j_1} \mathsf{J}_2\, A \xhookrightarrow{j_2} \cdots$$

such that its sequential colimit $\mathsf{J}_\infty\, A \simeq \Omega\,(\mathsf{Susp}\, A)$. Furthermore, $j_n : \mathsf{J}_n\, A \hookrightarrow \mathsf{J}_{n+1}\, A$ is $(n(k+1)+(k-1))$-connected.

A consequence of Proposition 4.1 is the following fact

Proposition 4.2. *Given a $(k \geq 0)$-connected type A, there is a $(3k+1)$-connected map $\mathsf{J}_2\, A \to \Omega\,(\mathsf{Susp}\, A)$.*

The proof of Proposition 4.2 uses that $\mathsf{J}_\infty\, A$, the sequential colimit of the sequence in Proposition 4.1, can be shown to be equivalent to $\Omega\,(\mathsf{Susp}\, A)$. This, paired with some results on the connectivity of sequential colimits, gives the statement. A key consequence of this is the following result which allows us to express $\pi_4(\mathbb{S}^3)$ as $\pi_3(\mathsf{J}_2\, \mathbb{S}^2)$ – a group which turns out to be quite a bit easier to reason about.

Theorem 4.3. $\pi_4(\mathbb{S}^3) \cong \pi_3(\mathsf{J}_2\, \mathbb{S}^2)$

Proof. Because $\mathbb{S}^2$ is 1-connected, Proposition 4.2 tells us that there is a 4-connected map

$$\mathsf{J}_2\, \mathbb{S}^2 \to \Omega\,(\mathsf{Susp}\, \mathbb{S}^2) = \Omega\,(\mathbb{S}^3)$$

In particular, it is 3-connected and induces an equivalence $\| \mathsf{J}_2\, \mathbb{S}^2 \|_3 \simeq \| \Omega\, \mathbb{S}^3 \|_3$. We get:

$$\pi_4(\mathbb{S}^3) \cong \pi_3(\Omega\, \mathbb{S}^3) \cong \pi_3(\mathsf{J}_2\, \mathbb{S}^2)$$

□

4.2. **Formalisation of the James construction.** This is a particularly technical part of Brunerie's thesis, primarily due to the high number of higher coherences which need to be verified in the proof of Proposition 4.1. While this has, subsequent to our efforts, been formalised in its entirety by Kang [Kan22a], we have taken a shortcut by giving a direct proof of Theorem 4.3, which means we do not in fact need the full James construction. Consequently, we instead give direct definitions of $\mathsf{J}_n\, A$ for $n \leq 2$ for a pointed type A.

Definition 4.4 (Low dimensional James construction)**.** We define $\mathsf{J}_0\, A = \mathbb{1}$ and $\mathsf{J}_1\, A = A$. The type $\mathsf{J}_2\, A$ is defined as the pushout:

$$\begin{array}{ccc} A \vee A & \xrightarrow{\nabla} & A \\ {\scriptstyle i^\vee}\downarrow & \ulcorner & \downarrow \\ A \times A & \longrightarrow & \mathsf{J}_2\, A \end{array}$$

We remark that the construction in Definition 4.4 is not definitionally the same as Brunerie's; in his thesis, these constructions are theorems rather than definitions. Here we take them as definitions. With $\mathsf{J}_n\, A$ defined this way, the map $j_0 : \mathsf{J}_0\, A \to \mathsf{J}_1\, A$ is just the constant pointed map and $j_1 : \mathsf{J}_1\, A \to \mathsf{J}_2\, A$ is inr.

Before we continue, let us temporarily redefine $\mathbb{S}^2$ to be the following equivalent HIT. This will make some of the following constructions more compact.

```
data 𝕊² : Type where
  base : 𝕊²
  surf : refl base ≡ refl base
```

The next lemma will be crucial. It is a special case of the *Wedge Connectivity Lemma* [Uni13, Lemma 8.6.2], of which we have formalised a version of the proof of the sphere case in [BLM22, Lemma 8]. From the point of view of formalisation, this proof is easier to work with since it gives more useful definitional equalities.

Lemma 4.5 (Wedge connectivity for $\mathbb{S}^2$)**.** *Let* $P : \mathbb{S}^2 \times \mathbb{S}^2 \to 2\text{-}\mathsf{Type}$. *Any function* $f : (x : \mathbb{S}^2 \times \mathbb{S}^2) \to P\, x$ *is induced by the following data:*

$$\begin{aligned} f_l &: (x : \mathbb{S}^2) \to P(x\,, \mathsf{base}) \\ f_r &: (y : \mathbb{S}^2) \to P(\mathsf{base}\,, y) \\ f_{lr} &: f_l\, \mathsf{base} \equiv f_r\, \mathsf{base} \end{aligned}$$

Before we discuss the formalisation of Theorem 4.3 stated with the low dimensional James construction, we first construct the following function. The goal is to define a family of equivalences $f_x : \| \mathsf{J}_2\, \mathbb{S}^2 \|_3 \simeq \| \mathsf{J}_2\, \mathbb{S}^2 \|_3$ over $x : \mathbb{S}^2$. We do this by truncation elimination and pattern matching on x, starting with the base case:

$$\begin{aligned} f_{\mathsf{base}}\, |\, \mathsf{inl}\, (x, y)\, | &= |\, \mathsf{inl}\, (x\,, y)\, | \\ f_{\mathsf{base}}\, |\, \mathsf{inr}\, z\, | &= |\, \mathsf{inl}\, (\mathsf{base}\,, z)\, | \\ f_{\mathsf{base}}\, |\, \mathsf{push}\, (\mathsf{inl}\, x)\, i\, | &= |\, (\mathsf{push}\, (\mathsf{inl}\, x) \cdot \mathsf{push}\, (\mathsf{inr}\, x)^{-1})\, i\, | \\ f_{\mathsf{base}}\, |\, \mathsf{push}\, (\mathsf{inr}\, y)\, i\, | &= |\, \mathsf{inl}\, (\mathsf{base}\,, y)\, | \\ f_{\mathsf{base}}\, |\, \mathsf{push}\, (\mathsf{push}\, y\, j)\, i\, | &= \ldots \end{aligned}$$

where the omitted step consists of a proof that $\mathsf{push}\, (\mathsf{inl}\, \mathsf{base}) \cdot \mathsf{push}\, (\mathsf{inr}\, \mathsf{base})^{-1} \equiv \mathsf{refl}$. It is an easy lemma that f_{base} is equal to the identity on $\| \mathsf{J}_2\, \mathbb{S}^2 \|_3$. To complete the definition of

f_x, we need to consider the case when $x = \mathsf{surf}\, i\, j$. This amounts to providing a dependent function:

$$f_{\mathsf{surf}} : (x : \| \mathsf{J}_2\, \mathbb{S}^2 \|_3) \to \Omega^2\, (\| \mathsf{J}_2\, \mathbb{S}^2 \|_3\, , f_{\mathsf{base}}\, x)$$

To do this, we will, in particular, need to provide a family of fillers

$$Q_{(x\, ,\, y)} : \mathsf{refl}_{|\, \mathsf{inl}\, (x\, ,\, y)\, |} \equiv \mathsf{refl}_{|\, \mathsf{inl}\, (x\, ,\, y)\, |}$$

This is a 1-type, and thus Lemma 4.5 applies. We define:

$$Q_{(\mathsf{base}\, ,\, y)}\, i\, j = |\, \mathsf{inl}\, (\mathsf{surf}\, i\, j\, , y)\, |$$
$$Q_{(x\, ,\, \mathsf{base})}\, i\, j = |\, \mathsf{inl}\, (x\, , \mathsf{surf}\, i\, j)\, |$$

The fact that these two constructions agree when both x and y are base is a technical but relatively straightforward lemma. Thereby, $Q_{(x\, ,\, y)}$ is defined. We may now define f_{surf}:

$$f_{\mathsf{surf}}\, |\, \mathsf{inl}\, (x\, , y)\, | = Q_{(x\, ,\, y)}$$
$$f_{\mathsf{surf}}\, |\, \mathsf{inr}\, z\, | = Q_{(\mathsf{base}\, ,\, z)}$$

The higher cases are easy due to the fact that the goal becomes 0-truncated, making it sufficient to define them for $\mathsf{base} : \mathbb{S}^2$. Thus, f_x is defined for all $x : \mathbb{S}^2$.

Lemma 4.6. *For $x : \mathbb{S}^2$, f_x is an automorphism on $\| \mathsf{J}_2\, \mathbb{S}^2 \|_3$.*

Proof. To make coming proofs easier, this is proved by explicitly constructing the inverse analogously to f_x.

$$f^{-1}_{\mathsf{base}}\, x = f_{\mathsf{base}}\, x$$
$$f^{-1}_{\mathsf{surf}}\, x = f_{\mathsf{surf}^{-1}}\, x$$

Proving that these cancel is technical, but direct. □

We are now ready to prove the following statement, which is a rephrasing of Theorem 4.3.

Proposition 4.7. $\Omega\, \| \mathbb{S}^3 \|_4 \simeq \| \mathsf{J}_2\, \mathbb{S}^2 \|_3$

Proof. We take $\mathbb{S}^3 = \mathsf{Susp}\, \mathbb{S}^2$, where $\mathbb{S}^2$ is defined using base/surf as above. We employ the encode-decode method and define a family of 3-types over $\| \mathbb{S}^3 \|_4$. Since the universe of 3-types is a 4-type, we may do so by truncation elimination:

$$\mathsf{Code} : \| \mathbb{S}^3 \|_4 \to 3\text{-}\mathsf{Type}$$
$$\mathsf{Code}\, |\, \mathsf{north}\, | \; = \| \mathsf{J}_2\, \mathbb{S}^2 \|_3$$
$$\mathsf{Code}\, |\, \mathsf{south}\, | = \| \mathsf{J}_2\, \mathbb{S}^2 \|_3$$
$$\mathsf{Code}\, |\, \mathsf{merid}\, x\, i\, | = \mathsf{ua}\, f_x\, i$$

We now need to define two families of functions

$$\mathsf{encode}_x : |\, \mathsf{north}\, | \equiv x \to \mathsf{Code}\, x$$
$$\mathsf{decode}_x : \mathsf{Code}\, x \to |\, \mathsf{north}\, | \equiv x$$

over $x : \| \mathbb{S}^3 \|_4$. We define encode_x by path induction, sending refl to the basepoint in $\| \mathsf{J}_2\, \mathbb{S}^2 \|_3$. We define decode_x by truncation elimination and pattern matching on x. The

crucial step is defining $\mathsf{decode}_{|\,\mathsf{north}\,|} : \| \mathsf{J}_2\,\mathbb{S}^2 \|_3 \to \Omega\,\| \mathbb{S}^3 \|_4$. On point constructors, it is given by

$$\mathsf{decode}_{|\,\mathsf{north}\,|}\,(\mathsf{inl}\,(x\,,y)) = \sigma\,x \cdot \sigma\,y$$
$$\mathsf{decode}_{|\,\mathsf{north}\,|}\,(\mathsf{inr}\,z) = \sigma\,z$$

which is easily verified to be coherent with the higher constructors. The case $\mathsf{decode}_{|\,\mathsf{south}\,|}$ is immediately induced by $\mathsf{decode}_{|\,\mathsf{north}\,|}$, since $\mathsf{north} \equiv \mathsf{south}$ via $\mathsf{merid}\ \mathsf{base}$. The case $\mathsf{decode}_{|\,\mathsf{merid}\ a\,i\,|}\,y$ amounts to showing that

$$\mathsf{decode}_{|\,\mathsf{north}\,|}(f_a^{-1}\,y) \equiv \mathsf{decode}_{|\,\mathsf{north}\,|}\,y \cdot (\sigma\,|\,a\,|)^{-1}$$

The proof is technical but is greatly aided by Lemma 4.5. The fact that $\mathsf{decode}_x(\mathsf{encode}_x\,p) \equiv p$ for each $p : \mathsf{north} \equiv x$ holds by path induction. Finally, the fact that $\mathsf{encode}_{\mathsf{north}}(\mathsf{decode}_{\mathsf{north}}\,y) \equiv y$ holds for each $y : \| \mathsf{J}_2\,\mathbb{S}^2 \|_3$ holds by some technical but simple path algebra. Hence $\mathsf{decode}_{|\,\mathsf{north}\,|} : \Omega\,\| \mathbb{S}^3 \|_4 \to \| \mathsf{J}_2\,\mathbb{S}^2 \|_3$ is an equivalence. □

We get Theorem 4.3 as an immediate corollary of Proposition 4.7 via the same sequence of isomorphisms as in the proof of Theorem 4.3.

4.3. **Definition of the Brunerie number.** Brunerie's goal is now to analyse $\pi_3(\mathsf{J}_2\,\mathbb{S}^2)$. The first result needed is the following:

Definition 4.8 (Whitehead map)**.** Given two pointed types A and B, there is a map:

$\mathsf{W} : A * B \to \mathsf{Susp}\ A \vee \mathsf{Susp}\ B$
$\mathsf{W}\ (\mathsf{inl}\ a) = \mathsf{inr}\ \mathsf{north}$
$\mathsf{W}\ (\mathsf{inr}\ b) = \mathsf{inl}\ \mathsf{north}$
$\mathsf{W}\ (\mathsf{push}\ (a\ ,\ b)\ i) = (\mathsf{cong}\ \mathsf{inr}\ (\sigma\ b) \cdot \mathsf{push}\ \star_1\ ^{-1} \cdot \mathsf{cong}\ \mathsf{inl}\ (\sigma\ a))\ i$

For our purposes, we only need the case when $A = B = \mathbb{S}^1$ (although all of the following results appear in full generality in Brunerie's thesis). We get a composite map:

$$\mathsf{e} : \mathbb{S}^3 \xrightarrow{\simeq} \mathbb{S}^1 * \mathbb{S}^1 \xrightarrow{\mathsf{W}} \mathbb{S}^2 \vee \mathbb{S}^2$$

This induces, via pre-composition, a *Whitehead product*:

$$\pi_2(\mathbb{S}^2) \times \pi_2(\mathbb{S}^2) \xrightarrow{[-,-]} \pi_3(\mathbb{S}^2)$$

by

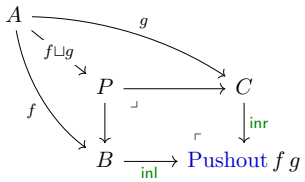
$$[|\,f\,|, |\,g\,|] := |\,\nabla \circ (f \vee g) \circ \mathsf{e}\,|$$

Recall that we denote by i_2 the generator of $\pi_2(\mathbb{S}^2)$. Brunerie shows, in particular, the following about its relation to the Whitehead product (see [Bru16a, Proposition 3.4.4.] for the full statement).

Theorem 4.9. *The kernel of the suspension map* $\sigma_* : \pi_3(\mathbb{S}^2) \to \pi_4(\mathbb{S}^3)$ *is generated by* $[i_2, i_2]$.

The key technical component in the proof is the *Blakers-Massey Theorem*, first formalised in HoTT by Favonia, Finster, Licata & Lumsdaine in [HFLL16]:

Theorem 4.10 (Blakers-Massey)**.** *Consider the diagram*

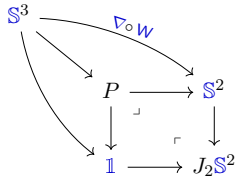


where P is the pullback along inl *and* inr*, i.e. $P = \Sigma_{(b,c):B\times C}(\mathsf{inl}\, b \equiv \mathsf{inr}\, c)$, and $f \sqcup g$ is defined by*

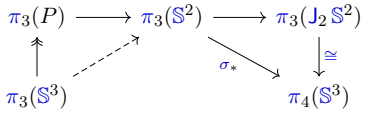
$$(f \sqcup g)\, a = (f\, a\, ,\, g\, a\, ,\, \mathsf{push}\, a)$$

If f and g are n- respectively m-connected, then $f \sqcup g$ is $(n+m)$-connected.

Theorem 4.9 is proved by considering the following diagram



Verifying that the outer square is a pushout square is technical and we refer to Brunerie's proof for the details. Above, P is simply the fibre of $\mathsf{inr} : \mathbb{S}^2 \to \mathsf{J}_2\,\mathbb{S}^2$. The leftmost map is 2-connected since $\mathbb{S}^3$ is 2-connected and the top map is 0-connected since $\mathbb{S}^3$ and $\mathbb{S}^2$ are both 1-connected. Consequently, by Theorem 4.10, we get that the map $\mathbb{S}^3 \to P$ is 2-connected and thus induces a surjection after application of π_3. This gives the diagram:



where the sequence on the top comes from the long exact sequence of homotopy groups associated to P. The dashed map sends the generator $i_3 : \pi_3(\mathbb{S}^3)$ to $[i_2, i_2] : \pi_3(\mathbb{S}^2)$ by definition.

Theorem 4.9 motivates the following definition. Recall that we denote by ψ the isomorphism $\pi_3(\mathbb{S}^2) \cong \mathbb{Z}$.

Definition 4.11 (Brunerie number)**.** We define the Brunerie number $\beta : \mathbb{Z}$ by $\beta = \psi[i_2, i_2]$.

We may now prove the main result of [Bru16a, Chapter 3].

Corollary 4.12. $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/\beta\mathbb{Z}$.

Proof. We have a homomorphism $\sigma_* \circ \psi^{-1} : \mathbb{Z} \to \pi_4(\mathbb{S}^3)$. This composition is surjective since ψ is an isomorphism and $\sigma_* : \pi_3(\mathbb{S}^2) \to \pi_4(\mathbb{S}^3)$ is surjective by Theorem 3.9. Since, by Theorem 4.9, the kernel of σ_* is generated by $[i_2, i_2]$, the kernel of $\sigma_* \circ \psi^{-1}$ is generated by $\psi[i_2, i_2]$, i.e. by β. The statement then follows from the first isomorphism theorem. □

4.4. **Formalisation of the definition of the Brunerie number.** The formalisation of this part was straightforward. Arguably the most technical result, the Blakers-Massey theorem, was already available in the library thanks to Kang [Kan22b]. Most of the remaining results were essentially just diagram chases which, in a proof assistant, can be somewhat technical. Most work went into verifying that $\mathsf{J}_2\,\mathbb{S}^2$ is the cofibre of $\nabla \circ \mathsf{W}$, the proof of which followed Brunerie's closely.

In this section we, found the only obvious mistake in Brunerie's thesis. On page 82, in his definition of the push-case for W, the path component in the middle was not inverted, making the term ill-typed. Naturally, this was of no mathematical significance and something Brunerie immediately would have noticed if he would have attempted to provide a computer formalisation of this construction.

5. Brunerie's proof that $|\beta| \equiv 2$

This section concerns the final three Chapters (4–6) of Brunerie's thesis. The main goal here is proving that $|\beta| \equiv 2$.

We will not discuss Chapter 4 in much detail. Chapter 4 is devoted to smash products and, in particular, their symmetric monoidal structure. Brunerie used this in subsequent chapters to define and prove properties about the *cup product*, a graded multiplicative operation on cohomology groups which will be used to show that $|\beta| \equiv 2$. This chapter has turned out to be incredibly difficult to formalise due to the large number of higher coherences involved in the proofs [Bru18]. In fact, the results of this chapter were proved in detail and fully formalised only 8 years after the publication of Brunerie's thesis by Ljungström [Lju24]. We remark that despite the fact that these results have now been made available to us, they are not needed. While, with these results, Brunerie's construction of the cup product appears correct, his use of smash products still leads to some rather cumbersome diagram chases (with many coherences which still need verification).

Luckily, it turns out that Chapter 4 can be avoided altogether and that this in fact makes some difficult proofs later on very direct. For this reason, the results in Chapter 4 were omitted completely from our formalisation. The reason for this is that all results regarding smash products in Brunerie's thesis concern, in some way, pointed maps out of smash products. In this case, we may exploit the adjunction of maps out of smash products and bi-pointed maps:

$$(A \wedge B \to_\star C) \simeq (A \to_\star (B \to_\star C))$$

Here, $B \to_\star C$ is taken to be pointed by the constant map. As shown by Brunerie et al. [BLM22], it is arguably easier to define the cup product on the right-hand side of the adjunction, which effectively means that we never have to work with smash products when formalising cohomology theory. The usefulness of the approach by Brunerie et al. [BLM22] is not only witnessed by our work—it has been used by Lamiaux et al. [LLM23] and Ljungström & Mörtberg [LM24] in the development of cohomology rings and is used to describe the cup product as an instance of the delooping machinery introduced by Wärn [Wär23, Section 4.3]. We remark that the same techniques (although independent from [BLM22]) can be found in the work of Christensen & Scoccola [CS20, Section 2.4] where it is utilised in a discussion of the magma structure on loop spaces.

5.1. **Cohomology and the Hopf invariant.** [Bru16a, Chapter 5] introduces integral cohomology groups and rings, and gives a construction of the Mayer-Vietoris sequence. In more detail, Brunerie defines the integral Eilenberg-MacLane spaces by $\mathsf{K}_0 = \mathbb{Z}$ and $\mathsf{K}_n = \| \mathbb{S}^n \|_n$ for $n \geq 1$. This allows for a definition of the (integral) cohomology of X:

$$\mathsf{H}^n(X) = \| X \to \mathsf{K}_n \|_0$$

The fact that $\Omega \mathsf{K}_{n+1} \simeq \mathsf{K}_n$ follows by a proof completely analogous to that of Corollary 3.10. Brunerie uses this equivalence to carry over the (commutative) H-space structure on $\Omega \mathsf{K}_{n+1}$ to that of K_n. This provides a notion of addition $+_k : \mathsf{K}_n \times \mathsf{K}_n \to \mathsf{K}_n$ which lifts to $\mathsf{H}^n(X)$ by post-composition, thereby endowing $\mathsf{H}^n(X)$ with a group structure.

Similarly, Brunerie gives a definition of a cup product $\smile_k : \mathsf{K}_n \wedge \mathsf{K}_m \to \mathsf{K}_{n+m}$ which lifts to the usual cup product $\smile : \mathsf{H}^n(X) \times \mathsf{H}^m(X) \to \mathsf{H}^{n+m}(X)$. This is shown to induce a graded commutative ring structure on $\mathsf{H}^*(X)$ using results from Chapter 4.

The synthetic construction of the Mayer-Vietoris sequence concerns the long exact sequence

$$\begin{array}{ccccc} \mathsf{H}^0(D) & \longrightarrow & \mathsf{H}^0(B) \times \mathsf{H}^0(C) & \longrightarrow & \mathsf{H}^0(A) \\ \mathsf{H}^1(D) & \longrightarrow & \dots & & \end{array}$$

where D denotes the pushout of a span $B \xleftarrow{f} A \xrightarrow{g} C$. A direct application gives us, for $n \geq 1$, that $\mathsf{H}^n(\mathbb{S}^m) \cong \mathbb{Z}$ if $n = m$ and $\mathsf{H}^n(\mathbb{S}^m) \cong \mathbb{1}$ otherwise. This gives, by another application of the sequence, the following result:

Lemma 5.1. *For any $f : \mathbb{S}^3 \to \mathbb{S}^2$ we have*

$$\mathsf{H}^n(\mathsf{cofib}\, f) \cong \begin{cases} \mathbb{Z} & n \in \{0, 2, 4\} \\ \mathbb{1} & \text{otherwise} \end{cases}$$

Let us briefly fix $f : \mathbb{S}^3 \to \mathbb{S}^2$. Denote by γ_2 and γ_4 the generators of $\mathsf{H}^2(\mathsf{cofib}\, f)$ and $\mathsf{H}^4(\mathsf{cofib}\, f)$ respectively given by the image of $1 : \mathbb{Z}$ under the isomorphism in Lemma 5.1. These generators may be used to define an invariant on $\mathbb{S}^3 \to \mathbb{S}^2$ called the *Hopf invariant*. This is done as follows:

Definition 5.2 (Hopf invariant)**.** The *Hopf invariant* of f is the unique integer $\mathsf{HI}\, f : \mathbb{Z}$ such that $\gamma_2 \smile \gamma_2 \equiv \mathsf{HI}\, f \cdot \gamma_4$.

We remark that the above definition is given for the more general class of maps $\mathbb{S}^{2n-1} \to \mathbb{S}^n$ in Brunerie's thesis. For our purposes, the above special case suffices. In particular, we may see HI as a function $\pi_3(\mathbb{S}^2) \to \mathbb{Z}$. The following turns out to be true:

Proposition 5.3. HI *is a homomorphism $\pi_3(\mathbb{S}^2) \to \mathbb{Z}$.*

Proof sketch. We first rephrase $f + g : \pi_3(\mathbb{S}^2)$ as a composition

$$\mathbb{S}^3 \to \mathbb{S}^3 \vee \mathbb{S}^3 \xrightarrow{f \vee g} \mathbb{S}^2 \vee \mathbb{S}^2 \xrightarrow{\nabla} \mathbb{S}^2$$

By analysing the cohomology of $\mathsf{cofib}\,(\nabla \circ (f \vee g))$ and the action on generators of the obvious maps from $\mathsf{cofib}\,(\nabla \circ (f \vee g))$, $\mathsf{cofib}\, f$ and $\mathsf{cofib}\, g$ into $\mathsf{cofib}\,(f + g)$, one arrives at the result with some elementary algebra. □

Finally, the Hopf invariant of our element of interest $[i_2, i_2]$ is computed (up to a sign), using an argument similar to that of the proof of Proposition 5.3.

Proposition 5.4. $|\mathsf{Hl}\,[i_2, i_2]| \equiv 2$

We are now almost done: if there is an element $f : \pi_3(\mathbb{S}^2)$ such that $\mathsf{Hl}\,f \equiv 1$, then Hl is an isomorphism $\pi_3(\mathbb{S}^2) \cong \mathbb{Z}$. Since isomorphisms of this type are unique up to a sign, Proposition 5.4 tells us that also for the standard isomorphism $\psi : \pi_3(\mathbb{S}^2) \cong \mathbb{Z}$, we must have $|\psi[i_2, i_2]| \equiv 2$, i.e. $|\beta| \equiv 2$. Hence, we have so far shown the following:

Lemma 5.5. *If* $\mathsf{Hl}\,f \equiv 1$ *for some* $f : \pi_3(\mathbb{S}^2)$*, then* $|\beta| \equiv 2$.

The final chapter of Brunerie's thesis is devoted to proving the antecedent of Lemma 5.5.

5.2. **Formalisation of cohomology and the Hopf invariant.** This section was largely covered by Brunerie, Ljungström and Mörtberg in [BLM22] and thus also available in `agda/cubical`. We briefly summarise:

- $+_k : \mathsf{K}_n \times \mathsf{K}_n \to \mathsf{K}_n$ was defined explicitly using a direct construction of the *Wedge Connectivity* for spheres—a generalisation of Lemma 4.5. This construction is of great convenience to our formalisation due to the fact that e.g. $\star_{\mathsf{K}_n} +_k |\, x \,| \equiv |\, x \,|$ holds definitionally. In fact, all of the basic laws governing $+_k$ are (trivially) provably path equal to refl at $\star_{\mathsf{K}_n}$, which simplifies a lot of path algebra.
- The cup product is defined via the following lift

$$\begin{array}{ccc} \mathbb{S}^n & \longrightarrow & (\mathsf{K}_m \to_\star \mathsf{K}_{n+m}) \\ {\scriptstyle |\text{-}|}\downarrow & \nearrow & \\ \mathsf{K}_n & & \end{array}$$

  for $n \geq 1$, where the top map may be thought of as being inductively defined via the equivalence

$$\begin{aligned} &(\mathbb{S}^{n+1} \to_\star (\mathsf{K}_m \to_\star \mathsf{K}_{(n+1)+m})) \\ \simeq\ &(\mathbb{S}^n \to_\star (\mathsf{K}_m \to_\star \Omega\, \mathsf{K}_{(n+1)+m})) \\ \simeq\ &(\mathbb{S}^n \to_\star (\mathsf{K}_m \to_\star \mathsf{K}_{n+m})) \end{aligned}$$

  The lift exists because the type of pointed functions $\mathsf{K}_m \to_\star \mathsf{K}_{n+m}$ is an n-type. This construction gives an inductively defined cup product which is remarkably easy to work with, as showcased in [LLM23] to compute cohomology rings of various classical spaces.
- The Mayer-Vietoris sequence was formalised by directly translating Brunerie's original proof.

Hence, what remained to be formalised in Chapter 5 was the Hopf invariant, Proposition 5.3 and Proposition 5.4. The formalisation of these propositions was straightforward and we were able to translate Brunerie's proofs in a direct manner. This is not surprising as the proofs are very algebraic.

For simplicity, we only formalised these propositions as they stand here and not their generalisations to higher spheres (i.e. as in [Bru16a, Proposition 5.4.3 & 5.4.4]). We remark, however, that the formalised proofs easily should be rephrasable for the general Hopf invariant of maps $\mathbb{S}^{2n-1} \to \mathbb{S}^n$.

5.3. **The Gysin sequence.** This section corresponds to [Bru16a, Chapter 6]. In order to be able to apply Lemma 5.5, this chapter is devoted to proving that $|\mathsf{Hl}\ \mathsf{hopf}| \equiv 1$, where, recall, $\mathsf{hopf} : \mathbb{S}^3 \to \mathbb{S}^2$ is the Hopf map—the generator of $\pi_3(\mathbb{S}^2)$ from Definition 3.5. This amounts to analysing the cup product on the cohomology of $\mathsf{cofib}\,\mathsf{hopf}$. It is well known that $\mathsf{cofib}\,\mathsf{hopf}$ is a model of the complex projective plane $\mathbb{C}P^2$ (see e.g. [Hat02, Example 4.45]), so let us simply write $\mathbb{C}P^2$ from now on. We hence have $\mathbb{C}P^2$ defined as the following pushout:

$$\begin{array}{ccc} \mathbb{S}^3 & \xrightarrow{\mathsf{hopf}} & \mathbb{S}^2 \\ \downarrow & \ulcorner & \downarrow \\ \mathbb{1} & \longrightarrow & \mathbb{C}P^2 \end{array}$$

In order to show that $|\mathsf{Hl}\ \mathsf{hopf}| \equiv 1$, it suffices to show that $- \smile \gamma_2 : \mathsf{H}^2(\mathbb{C}P^2) \to \mathsf{H}^4(\mathbb{C}P^2)$ is an isomorphism for $\gamma_2 : \mathsf{H}^2(\mathbb{C}P^2)$ a generator. Brunerie does this by constructing the Gysin sequence.

Proposition 5.6 (The Gysin sequence)**.** *Let B be a pointed and 0-connected type and $P : B \to \mathsf{Type}$ be a fibration with $P\ \star_B \simeq \mathbb{S}^{n-1}$. Let $E = \Sigma_{b:B}(P\,b)$ be the total space of P. If there is a family of maps $c : (b : B) \to (\mathsf{Susp}\,(P\,b) \to_\star \mathsf{K}_n)$ with $c_{\star_B}$ a generator of $\mathsf{H}^n(\mathbb{S}^n)$, then there is an element $e_n : \mathsf{H}^n(B)$ and a long exact sequence*

$$\begin{array}{ccccc} & & \cdots & \longrightarrow & \mathsf{H}^{i-1}(B) \\ & \swarrow & & & \\ \mathsf{H}^{i-1}(E) & \longrightarrow & \mathsf{H}^{i-n}(B) & \xrightarrow{- \smile e_n} & \mathsf{H}^i(B) \\ & \searrow & & & \\ \mathsf{H}^i(E) & \longrightarrow & \cdots & & \end{array}$$

Moreover, c (and also e_n) exists when B is 1-connected.

In order to make use of this, we need the following result.

Proposition 5.7. *There is a fibration $P : \mathbb{C}P^2 \to \mathsf{Type}$ with $P\ \star_{\mathbb{C}P^2} \simeq \mathbb{S}^1$ and total space $\mathbb{S}^5$.*

Proposition 5.7 is a special case of the following result.

Proposition 5.8 (Iterated Hopf construction)**.** *Given an associative H-space A, let $h_A : A * A \to \mathsf{Susp}\,A$ denote the associated Hopf map. There is a fibration $\mathsf{cofib}\,h_A \to \mathsf{Type}$* with fibre A and total space $A * A * A$.

We consider the particular case when $A = \mathbb{S}^1$ in Proposition 5.8. In this case, the map $h_{\mathbb{S}^1} : \mathbb{S}^1 * \mathbb{S}^1 \to \mathbb{S}^2$ corresponds to the usual Hopf map under the equivalence $\mathbb{S}^1 * \mathbb{S}^1 \simeq \mathbb{S}^3$ and hence $\mathsf{cofib}\,h_{\mathbb{S}^1} \simeq \mathbb{C}P^2$. The total space of this is $\mathbb{S}^1 * \mathbb{S}^1 * \mathbb{S}^1$ which is equivalent to $\mathbb{S}^5$ by Proposition 3.4 and thus we have proved Proposition 5.7. The associated Gysin sequence gives us the main result of this section:

Proposition 5.9 (Hopf invariant of the Hopf map)**.** $|\mathsf{Hl}\,\mathsf{hopf}| \equiv 1$

Proof. Since $\mathbb{C}P^2$ is 1-connected, Proposition 5.6 combined with Proposition 5.7 gives us an element $e_2 : \mathsf{H}^2(\mathbb{C}P^2)$ and a sequence

$$\mathsf{H}^{i-1}(\mathbb{S}^5) \to \mathsf{H}^{i-2}(\mathbb{C}P^2) \xrightarrow{- \smile e_2} \mathsf{H}^i(\mathbb{C}P^2) \to \mathsf{H}^i(\mathbb{S}^5)$$

When $1 \leq i \leq 4$, $\mathsf{H}^i(\mathbb{S}^5)$ vanishes. Setting $i = 2$, we get that e_2 must be a generator of $\mathsf{H}^2(\mathbb{C}P^2)$, and thus equal to the generator $\gamma_2 : \mathsf{H}^2(\mathbb{C}P^2)$ up to a sign. Setting $i = 4$, we get that $- \smile e_2$ must be an isomorphism of groups $\mathsf{H}^2(\mathbb{C}P^2) \cong \mathsf{H}^4(\mathbb{C}P^2)$ and hence $e_2 \smile e_2$ is a generator. Consequently, so is $\gamma_2 \smile \gamma_2$, and thus $|\mathsf{HI\ hopf}| \equiv 1$. □

Proposition 5.9 combined with Lemma 5.5 gives the desired path: $|\beta| \equiv 2$. This completes Brunerie's proof and Corollary 4.12 gives us the main result:

Theorem 5.10. $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$

5.4. **Formalisation of the Gysin sequence.** Formalising the results from Chapter 6 was more challenging, but was greatly aided by the alternative construction of the cup product discussed above. The first technical lemma, which is crucial for the construction of the Gysin sequence is:

Lemma 5.11. *Given $x : \mathsf{K}_n$ and $y : \mathsf{K}_m$, we have*

$$\mathsf{cong}\ (\lambda\, a \to a \smile_k y)\ (\sigma_n\, x) \equiv \sigma_{n+m}(x \smile_k y)$$

In Brunerie's thesis, this lemma relies on a result which in turn requires the symmetric monoidal structure of the smash product (in particular, it uses the *pentagon identity*). With the alternative construction of the cup product, however, this result follows immediately from the definition of the cup product.

Lemma 5.11 is used to show that the map

$$g^i : \mathsf{K}_i \to (\mathbb{S}^n \to_\star \mathsf{K}_{i+n})$$
$$g^i\, x = \lambda y \to x \smile_k \iota y$$

is an equivalence, which is crucially used in the construction of the Gysin sequence. Above, $\iota : \mathbb{S}^n \to \mathsf{K}_n$ is a generator of $\mathsf{H}^n(\mathbb{S}^n)$. For reference, g^i is the map $g^i_{\star_B}$ in the proof of [Bru16a, Proposition 6.1.2]. Brunerie's proof proceeds by induction on i: the fact that g^0 is an equivalence is easy; for the inductive step, it suffices to show that $\Omega\, g^{i+1} : \Omega\, \mathsf{K}_{i+1} \to \Omega\, (\mathbb{S}^n \to_\star \mathsf{K}_{(i+1)+n})$ is an equivalence for reasons of connectedness. This is done by showing that the following diagram commutes

$$\begin{array}{ccc} \Omega\, \mathsf{K}_{i+1} & \xrightarrow{\Omega\, g^{i+1}} & \Omega\, (\mathbb{S}^n \to_\star \mathsf{K}_{(i+1)+n}) \\ \simeq\downarrow & & \downarrow\simeq \\ \mathsf{K}_i & \xrightarrow[g^i]{} & (\mathbb{S}^n \to_\star \mathsf{K}_{i+n}) \end{array}$$

hence getting that $\Omega\, g^{i+1}$ is an equivalence from the induction hypothesis.

While the general idea of Brunerie's proof of this statement is correct, it was difficult to formalise directly. The primary reason for this is that Brunerie does not pay much attention to the fact that the objects of interest are not just functions, but *pointed* functions. In particular, his argument for the commutativity of the diagram above treats $g^i\, x$ as a plain function rather than a pointed function. Fortunately for us, the whole proof is very direct with the alternative definition of the cup product. Formalising Brunerie's proof with pointedness of functions respected would have been hard, especially without machinery external to [Bru16a] (e.g. [BLM22, Lemma 14.]).

After these subtleties were dealt with, the formalisation of the Gysin sequence could proceed following Brunerie's proof closely. In our initial formalisation, we made a slight adjustment to the indexing of the Gysin sequence. This removed some bureaucracy but happened at the cost of generality.[3] This made verifying Proposition 5.9 slightly less direct, because we no longer had access to the case

$$\mathsf{H}^1(\mathbb{S}^5) \longrightarrow \mathsf{H}^0(\mathbb{C}P^2) \xrightarrow{- \smile e_2} \mathsf{H}^2(\mathbb{C}P^2) \longrightarrow \mathsf{H}^2(\mathbb{S}^5)$$

which is used by Brunerie to show that the element $e_2 : \mathsf{H}^2(\mathbb{C}P^2)$, for which $- \smile e_2 : \mathsf{H}^2(\mathbb{C}P^2) \to \mathsf{H}^4(\mathbb{C}P^2)$ is an isomorphism, is indeed a generator. However, in practice, this is not a big problem. In fact, it provides a nice example of a proof by computation. It is very direct to manually show that the map $i : \mathbb{C}P^2 \to \mathsf{K}_2$ induced by $i(\mathsf{inl}\, x) = |\, x\, |$ is equal to the underlying map of e_2. The fact that i generates $\mathsf{H}^2(\mathbb{C}P^2)$ can then be verified by computation: applying the isomorphism $\mathsf{H}^2(\mathbb{C}P^2) \cong \mathbb{Z}$ to $|\, i\, |$ returns 1 by normalisation in `Cubical Agda`. We stress, for those sceptical of this method, that it also is very direct to provide a "manual" formalisation of this fact.

The final step of the formalisation was Proposition 5.8, i.e. the iterated Hopf construction. Although technical, the formalisation could be carried out following Brunerie closely.

6. The simplified proof and normalisation of a Brunerie number

It turns out that not only Chapter 4, but also Chapters 5–6 can be avoided. As conjectured by Brunerie, it would be possible to do this by simply normalising the Brunerie number. While we still cannot normalise his original definition of it, we can at least provide a computation of a substantially simplified Brunerie number. This is defined via a more tractable description of the isomorphism $\pi_3(\mathbb{S}^2) \cong \mathbb{Z}$ as a composition of simpler isomorphisms, relying on an alternative definition of π_3 in terms of $\mathbb{S}^1 * \mathbb{S}^1$. The idea is then to trace $[i_2, i_2] : \pi_3(\mathbb{S}^2)$ step by step through these isomorphisms. This gives a sequence of new Brunerie numbers and one of these normalises to -2 in `Cubical Agda` in a matter of seconds.

The trick to give a more tractable definition of $\pi_3(\mathbb{S}^2) \cong \mathbb{Z}$ is to redefine the third homotopy group of a type A as $\pi_3^*(A) = \| \mathbb{S}^1 * \mathbb{S}^1 \to_\star A \|_0$. This reformulation of π_3 can be given an explicit group structure, such that pre-composition by $\mathbb{S}^1 * \mathbb{S}^1 \simeq \mathbb{S}^3$ induces an isomorphism $\pi_3(A) \cong \pi_3^*(A)$. Let us first set up machinery we need (and a bit more).

6.1. Interlude: joins and smash products of spheres.

We have seen that the equivalence $\mathbb{S}^3 \simeq \mathbb{S}^1 * \mathbb{S}^1$ played a crucial role in Brunerie's original proof. What is less clear, however, is what this equivalence actually looks like. It turns out that it is closely related to the multiplication $\mathbb{S}^1 \wedge \mathbb{S}^1 \to \mathbb{S}^2$ and, as such, has a rather direct and algebraic description. Let us therefore briefly study this multiplication and describe its relation to the decomposition of spheres into joins. Although we only need low-dimensional special cases of these facts, we take the opportunity to tell the general story.

Remark. *In this subsection only, we will use the definition* $\mathbb{S}^n := \mathsf{Susp}^n\, \mathsf{Bool}$. *In particular, this means that we redefine* $\mathbb{S}^1 := \mathsf{Susp}\, \mathsf{Bool}$ *instead of using the* base/loop *construction. This is only done for ease of presentation and is not used in the formalisation.*

[3] A more general form of the Gysin sequence using Brunerie's indexing has later been added to `agda/cubical`.

Let us use the following (explicit) definition of the smash product $A \wedge B$.

```
data _∧_ (A B : Pointed) : Type where
  ⋆∧ : A ∧ B
  ⟨_⟩ : A × B → A ∧ B
  push_l : (a : A) → ⟨ a , ⋆_B ⟩ ≡ ⋆∧
  push_r : (b : B) → ⟨ ⋆_A , b ⟩ ≡ ⋆∧
  push_lr : push_l ⋆_A ≡ push_r ⋆_B
```

This construction is well known and can easily be seen to be (bi-)functorial (see e.g. [Lju24, Definition 6]), i.e. given pointed maps $f : A \to_\star C$ and $g : B \to_\star D$, there is a map $f \wedge g : A \wedge B \to C \wedge D$ with $(f \wedge g)\langle x , y\rangle := \langle f\, x , g\, y\rangle$.

The first goal is to define a multiplication $\mathbb{S}^n \wedge \mathbb{S}^m \to \mathbb{S}^{n+m}$. To facilitate future proofs, we first introduce the following construction which lifts maps $A \times B \to C$ to maps $(\mathsf{Susp}\, A) \times B \to \mathsf{Susp}\, B$:

```
^_ : (A × B → C) → ((Susp A) × B → Susp C)
(^ f) (north , b) = north
(^ f) (south , b) = north
(^ f) (merid a i , b) = σ (f (a , b)) i
```

The function $\widehat{\ }f$ is pointed in the left-argument by construction. It is pointed also in the right-argument if this is the case for f. Hence, given any function $g : A \wedge B \to C$, we also get (with some abuse of notation) $\widehat{\ }g : (\mathsf{Susp}\, A) \wedge B \to \mathsf{Susp}\, C$.

Lemma 6.1. *For any pointed types A and B, the map $\widehat{\ }\,\mathsf{id}_{A\wedge B} : (\mathsf{Susp}\, A) \wedge B \to \mathsf{Susp}\,(A \wedge B)$ is an equivalence.*

Proof. The inverse of $\widehat{\ }\,\mathsf{id}_{A\wedge B}$ is induced by the map $A \times B \to \Omega\,((\mathsf{Susp}\, A) \wedge B)$ defined by mapping $(a, b) : A \times B$ to the composite loop given by:

$$\star_\wedge \xrightarrow{\mathsf{push}_r^{-1}} \langle \mathsf{north}, b\rangle \xrightarrow{\mathsf{cong}\ \langle -\,, b\rangle\ (\sigma\, a)} \langle \mathsf{north}\,, b\rangle \xrightarrow{\mathsf{push}_r} \star_\wedge$$

The fact that these maps cancel follows by some technical but elementary path algebra. For the details, we refer to the formalisation. □

Lemma 6.2. *If $f : A \wedge B \to C$ is an equivalence, then so is $\widehat{\ }f : (\mathsf{Susp}\, A) \wedge B \to \mathsf{Susp}\ A$*

Proof. Using equivalence induction (see e.g. [Uni13, Corollary 5.8.5]), it is enough to prove the lemma for $C := A \wedge B$ and $f := \mathsf{id}_{A\wedge B}$. In this case, the statement is precisely that of Lemma 6.1. □

These lemmas allow us to define an equivalence $\wedge_{n,m} : \mathbb{S}^n \wedge \mathbb{S}^m \simeq \mathbb{S}^{n+m}$ (we borrow this notation from [Bru16a, Proposition 4.2.2]). We will write $\smile : \mathbb{S}^n \times \mathbb{S}^m \to \mathbb{S}^{n+m}$ for the underlying function, i.e. $x \smile y := \wedge_{n,m}\langle x , y\rangle$. The name $\smile$ is suggestive: modulo the quotient maps $\mathbb{S}^\bullet \to \mathsf{K}_\bullet$, it is precisely the cup product; this justifies overloading the symbol. We define it by induction on n. In the case $n = 0$, we define it on canonical points $\langle x , y\rangle$ by case distinction on x:

$$\mathsf{false} \smile y = y$$
$$\mathsf{true} \smile y = \star_{\mathbb{S}^m}$$

This map induces a map on the full smash product $\mathbb{S}^0 \wedge \mathbb{S}^m$ [Bru16a, Section 4.1]. In fact, it is an equivalence, and thereby $\wedge_{0,m}$ is defined. For $n > 0$ we use the fact that $\mathbb{S}^n := \mathsf{Susp}\,\mathbb{S}^{n-1}$ and simply define $\wedge_{n,m} := \widehat{\ }\wedge_{n-1,m}$. By Lemma 6.1, this is an equivalence.

Let us try to transfer this construction from smash products to joins. To begin with, consider the following map, defined for any two pointed types A and B:

```
pinch : A * B → Susp (A ∧ B)
pinch (inl a) = north
pinch (inr b) = south
pinch (push (a , b) i) = merid ⟨ a , b ⟩ i
```

Proposition 6.3. *For any two pointed types A and B, the map* pinch *is an equivalence.*

Proof. While we have, in our formalisation, explicitly constructed an inverse of pinch and proved directly that the two maps cancel, a recent (independent) result by Cagne et al. [CBKB24] allows us to give a more principled proof. We proceed by noting that for any pointed type C, we have

$$(\mathsf{Susp}\,(A \wedge B) \to_\star C) \simeq (A \wedge B \to_\star \Omega\, C) \simeq (A \to_\star (B \to_\star \Omega\, C)) \simeq (A * B \to_\star C)$$

where the first equivalence comes from the adjunction between Susp and Ω and the second from the adjunction between smash products and doubly pointed maps. The third equivalence is [CBKB24, Lemma 6.1]. This shows that $\mathsf{Susp}\,(A \wedge B)$ and $A * B$ have the same elimination principle, which implies the desired statement. □

Let $\mathsf{F}_{n,m} : \mathbb{S}^n * \mathbb{S}^m \to \mathbb{S}^{n+m+1}$ denote the following composition:

$$\mathbb{S}^n * \mathbb{S}^m \xrightarrow{\mathsf{pinch}} \mathsf{Susp}\,(\mathbb{S}^n \wedge \mathbb{S}^m) \xrightarrow{\mathsf{Susp}\,(\wedge_{n,m})} \mathsf{Susp}\,\mathbb{S}^{n+m} =: \mathbb{S}^{n+m+1}$$

Unfolding the definition of $\mathsf{F}_{n,m}$, we see that it has an incredibly compact description:

```
F_{n,m} : 𝕊^n * 𝕊^m → 𝕊^{n+m+1}
F_{n,m} (inl a) = north
F_{n,m} (inr b) = north
F_{n,m} (push (a , b) i) = σ (a ⌣ b) i
```

Since $\mathsf{F}_{n,m}$ is a composition of two equivalences, we immediately arrive at the following result.

Proposition 6.4. $\mathsf{F}_{n,m}$ *is an equivalence*

We have already seen that the special case $\mathbb{S}^1 * \mathbb{S}^1 \simeq \mathbb{S}^3$ plays an important role in Brunerie's proof. Now that we have moved from the rather opaque definition of this equivalence presented in Brunerie's thesis to the definition in terms of the very explicit function $\mathsf{F}_{n,m}$, we can hope to better understand its role in the definition of the Brunerie number. Since the only non-trivial component of the construction of $\mathsf{F}_{n,m}$ is $\smile$, we may hope that it inherits some of its properties. We study these now.

Let us first analyse the interaction of $\smile$ with inversion. Recall, given a pointed type A we can define inversion on $\mathsf{Susp}\,A$ by:

```
- : Susp A → Susp A
- north = north
```

- south = north
- (merid a i) = σ a ($\tilde{}$ i)

We get sphere inversion by letting $- : \mathbb{S}^n \to \mathbb{S}^n$ be boolean negation when $n := 0$ and the suspension inversion defined above when $n > 1$.[4]

Proposition 6.5. *The multiplication $\smile$ is graded-commutative, i.e. for $x : \mathbb{S}^n$ and $y : \mathbb{S}^m$, we have $x \smile y \equiv -^{nm}(y \smile x)$.*

For the proof, we refer to [BLM22, Proposition 18] which is the corresponding statement for the cup product on $\mathsf{K}_n := \| \mathbb{S}^n \|_n$ and whose proof directly applies also in our setting. Associativity follows, just like in the proof of [BLM22, Proposition 17], by sphere induction:

Proposition 6.6. *The multiplication $\smile$ is associative.*

Proof. Let $x : \mathbb{S}^n$, $y : \mathbb{S}^m$ and $z : \mathbb{S}^k$. We show that $x \smile (y \smile z) \equiv (x \smile y) \smile z$ by induction on n and x. When $n = 0$, the two equalities

$$\mathsf{false} \smile (y \smile z) \equiv (\mathsf{false} \smile y) \smile z$$
$$\mathsf{true} \smile (y \smile z) \equiv (\mathsf{true} \smile y) \smile z$$

hold definitionally. For the inductive step, we use suspension elimination on $x : \mathbb{S}^{n+1}$. The two equalities

$$\mathsf{north} \smile (y \smile z) \equiv (\mathsf{north} \smile y) \smile z$$
$$\mathsf{south} \smile (y \smile z) \equiv (\mathsf{south} \smile y) \smile z$$

also hold definitionally. So, by inspection of the definition of $\smile$, we need to show that

$$\sigma(x \smile y) \equiv \mathsf{cong}\ (- \smile z)\ (\sigma(x \smile y))$$

Using the action of cong on path composition, we can unfold the right-hand side as follows:

$$\begin{aligned} \mathsf{cong}\ (- \smile z)\ (\sigma(x \smile y)) &\equiv \mathsf{cong}\ (- \smile z)\ (\mathsf{merid}\ (x \smile y)) \cdot \mathsf{cong}\ (- \smile z)\ (\mathsf{merid}\ \mathsf{north})^{-1} \\ &:= \sigma(x \smile y) \cdot \sigma(\mathsf{north} \smile y)^{-1} \\ &\equiv \sigma(x \smile y) \end{aligned}$$

□

6.2. **Homotopy groups in terms of joins.** As we have seen in Brunerie's construction of the Hopf map, it is often easier to describe maps of type $\mathbb{S}^n * \mathbb{S}^m \to A$ than those of type $\mathbb{S}^{n+m+1} \to A$. However, the definition of homotopy groups we have relied on so far uses the latter type. This forces us to translate back and forth whenever we want to use the definition in terms of joins. The key strategy behind our new calculation of the Brunerie number is to rephrase homotopy groups in terms of maps out of joins of spheres.

Definition 6.7. Given a pointed type A, we define $\pi^*_{n+m+1}(A) := \| \mathbb{S}^n * \mathbb{S}^m \to_\star A \|_0$.

Clearly, this is equivalent to the usual definition of $\pi_{n+m+1}(A)$ via pre-composition by $\mathsf{F}_{n,m}$. However, $\pi^*_{n+m+1}(A)$ can be endowed with an explicit group structure which $\mathsf{F}_{n,m}$ turns out to respect. In order to construct the group structure on $\pi^*_{n+m+1}(A)$, let us

[4] If we prefer to use the base/loop-construction of $\mathbb{S}^1$, we may define the inversion map simply by sending loop to loop^{-1}.

construct a map $\ell : A \times B \to \Omega\,(A * B)$ for all pointed types A and B. Recall, we take $A * B$ to be pointed by $\mathsf{inl}\,\star_A$. We define ℓ by:

$$\ell(a\,,b) := \mathsf{push}\,(\star_A\,,\star_B) \cdot \mathsf{push}\,(a\,,\star_B)^{-1} \cdot \mathsf{push}\,(a\,,b) \cdot \mathsf{push}\,(\star_A\,,b)^{-1}$$

Note that ℓ is pointed in both arguments. Let us also define explicitly (once and for all) a pointed version of cong taking a pointed functions $f : A \to_\star B$ to a pointed function $\mathsf{cong}_\star\, f : \Omega\, A \to_\star \Omega\, B$. We define it by

$$\mathsf{cong}_\star\, f\, p := \star_f^{-1} \cdot \mathsf{cong}\; f\; p \cdot \star_f$$

where, recall, $\star_f : f\;\star_A \equiv \star_B$. In other words, $\mathsf{cong}_\star$ is the functorial action of Ω.

We can now add two functions f and g of type $A * B \to_\star C$ by

$(f +^* g)\; :\; A * B \to C$
$(f +^* g)\;(\mathsf{inl}\;a) = \star_C$
$(f +^* g)\;(\mathsf{inr}\;b) = \star_C$
$(f +^* g)\;(\mathsf{push}\;(a\;,\;b)\;i) = (\mathsf{cong}_\star\, f\;(\ell\;(a\;,\;b)) \cdot \mathsf{cong}_\star\, g\;(\ell\;(a\;,\;b)))\;i$

We take this function to be pointed by refl. Note that, since ℓ is pointed in both arguments, both $\mathsf{cong}\;(f +^* g)\;(\mathsf{push}\,(a\,,\star_B))$ and $\mathsf{cong}\;(f +^* g)\;(\mathsf{push}\,(\star_A\,,b))$ vanish. Let us compare this with the addition on the usual definition homotopy groups. In general, we may add any two functions f and g of type $\mathsf{Susp}\,A \to_\star B$ by

$(f +^{\mathsf{Susp}} g)\; :\; \mathsf{Susp}\;A \to B$
$(f +^{\mathsf{Susp}} g)\;\mathsf{north} = \star_B$
$(f +^{\mathsf{Susp}} g)\;\mathsf{south} = \star_B$
$(f +^{\mathsf{Susp}} g)\;(\mathsf{merid}\;a\;i) = (\mathsf{cong}_\star\, f\;(\sigma\;a) \cdot \mathsf{cong}_\star\, g\;(\sigma\;a))\;i$

This is precisely the construction used to define the group structure on π_n whenever $n > 0$. Note that, by construction, we have $\mathsf{cong}\;(f +^{\mathsf{Susp}} g)\;(\mathsf{merid}\;\star_A) \equiv \mathsf{refl}$.

Proposition 6.8. *Given $f, g : \mathbb{S}^{n+m+1} \to_\star A$, we have*

$$(f +^{\mathsf{Susp}} g) \circ \mathsf{F}_{n,m} \equiv (f \circ \mathsf{F}_{n,m}) +^* (g \circ \mathsf{F}_{n,m})$$

Proof. The two functions agree on inl and inr by refl. Let us consider the action on $\mathsf{push}(x\,,y)$. We have

$$\begin{aligned}\mathsf{cong}\;((f +^{\mathsf{Susp}} g) \circ \mathsf{F}_{n,m})\;(\mathsf{push}\,(x\,,y)) &:= \mathsf{cong}\;(f +^{\mathsf{Susp}} g)\;(\sigma(x \smile y)) \\ &\equiv \mathsf{cong}\;(f +^{\mathsf{Susp}} g)\;(\mathsf{merid}\;(x \smile y)) \\ &\cdot \mathsf{cong}\;(f +^{\mathsf{Susp}} g)\;(\mathsf{merid}\;\mathsf{north})^{-1} \\ &\equiv \mathsf{cong}\;(f +^{\mathsf{Susp}} g)\;(\mathsf{merid}\;(x \smile y))\end{aligned}$$

which, by definition, unfolds to

$$\mathsf{cong}_\star\, f\,(\sigma(x \smile y)) \cdot \mathsf{cong}_\star\, g\,(\sigma(x \smile y)) \tag{6.1}$$

On the other hand, $\mathsf{cong}\;((f \circ \mathsf{F}_{n,m}) +^* (g \circ \mathsf{F}_{n,m}))\;(\mathsf{push}\,(x\,,y))$ unfolds to

$$\mathsf{cong}_\star\, f\,(\mathsf{cong}\;\mathsf{F}_{n,m}\;(\ell(x\,,y))) \cdot \mathsf{cong}_\star\, g\,(\mathsf{cong}\;\mathsf{F}_{n,m}\;(\ell(x\,,y))) \tag{6.2}$$

Hence, comparing (6.1) and (6.2), we see that it is enough to show that

$$\mathsf{cong}\;\mathsf{F}_{n,m}\;(\ell(x\,,y)) \equiv \sigma(x \smile y)$$

Unfolding ℓ, we get

$$\begin{aligned}\mathsf{cong}\ \mathsf{F}_{n,m}\ (\ell(x\,,y)) &\equiv \sigma(\star_{\mathbb{S}^n} \smile \star_{\mathbb{S}^m}) \cdot \sigma(x \smile \star_{\mathbb{S}^m})^{-1} \cdot \sigma(x \smile y) \cdot \sigma(\star_{\mathbb{S}^n} \smile y)^{-1} \\ &\equiv \sigma\,\mathsf{north} \cdot \sigma\,\mathsf{north}^{-1} \cdot \sigma(x \smile y) \cdot \sigma\,\mathsf{north}^{-1} \\ &\equiv \sigma(x \smile y)\end{aligned}$$

□

Proposition 6.9. *For any pointed type A, the set $\pi^*_{n+m+1}(A)$ is a group with group structure induced by $+^*$. Furthermore, pre-composition $(\mathsf{F}_{n,m})^* : \pi_{n+m+1}(A) \to \pi^*_{n+m+1}(A)$ is an isomorphism.*

Proof. We know that $(\mathsf{F}_{n,m})^*$ is an equivalence of types. By Proposition 6.8 and the Structure Identity Principle [Uni13, Section 9.8], it induces a path

$$(\pi_{n+m+1}(A), +^{\mathrm{Susp}}) \equiv (\pi^*_{n+m+1}(A), +^*)$$

of raw monoids (i.e. elements of type $\Sigma_{A:\mathsf{Type}}(A \times A \to A)$). Since the the left-hand side of this equality can be extended to form a group, so can the right-hand side. This is precisely what we set out to show. □

The following result follows in exactly the same manner.

Proposition 6.10. *π^*_{n+m+1} is functorial with its action on maps being defined by post-composition.*

6.3. **The new synthetic proof that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$.** Let us now return to the new proof. We will use $\smile$ from above in dimensions $\mathbb{S}^1 \times \mathbb{S}^1 \to \mathbb{S}^2$. We remark that, by Proposition 6.5, it is anti-commutative in these dimensions. In order to make the following constructions somewhat more direct, let us return to the base/loop definition of $\mathbb{S}^1$. Under the equivalence, $\mathsf{Susp}\,\mathsf{Bool} \simeq \mathbb{S}^1$, the multiplication is described by

$$\begin{aligned}&\_\smile\_ : \mathbb{S}^1 \to \mathbb{S}^1 \to \mathbb{S}^2 \\ &\mathsf{base} \smile y = \mathsf{north} \\ &(\mathsf{loop}\ i) \smile y = \sigma\ y\ i\end{aligned}$$

In addition to anti-commutativity and associativity, we have the following distributivity-like fact about $\smile$:

Lemma 6.11. *For $x, y : \mathbb{S}^1$, we have $x \smile (x + y) \equiv x \smile y$*

Proof. We proceed by $\mathbb{S}^1$-induction on x. The equality $\mathsf{base} \smile (\mathsf{base} + y) \equiv \mathsf{base} \smile y$ holds by refl, so we are left to verify the equality

$$\mathsf{cong}\ (x \mapsto x \smile (x + y))\ \mathsf{loop} \equiv \sigma\ y$$

Simplifying the left-hand side using functoriality of binary cong [LM24, Definition 1], we get

$$\begin{aligned}\mathsf{cong}\ (x \mapsto x \smile (x + y))\ \mathsf{loop} \equiv\ &\mathsf{cong}\ (x \mapsto \mathsf{north} \smile (x + y))\ \mathsf{loop} \\ &\cdot \mathsf{cong}\ (x \mapsto x \smile (\mathsf{north} + y))\ \mathsf{loop} \\ :=\ &\mathsf{refl} \cdot \sigma\ y \equiv \sigma\ y\end{aligned}$$

□

We now redefine $\pi_3(\mathbb{S}^2) \cong \mathbb{Z}$ via the following decomposition, primarily defined in terms of post- and pre-composition with $\mathsf{F}_{1,1} : \mathbb{S}^1 * \mathbb{S}^1 \cong \mathbb{S}^3$ and its inverse. In what follows, let us simply write $\mathsf{F} := \mathsf{F}_{1,1}$ and $\pi_3^*(A) := \pi^*_{1+1+1}(A) := \| \mathbb{S}^1 * \mathbb{S}^1 \to_\star A \|_0$. We also remind the reader of the map $\mathsf{h} : \mathbb{S}^1 * \mathbb{S}^1 \to \mathbb{S}^2$ from Definition 3.5 for which h_* is an isomorphism–this follows from Proposition 3.11.

Definition 6.12. Let $\theta : \pi_3(\mathbb{S}^2) \cong \mathbb{Z}$ be defined by the following sequence of isomorphisms

$$\pi_3(\mathbb{S}^2) \xrightarrow{\mathsf{F}^*} \pi_3^*(\mathbb{S}^2) \xrightarrow{(\mathsf{h}_*)^{-1}} \pi_3^*(\mathbb{S}^1 * \mathbb{S}^1) \xrightarrow{\mathsf{F}_*} \pi_3^*(\mathbb{S}^3) \xrightarrow{(\mathsf{F}^{-1})^*} \pi_3(\mathbb{S}^3) \xrightarrow{\xi} \mathbb{Z}$$

where the last map can be chosen to be any reasonable description of the isomorphism $\xi : \pi_3(\mathbb{S}^3) \cong \mathbb{Z}$ sending i_3 to 1.

The goal is to trace the image of $[i_2, i_2] : \pi_3(\mathbb{S}^2)$ under θ. Let us define the following three underlying functions of elements $\eta_1 : \pi_3^*(\mathbb{S}^2)$, $\eta_2 : \pi_3^*(\mathbb{S}^1 * \mathbb{S}^1)$ and $\eta_3 : \pi_3^*(\mathbb{S}^3)$:

```
η₁-fun : S¹ * S¹ → S²
η₁-fun (inl x) = north
η₁-fun (inr y) = north
η₁-fun (push (x , y) i) = (σ y ∙ σ x) i

η₂-fun : S¹ * S¹ → S¹ * S¹
η₂-fun (inl x) = inr (- x)
η₂-fun (inr y) = inr y
η₂-fun (push (x , y) i) = (push (y - x , - x) ⁻¹ ∙ push (y - x , y)) i

η₃-fun : S¹ * S¹ → S³
η₃-fun (inl x) = north
η₃-fun (inr y) = north
η₃-fun (push (x , y) i) = (σ (x ⌣ y) ⁻¹ ∙ σ (x ⌣ y) ⁻¹) i
```

The claim is now that the image of $[i_2, i_2]$ under the chain of isomorphisms can be described as follows:

$$[i_2, i_2] \xmapsto{\mathsf{F}^*} \eta_1 \xmapsto{(\mathsf{h}_*)^{-1}} \eta_2 \xrightarrow{\mathsf{F}_*} \eta_3 \xmapsto{(\mathsf{F}^{-1})^*} (-2)i_3 \xmapsto{\xi} \pm 2$$

Lemma 6.13. $\mathsf{F}^*\,[i_2, i_2] \equiv \eta_1$

Proof. The definition of η_1 matches that of $|\nabla \circ \mathsf{W}| : \pi_3^*(\mathbb{S}^2)$, and so the statement holds by construction of the Whitehead product. □

Lemma 6.14. $(\mathsf{h}_*)^{-1}\,\eta_1 \equiv \eta_2$

Proof. Applying h_* on both sides gives the equation $\eta_1 \equiv \mathsf{h}_*\,\eta_2$. Thus, we are done if we can show that $\eta_1\text{-fun}\,a \equiv \mathsf{h}\,(\eta_2\text{-fun}\,a)$ for $a : \mathbb{S}^1 * \mathbb{S}^1$. We do it by induction on a. When a is $\mathsf{inl}\,x$ or $\mathsf{inr}\,y$, the equality holds by refl. Thus, it remains to show that

$$\mathsf{cong}\ \eta_1\text{-fun}\ (\mathsf{push}\,(x\,,y)) \equiv \mathsf{cong}\ (\mathsf{h} \circ \eta_2\text{-fun})\ (\mathsf{push}\,(x\,,y))$$

We show the identity by unfolding the right-hand side:

$$\begin{aligned}
\mathsf{cong}\ (\mathsf{h} \circ \eta_2\text{-}\mathsf{fun})\ (\mathsf{push}\ (x\ , y)) &:\equiv \mathsf{cong}\ \mathsf{h}\ (\mathsf{push}\ (y - x\ , -x))^{-1} \cdot \mathsf{push}\ (y - x\ , y)) \\
&\equiv \mathsf{cong}\ \mathsf{h}\ (\mathsf{push}\ (y - x\ , -x))^{-1} \cdot \mathsf{cong}\ \mathsf{h}\ (\mathsf{push}\ (y - x\ , y)) \\
&:\equiv \sigma((-x) - (y - x))^{-1} \cdot \sigma(y - (y - x)) \\
&\equiv \sigma(-y)^{-1} \cdot \sigma\, x \\
&\equiv \sigma\, y \cdot \sigma\, x \\
&=: \mathsf{cong}\ \eta_1\text{-}\mathsf{fun}\ (\mathsf{push}\ (x\ , y))
\end{aligned}$$

□

Lemma 6.15. $\mathsf{F}_*\, \eta_2 \equiv \eta_3$

Proof. The identity follows if we can show that $\mathsf{F}(\eta_2\text{-}\mathsf{fun}\, a) \equiv \eta_3\text{-}\mathsf{fun}\, a$ for $a : \mathbb{S}^1 * \mathbb{S}^1$. Again, the identity holds by refl when a is $\mathsf{inl}\, x$ or $\mathsf{inr}\, y$. So it remains to show that

$$\mathsf{cong}\ (\mathsf{F} \circ \eta_2\text{-}\mathsf{fun})\ (\mathsf{push}\ (x\ , y)) \equiv \mathsf{cong}\ \eta_3\text{-}\mathsf{fun}\ (\mathsf{push}\ (x\ , y))$$

Just like in the proof of Lemma 6.15, we show this simply by unfolding the definitions of the, in this case, left-hand side. We get:

$$\begin{aligned}
\mathsf{cong}\ (\mathsf{F} \circ \eta_2\text{-}\mathsf{fun})\ (\mathsf{push}\ (x\ , y)) &:\equiv \mathsf{cong}\ \mathsf{F}\ (\mathsf{push}\ (y - x\ , -x))^{-1} \cdot \mathsf{push}\ (y - x\ , y)) \\
&\equiv \mathsf{cong}\ \mathsf{F}\ (\mathsf{push}\ (y - x\ , -x))^{-1} \cdot \mathsf{cong}\ \mathsf{F}\ (\mathsf{push}\ (y - x\ , y)) \\
&:\equiv \sigma((y - x) \frown (-x))^{-1} \cdot \sigma((y - x) \frown y) \\
&\equiv \sigma((-x) \frown ((-x) + y)) \cdot \sigma(y \frown (y - x))^{-1} \\
&\equiv \sigma((-x) \frown y) \cdot \sigma(y \frown (-x))^{-1} \\
&\equiv \sigma(x \frown y)^{-1} \cdot \sigma(y \frown (-x))^{-1} \\
&\equiv \sigma(x \frown y)^{-1} \cdot \sigma(x \frown y)^{-1} \\
&=: \mathsf{cong}\ \eta_3\text{-}\mathsf{fun}\ (\mathsf{push}\ (x\ , y))
\end{aligned}$$

where the fourth and seventh equalities come from anti-commutativity and the fifth equality from Lemma 6.11. The fact that σ commutes with inversion is used throughout. □

Theorem 6.16. $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$

Proof. By uniqueness (up to a sign) of isomorphisms $\pi_3(\mathbb{S}^2) \cong \mathbb{Z}$, it suffices, according to Corollary 4.12, to show that the image of $[i_2, i_2]$ under θ is ± 2. That is:

$$(\xi \circ (\mathsf{F}^{-1})^* \circ \mathsf{F}_* \circ (\mathsf{h}_*)^{-1} \circ \mathsf{F}^*)[i_2, i_2] \equiv \pm 2$$

By Lemma 6.13, Lemma 6.14 and Lemma 6.15, it suffices to show that

$$(\xi \circ (\mathsf{F}^{-1})^*)\, \eta_3 \equiv \pm 2$$

One can easily show that $\mathsf{F}^{-1}\, \eta_3 \equiv (-2)\, i_3$, and hence

$$(\xi \circ (\mathsf{F}^{-1})^*)\, \eta_3 \equiv (-2)\, (\xi\, i_3) \equiv -2$$

□

In addition to providing a much shorter proof of $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$, this gives us a sequence of new Brunerie numbers, $\beta_1, \beta_2, \beta_3 : \mathbb{Z}$, of decreasing complexity:

$$\begin{aligned}
\beta_1 &= (\xi \circ (\mathsf{F}^{-1})^* \circ \mathsf{F}_* \circ (\mathsf{h}_*)^{-1})\, \eta_1 \\
\beta_2 &= (\xi \circ (\mathsf{F}^{-1})^* \circ \mathsf{F}_*)\, \eta_2 \\
\beta_3 &= (\xi \circ (\mathsf{F}^{-1})^*)\, \eta_3
\end{aligned}$$

This gives new hope for Brunerie's conjecture about a proof by normalisation. This may be captured as follows:

Theorem 6.17 (New Brunerie numbers)**.** *If either of* $\beta_1, \beta_2, \beta_3 : \mathbb{Z}$ *normalises to* ± 2*, then* $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$.

Ideally, we could normalise β_1. This, however, turns out to be difficult, as it does not bypass the main hurdle of computing the inverse of the isomorphism $\pi_3^*(\mathbb{S}^2) \cong \pi_3^*(\mathbb{S}^1 * \mathbb{S}^1)$ induced by the Hopf map, which has a rather indirect construction coming from the LES of homotopy groups associated to the Hopf fibration. This problem does not apply to β_2, for which the computation does not rely on the problematic inverse. Unfortunately, also β_2 fails to normalise in reasonable time in `Cubical Agda`. This is surprising, as the only maps playing a fundamental role here are two applications of the equivalence $\mathbb{S}^1 * \mathbb{S}^1 \simeq \mathbb{S}^3$, which is not too involved, and one application of ξ which may be compactly described via

$$\pi_3(\mathbb{S}^3) \xrightarrow{|-|_*} \mathsf{H}^3(\mathbb{S}^3) \xrightarrow{\cong} \mathbb{Z}$$

and computes relatively well if the last isomorphism is constructed as in [BLM22].[5] We have hence, at the time of writing, not been able to normalise even β_2, despite many optimisations of the functions involved. We are, however, able to normalise β_3 after some minor modifications to η_3 and the map $\pi_3^*(\mathbb{S}^3) \to \mathbb{Z}$. This optimised version of β_3, normalises to -2 in `Cubical Agda` in just under 4 seconds, thereby giving us an at least partially computer-assisted proof of $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$.

We emphasise again that β_2 is a vastly simplified version of β since the isomorphism $\pi_3(\mathbb{S}^2) \cong \pi_3(\mathbb{S}^3)$ never has to be computed. Hence, it is rather surprising that computations break down already at this stage. This tells us that `Cubical Agda` has a long way to go before any direct computation of the original β is feasible. We hope that this could be useful for benchmarking in future optimisations of `Cubical Agda` and related systems.

Finally, we address the elephant in the room: why is there a minus sign popping up? In other words, have we really chosen the, in some way, canonical isomorphism? The isomorphism $\pi_3(\mathbb{S}^3) \cong \mathbb{Z}$ maps, as expected, i_3 to 1, so it can hardly be the culprit. Neither can the equivalence $\mathsf{F} : \mathbb{S}^1 * \mathbb{S}^1 \simeq \mathbb{S}^3$, since it is applied equally in the constructions of `hopf` and of $[i_2, i_2]$. We could, however, have defined the `push`-case for `h` by

```
h (push (x , y) i) = σ (x - y) i
```

in which case θ would have sent $[i_2, i_2]$ to 2 and `hopf` to 1 (note that this is only possible since altering `h` would alter the definition of θ). The construction of `h` that we have given is, however, precisely the one which fell out by unfolding our formalisation of Brunerie's construction of the corresponding map. If this indeed is what Brunerie intended, we may also conclude that the original Brunerie number β is equal to -2. We stress that this merely is a fun fact and of no mathematical importance to Brunerie's proof or our formalisation.

6.4. **A stand-alone proof of Brunerie's theorem?** We saw above that the new proof of $\beta \equiv \pm 2$ together with Corollary 4.12 implies Brunerie's theorem. However, what conclusions can we draw concerning the cardinality of $\pi_4(\mathbb{S}^3)$ in the absence of Corollary 4.12? In other words, how self-contained is the new proof? While the fact that $\beta \equiv \pm 2$ does not

[5] As noted in [Lju20], the Freudenthal suspension theorem should be avoided here as it has a tendency to lead to very slow computations. This is another way in which we deviate from Brunerie's β.

automatically imply that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$, it does provide all ingredients necessary for a stand-alone proof of the following fact:

Theorem 6.18. *If* $\pi_4(\mathbb{S}^3) \not\simeq \mathbb{1}$*, then* $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$.

Before we prove Theorem 6.18, we need to analyse the action of suspension on Whitehead products and, in particular, on $[i_2, i_2] : \pi_3(\mathbb{S}^2)$. In what follows, let A, B and C be pointed types and let us fix two pointed functions $f : \mathsf{Susp}\, A \to_\star C$ and $g : \mathsf{Susp}\, B \to_\star C$. The Whitehead product of f and g can be understood as the composition

$$A * B \xrightarrow{\mathsf{W}} \mathsf{Susp}\, A \vee \mathsf{Susp}\, B \xrightarrow{f \vee g} C \vee C \xrightarrow{\nabla} C$$

We remark that this construction has been independently studied by Cagne et al. [CBKB24, Definition 6.3] who call it the 'generalised Whitehead product'. After a bit of massaging, this function can be given a very simple description:

$$\begin{aligned}
&(f \cdot_w g) : A * B \to C \\
&(f \cdot_w g)\ (\mathsf{inl}\ x) = \star_C \\
&(f \cdot_w g)\ (\mathsf{inr}\ y) = \star_C \\
&(f \cdot_w g)\ (\mathsf{push}\ (x\ ,\ y)\ i) = (\mathsf{cong}_\star\, g\ (\sigma\ y) \cdot \mathsf{cong}_\star\, f\ (\sigma\ x))\ i
\end{aligned}$$

We remark that this composition gives η_1-fun when $A = B = \mathbb{S}^1$, $C = \mathbb{S}^2$ and $f = g = \mathsf{id}_{\mathbb{S}^2}$.

Our aim is to show that $f \cdot_w g$ vanishes under suspension. To this end, let us consider a function very similar to $f \cdot_w g$:

$$\begin{aligned}
&\gamma : A * B \to C \\
&\gamma\ (\mathsf{inl}\ x) = \star_C \\
&\gamma\ (\mathsf{inr}\ y) = \star_C \\
&\gamma\ (\mathsf{push}\ (x\ ,\ y)\ i) = (\mathsf{cong}_\star\, f\ (\sigma\ x) \cdot \mathsf{cong}_\star\, g\ (\sigma\ y))\ i
\end{aligned}$$

Despite the similarity of $f \cdot_w g$ and γ, the latter turns out to be trivial.

Lemma 6.19. γ *is constant.*

Proof. We show that $\gamma\, a \equiv \star_C$ for all $a : \mathbb{S}^1 * \mathbb{S}^1$ by induction on a. When a is $\mathsf{inl}\, x$, the left-hand side reduces to $\star_C$, so we need to provide a path $\star_C \equiv \star_C$. Instead of choosing the obvious path refl, we provide $\mathsf{cong}_\star\, f\, (\sigma\, x) : \star_C \equiv \star_C$. When a is $\mathsf{inr}\, y$, we have the same goal. This time, we provide the path $\mathsf{cong}_\star\, g\, (\sigma\, y)^{-1}$. For the final step, i.e. the action of γ on $\mathsf{push}\, (x\, , y)$, we need to provide a filler of the following square of paths:

$$\begin{array}{ccc}
\star_C & \xrightarrow{\quad\mathsf{refl}\quad} & \star_C \\
{\scriptstyle \mathsf{cong}_\star\, f\, (\sigma\, x)}\uparrow & & \uparrow{\scriptstyle (\mathsf{cong}_\star\, g\, (\sigma\, y))^{-1}} \\
\star_C & \xrightarrow[\mathsf{cong}_\star\, f\, (\sigma\, x) \cdot \mathsf{cong}_\star\, g\, (\sigma\, y)]{} & \star_C
\end{array}$$

Squares of this shape always have a filler by definition of path composition, and thus the statement holds. □

Now, although $f \cdot_w g$ and γ may look similar, it it is now, in light of Lemma 6.19, clear that they are not the same. This happens because the actions of the functions on $\mathsf{push}\, (x\, , y)$ only are the same up to commutation of paths—something which is not always legal in the possibly non-commutative loop space $\Omega\, C$. Nevertheless, after suspending the function, the situation is different:

Lemma 6.20. *The pointed functions* $\mathsf{Susp}\,(f \cdot_w g), \mathsf{Susp}\,\gamma : \mathsf{Susp}\,(A * B) \to_\star \mathsf{Susp}\,C$ *are equal.*

Proof. Under the adjunction $\mathsf{Susp} \dashv \Omega$, it is enough to show that for every $a : A * B$, we have an equality of loops in $\Omega\,(\mathsf{Susp}\,C)$:

$$\sigma((f \cdot_w g)\,a) \equiv \sigma(\gamma\,a)$$

We proceed by induction on a. When a is $\mathsf{inl}\,x$ or $\mathsf{inr}\,y$, the equality holds by refl. Thus, it remains to show that

$$\mathsf{cong}\;\sigma\;(\mathsf{cong}\;(f \cdot_w g)\;(\mathsf{push}\,(x\,,y))) \equiv \mathsf{cong}\;\sigma\;(\mathsf{cong}\;\gamma\;(\mathsf{push}\,(x\,,y)))$$

As before, this is a simple exercise in unfolding the definitions of each respective function:

$$\begin{aligned}
\mathsf{cong}\;\sigma\;(\mathsf{cong}\;(f \cdot_w g)\;(\mathsf{push}\,(x\,,y))) &:\equiv \mathsf{cong}\;\sigma\;(\mathsf{cong}_\star\,g\,(\sigma\,y) \cdot \mathsf{cong}_\star\,f\,(\sigma\,x)) \\
&\equiv \mathsf{cong}\;\sigma\;(\mathsf{cong}_\star\,g\,(\sigma\,y)) \cdot \mathsf{cong}\;\sigma\;(\mathsf{cong}_\star\,f\,(\sigma\,x)) \\
&\equiv \mathsf{cong}\;\sigma\;(\mathsf{cong}_\star\,f\,(\sigma\,x)) \cdot \mathsf{cong}\;\sigma\;(\mathsf{cong}_\star\,g\,(\sigma\,y)) \quad \text{(EH)} \\
&\equiv \mathsf{cong}\;\sigma\;(\mathsf{cong}_\star\,f\,(\sigma\,x) \cdot \mathsf{cong}_\star\,g\,(\sigma\,y)) \\
&\equiv \mathsf{cong}\;\gamma\;(\mathsf{push}\,(x\,,y))
\end{aligned}$$

where the step labelled (EH) is an application of the Eckmann-Hilton argument which says that path composition in $\Omega^2\,A$ is commutative for any pointed type A [Uni13, Theorem 2.1.6]. In particular, since we may interpret $\mathsf{cong}\;\sigma\;(\mathsf{cong}_\star\,f\,(\sigma\,x))$ and $\mathsf{cong}\;\sigma\;(\mathsf{cong}_\star\,g\,(\sigma\,y))$ as loops in $\Omega^2\,(\mathsf{Susp}\,C)$, the identity holds. □

Proposition 6.21. *The pointed function* $\mathsf{Susp}\,(f \cdot_w g) : \mathsf{Susp}\,(A * B) \to_\star \mathsf{Susp}\,C$ *is constant.*

Proof. Since γ is constant and constant functions are preserved by suspension, Lemma 6.19 gives us the desired equality of (pointed) functions:

$$\mathsf{Susp}\,(f \cdot_w g) \equiv \mathsf{Susp}\,\gamma \equiv \mathsf{const}$$ □

As we have seen before, setting $A = \mathbb{S}^n$ and $B = \mathbb{S}^m$ in the definition of $f \cdot_w g$, so that $f : \mathbb{S}^{n+1} \to_\star C$ and $g : \mathbb{S}^{m+1} \to_\star C$ we obtain the usual Whitehead product. $[|\,f\,|, |\,g\,|] : \pi_{n+m+1}(C)$, that is

$$[|\,f\,|, |\,g\,|] \equiv |\,(f \cdot_w g) \circ \mathsf{F}^{-1}_{n,m}\,|$$

Let us translate Proposition 6.21 to a result concerning these maps.

Proposition 6.22. *For* $f : \mathbb{S}^n \to_\star C$ *and* $g : \mathbb{S}^m \to_\star C$*, their Whitehead product* $(f \cdot_w g) \circ \mathsf{F}^{-1}_{n,m}$ *vanishes under suspension, i.e.*

$$\mathsf{Susp}\,(|\,(f \cdot_w g) \circ \mathsf{F}^{-1}_{n,m}\,|) \equiv \mathsf{const}$$

Proof. The result follows immediately from the fact that the action of suspension $\mathsf{Susp} : (X \to_\star Y) \to (\mathsf{Susp}\,X \to_\star \mathsf{Susp}\,Y)$ is functorial and from Proposition 6.21. □

We get the following classically well-known theorem as an immediate corollary:

Theorem 6.23. *For any* $x : \pi_{n+1}(C)$ *and* $y : \pi_{m+1}(C)$*, the Whitehead product* $[x, y] : \pi_{n+m+1}(C)$ *lies in the kernel of the suspension map* $\sigma_* : \pi_{n+m+1}(C) \to \pi_{n+m+2}(\mathsf{Susp}\,C)$

We now have all that we need in order to prove Theorem 6.18.

Proof of Theorem 6.18. By the Freudenthal suspension theorem, we know that $\sigma_* : \pi_3(\mathbb{S}^2) \to \pi_4(\mathbb{S}^3)$ is surjective. Furthermore, we know that the domain of this function is isomorphic to $\mathbb{Z}$ via θ from Definition 6.12 and thus we have a surjection $\sigma_* \circ \theta^{-1} : \mathbb{Z} \to \pi_4(\mathbb{S}^3)$. We know from the new direct calculation of the Brunerie number that $\theta^{-1}(-2) \equiv [i_2, i_2]$ and thus we have

$$\sigma_*(\theta^{-1}(-2)) \equiv \sigma_*[i_2, i_2] \equiv 0_{\pi_4(\mathbb{S}^3)}$$

where the second equality comes from Theorem 6.23. Hence, we have shown that there exists a surjection from $\mathbb{Z}$ onto $\pi_4(\mathbb{S}^3)$ with -2 in its kernel. This implies the theorem. □

Now, with Theorem 6.18 in mind, we seem to be very close to having produced a remarkably short proof of Brunerie's theorem. All that remains is showing that $\pi_4(\mathbb{S}^3)$ is not trivial. This, however, turns out not to be entirely straightforward. One possible proof uses the so called *Steenrod Squares*. This is a cohomology operation which was originally defined in HoTT by Brunerie [Bru16b] and whose theory was recently made available in HoTT by Ljungström and Wärn [LW24]. Such an approach, however, can hardly be said to simplify Brunerie's original proof, as the Steenrod Squares are rather advanced constructions. A solution to this problem which would truly be impressive would be a direct construction of two elements $x, y : \pi_4(\mathbb{S}^3)$ and a proof that $x \not\equiv y$. While this appears to be difficult to do by hand, we can, since we are working constructively, reformulate this problem as a computational challenge.

Challenge. *Construct a function* $f : \pi_4(\mathbb{S}^3) \to \mathsf{Bool}$ *and an element* $e : \pi_4(\mathbb{S}^3)$ *such that*

- $f\, 0_{\pi_4(\mathbb{S}^3)}$ *computes to* true *and*
- $f\, e$ *computes to* false.

In fact, such a computation was successfully run by Jack [Jac23] in `cubicaltt` [CCHM]. Unfortunately, `Cubical Agda` has not yet been able to perform the computation.

7. Conclusion

In this paper, we have presented three formalisations of $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$ in the `Cubical Agda` system. For the different proofs that $|\beta| \equiv 2$, the line count is roughly as follows:

(1) Brunerie's original proof [$\sim 9,000$ LOC]
(2) A direct calculation of β [~ 600 LOC]
(3) A computer-assisted reformulation of (2) [~ 400 LOC]

As always, the number of lines of code (LOC) should be taken with a grain of salt. First, the $9,000$ LOC in the first formalisation exclude over $8,000$ LOC from [Kan22b, Cav20, BLM22] which we have imported as libraries. In addition, these numbers also exclude many elementary results used in the formalisation, including ~ 9000 LOC for Chapters 1–3. We also stress that the line count for formalisations (2) and (3) only concern the part of the proof discussed in section 6.

Formalisation (1), which constituted the bulk of this paper, was a formalisation of Brunerie's pen-and-paper proof, taking some convenient shortcuts when possible. The problem of formalising Brunerie's proof has been a widely discussed open problem in HoTT/UF, and we hope that our efforts here provide a satisfactory solution to it. Formalisations (2) and (3) were of a simplified calculation of the Brunerie number, β. The very similar proofs (2) and (3) differ in that (3) uses `Cubical Agda` to carry out part of the computation of the new Brunerie number automatically. Perhaps equally important, we have seen that (3)

provides us with new Brunerie numbers $\beta_1, \beta_2 : \mathbb{Z}$ which are far simpler than the original one, but still do not normalise in a reasonable amount of time. Our hope is that these can prove useful in future optimisations of `Cubical Agda` and related systems, as they could help shed some light on where the normalisation of the original Brunerie number breaks down.

We remark that proofs (1) and (2) could be done in Book HoTT and do not use any cubical machinery in a fundamental way, making them interpretable in any suitably structured $(\infty, 1)$-topos [Shu19]. We hence claim that, in our formalisations, we do not crucially rely on computations using univalence and HITs to prove anything that we could not have proved by hand in Book HoTT. Nevertheless, the `Cubical Agda` system has been very helpful in the formalisation, primarily due to its native support for HITs and definitional computation rules for higher constructors. Formalisation (3), however, is only valid in a system with computational support for univalence as it crucially relies on normalisation of proof terms involving univalence. It would be interesting to run this in other cubical systems, like `cubicaltt` [CCHM], `redtt` [Redb], `cooltt` [Reda], etc.

In addition to the above, we have also taken the opportunity to include some important constructions and results concerning joins of spheres and Whitehead products. In particular, we have given a very explicit definition of the decomposition of spheres into joins of spheres, given a new construction of homotopy groups in terms of maps out of joins of spheres and shown that Whitehead products vanish under suspension. The vanishing of Whitehead products allowed us to extend (2) to a stand-alone proof of the fact that $\pi_4(\mathbb{S}^3)$ is either trivial or isomorphic to $\mathbb{Z}/2\mathbb{Z}$. Interestingly, another direct proof using an entirely different approach of this very fact was recently announced by Baker [Bak24]. Baker's argument is concerned with showing that a certain path constructed via the Eckmann-Hilton argument generates $\pi_3(\mathbb{S}^2)$ and then concludes that two times this generator must vanish under suspension due to the so called *syllepsis* [SK22]. We leave it to future work to investigate if anything interesting can be said about the relation between Baker's proof and ours.

We also remark that our formalisation of Brunerie's proof does not cover all results of Brunerie's thesis in full generality. For instance, we have not developed his proof concerning Whitehead products in full generality. We leave this generalisation for future work. This would tie in nicely with another possible direction of future research, namely that of investigating whether the approach outlined in section 6 can be used to compute other Whitehead products. In addition, describing their graded quasi-Lie algebra structure is work in progress.

Acknowledgement

First and foremost we would like to thank Guillaume Brunerie for his excellent thesis, the conjecture about the computability of β, and for the many discussions about this over the years. We would also like to thank Thierry Coquand and Simon Huber for the first attempt to compute the number together with Guillaume using `cubical` in December 2014 and everyone else who has tried to compute the number and contributed ideas to this since. The `Cubical Agda` formalisation relies on many contributions to `agda/cubical` by more people than we can mention, but we are especially grateful to Evan Cavallo for the Freudenthal suspension theorem as well as many cool cubical tricks and to Rongji Kang for the Blakers-Massey theorem.

References

[Agd24] The Agda Development Team. The Agda Programming Language, 2024. URL: http://wiki.portal.chalmers.se/agda/.

[AW09] Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 146(1):45–55, January 2009. doi:10.1017/S0305004108001783.

[Bak24] Raymond Baker. Eckmann-Hilton and the Hopf Fibration. Extended abstract at *Workshop on Homotopy Type Theory / Univalent Foundations (HoTT/UF24*, 2024. URL: https://hott-uf.github.io/2024/abstracts/HoTTUF_2024_paper_24.pdf.

[BHF18] Ulrik Buchholtz and Kuen-Bang Hou Favonia. Cellular Cohomology in Homotopy Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 521–529, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3209108.3209188.

[BLM22] Guillaume Brunerie, Axel Ljungström, and Anders Mörtberg. Synthetic Integral Cohomology in Cubical Agda. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*, volume 216 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2022/15731, doi:10.4230/LIPIcs.CSL.2022.11.

[Bru16a] Guillaume Brunerie. *On the homotopy groups of spheres in homotopy type theory*. PhD thesis, Université Nice Sophia Antipolis, 2016. URL: http://arxiv.org/abs/1606.05916.

[Bru16b] Guillaume Brunerie. The steenrod squares in homotopy type theory. Abstract at *23rd International Conference on Types for Proofs and Programs (TYPES 2017)*, 2016. URL: https://types2017.elte.hu/proc.pdf#page=45.

[Bru18] Guillaume Brunerie. Computer-generated proofs for the monoidal structure of the smash product. *Homotopy Type Theory Electronic Seminar Talks*, November 2018. URL: https://www.uwo.ca/math/faculty/kapulkin/seminars/hottest.html.

[Bru19] Guillaume Brunerie. The James Construction and $\pi_4(\mathrm{S}^3)$ in Homotopy Type Theory. *Journal of Automated Reasoning*, 63:255–284, 2019.

[Cav20] Evan Cavallo. Formalisation of the Freudenthal Suspension Theorem, 2020. URL: https://github.com/agda/cubical/blob/master/Cubical/Homotopy/Freudenthal.agda.

[CBKB24] Pierre Cagne, Ulrik Buchholtz, Nicolai Kraus, and Marc Bezem. On symmetries of spheres in univalent foundations, 2024. arXiv:2401.15037.

[CCHM] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. CUBICALTT: Cubical Type Theory. Implementation available at https://github.com/mortberg/cubicaltt.

[CCHM18] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In Tarmo Uustalu, editor, *21st International Conference on Types for Proofs and Programs (TYPES 2015)*, volume 69 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:34, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TYPES.2015.5.

[CH19] Evan Cavallo and Robert Harper. Higher Inductive Types in Cubical Computational Type Theory. *Proceedings of the ACM on Programming Languages*, 3(POPL):1:1–1:27, January 2019. doi:10.1145/3290314.

[CHM18] Thierry Coquand, Simon Huber, and Anders Mörtberg. On Higher Inductive Types in Cubical Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 255–264. ACM, 2018. doi:10.1145/3209108.3209197.

[CS20] J. Daniel Christensen and Luis Scoccola. The Hurewicz theorem in Homotopy Type Theory, 2020. Preprint. URL: https://arxiv.org/abs/2007.05833, arXiv:2007.05833.

[Hat02] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002. URL: https://pi.math.cornell.edu/~hatcher/AT/AT.pdf.

[HFLL16] Kuen-Bang Hou (Favonia), Eric Finster, Daniel R. Licata, and Peter LeFanu Lumsdaine. A Mechanization of the Blakers-Massey Connectivity Theorem in Homotopy Type Theory. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '16, pages 565–574, New York, NY, USA, 2016. ACM. doi:10.1145/2933575.2934545.

[Jac23] Tom Jack. $\pi_4\mathbb{S}^3 \ncong 1$ and another Brunerie number in CCHM. Extended abstract at *The Second International Conference on Homotopy Type Theory (HoTT 2023)*, 2023. URL: https://hott.github.io/HoTT-2023/abstracts/HoTT-2023_abstract_21.pdf.

[Jam55] I. M. James. Reduced product spaces. *Annals of Mathematics*, 62(1):170 – 197, 1955.

[Kan22a] Rongji Kang. Formalisation of the James Construction, 2022. URL: https://github.com/agda/cubical/tree/master/Cubical/HITs/James.

[Kan22b] Rongji Kang. Formalisation of the James Construction, 2022. URL: https://github.com/agda/cubical/tree/master/Cubical/HITs/James.

[Lic14] Daniel R. Licata. Another proof that univalence implies function extensionality, 2014. Blog post at https://homotopytypetheory.org/2014/02/17/another-proof-that-univalence-implies-function-extensionality/.

[Lju20] Axel Ljungström. Computing Cohomology in Cubical Agda. Master's thesis, Stockholm University, 2020.

[Lju22] Axel Ljungström. The Brunerie Number Is -2, 2022. Blog post at https://homotopytypetheory.org/2022/06/09/the-brunerie-number-is-2/.

[Lju24] Axel Ljungström. Symmetric Monoidal Smash Products in Homotopy Type Theory, 2024. arXiv:2402.03523.

[LLM23] Thomas Lamiaux, Axel Ljungström, and Anders Mörtberg. Computing cohomology rings in cubical agda. In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2023, page 239–252, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3573105.3575677.

[LM23] Axel Ljungström and Anders Mörtberg. Formalizing $\pi 4(\mathrm{S}^3) \cong \mathrm{Z}/2\mathrm{Z}$ and Computing a Brunerie Number in Cubical Agda. In *LICS*, pages 1–13, 2023. doi:10.1109/LICS56636.2023.10175833.

[LM24] Axel Ljungström and Anders Mörtberg. Computational Synthetic Cohomology Theory in Homotopy Type Theory, 2024. arXiv:2401.16336.

[LS13] Daniel R. Licata and Michael Shulman. Calculating the Fundamental Group of the Circle in Homotopy Type Theory. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '13, pages 223–232, Washington, DC, USA, 2013. IEEE Computer Society. doi:10.1109/LICS.2013.28.

[LS20] Peter LeFanu Lumsdaine and Michael Shulman. Semantics of higher inductive types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 169(1):159–208, 2020. doi:10.1017/S030500411900015X.

[LW24] Axel Ljungström and David Wärn. The Steenrod Squares in HoTT Revisited. Extended abstract at *Workshop on Homotopy Type Theory / Univalent Foundations (HoTT/UF24*, 2024. URL: https://hott-uf.github.io/2024/abstracts/HoTTUF_2024_paper_8.pdf.

[ML75] Per Martin-Löf. An Intuitionistic Theory of Types: Predicative Part. In H. E. Rose and J. C. Shepherdson, editors, *Logic Colloquium '73*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 73–118. North-Holland, 1975. doi:10.1016/S0049-237X(08)71945-1.

[ML84] Per Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in Proof Theory*. Bibliopolis, 1984.

[MP20] Anders Mörtberg and Loïc Pujet. Cubical Synthetic Homotopy Theory. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, pages 158–171, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3372885.3373825.

[Reda] RedPRL Development Team. cooltt. https://www.github.com/RedPRL/cooltt.

[Redb] RedPRL Development Team. redtt. https://www.github.com/RedPRL/redtt.

[Shu19] Michael Shulman. All $(\infty, 1)$-toposes have strict univalent universes, April 2019. Preprint. URL: https://arxiv.org/abs/1904.07004, arXiv:1904.07004.

[SK22] Kristina Sojakova and G. A. Kavvos. Syllepsis in Homotopy Type Theory. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '22, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531130.3533347.

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Self-published, Institute for Advanced Study, 2013. URL: https://homotopytypetheory.org/book/.

[VMA21] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types. *Journal of Functional Programming*, 31:e8, 2021. doi:10.1017/S0956796821000034.

[Voe10a] Vladimir Voevodsky. The equivalence axiom and univalent models of type theory, February 2010. Notes from a talk at Carnegie Mellon University. URL: http://www.math.ias.edu/vladimir/files/CMU_talk.pdf.

[Voe10b] Vladimir Voevodsky. Univalent foundations, September 2010. Notes from a talk in Bonn. URL: https://www.math.ias.edu/vladimir/sites/math.ias.edu.vladimir/files/Bonn_talk.pdf.

[Wär23] David Wärn. Eilenberg–maclane spaces and stabilisation in homotopy type theory. *Journal of Homotopy and Related Structures*, 18(2):357–368, Sep 2023. doi:10.1007/s40062-023-00330-5.

Paper III

Mathematical Structures in Computer Science (2024), 1–23
doi:10.1017/S0960129524000318

CAMBRIDGE UNIVERSITY PRESS

PAPER

Symmetric monoidal smash products in homotopy type theory

Axel Ljungström

Department of Mathematics, Stockholm University, Stockholm, Sweden
Email: axel.ljungstrom@math.su.se

(Received 31 January 2024; revised 7 August 2024; accepted 4 September 2024)

Abstract

In homotopy type theory, few constructions have proved as troublesome as the smash product. While its definition is just as direct as in classical mathematics, one quickly realises that in order to define and reason about functions over iterations of it, one has to verify an exponentially growing number of coherences. This has led to crucial results concerning smash products remaining open. One particularly important such result is the fact that the smash product forms a (1-coherent) symmetric monoidal product on the universe of pointed types. This fact was used, without a complete proof, by, for example, Brunerie ((2016) PhD thesis, Université Nice Sophia Antipolis) to construct the cup product on integral cohomology and is, more generally, a fundamental result in traditional algebraic topology. In this paper, we salvage the situation by introducing a simple informal heuristic for reasoning about functions defined over iterated smash products. We then use the heuristic to verify, for example, the hexagon and pentagon identities, thereby obtaining a proof of symmetric monoidality. We also provide a formal statement of the heuristic in terms of an induction principle concerning the construction of homotopies of functions defined over iterated smash products. The key results presented here have been formalised in the proof assistant Cubical Agda.

Keywords: Smash products; synthetic homotopy theory; symmetric monoidal categories; homotopy type theory; univalent foundations; constructive mathematics

1. Introduction

In his 2016 proof of $\pi_4(S^3) \cong \mathbb{Z}/2\mathbb{Z}$ in homotopy type theory (HoTT), Brunerie (2016) crucially uses – but never proves in detail – that the smash product is (1-coherent) symmetric monoidal. Due to the vast amount of path algebra involved when reasoning about smash products in HoTT, this has since remained open. While it turns out that smash products are not needed for Brunerie's proof (Ljungström and Mörtberg 2023), the problem is still interesting in its own right.

Several attempts have been made at salvaging the situation. van Doorn (2018) came very close by considering an argument using closed monoidal categories but left a gap where the path algebra became too technical. To be more precise, Van Doorn never verified that the equivalence

$$(A \wedge B \to_\star C) \simeq (A \to_\star (B \to_\star C)) \tag{1}$$

is a pointed natural equivalence. Another line of attack by Cavallo and Harper (Cavallo and Harper 2020; Cavallo 2021a) is the addition of parametricity to the type theory, which leads to a rather ingenious proof of the theorem. This, of course, happens at the expense of making the type theory

more complicated. Yet another solution was studied by Brunerie (2018) who used Agda metaprogramming to generate the relevant proofs. Possible philosophical objections to such a solution aside, Brunerie's generated proof of the pentagon identity failed to type-check due to its high memory consumption.

In this paper, we provide another solution by introducing an informal heuristic for reasoning about functions defined over iterated smash products. This approach vastly reduces the complexity of the identity proofs involved. We use this to give a complete proof of the fact that the smash product is 1-coherent symmetric monoidal. We finally discuss how to make the heuristic formal and express (a version of it) as a theorem which we also prove (see Theorem 30).

The paper is structured as follows. In Section 2, we introduce the two fundamental concepts of interest, namely symmetric monoidal wild categories and smash products. Section 3 introduces a new higher inductive type (HIT) capturing double smash products. We use this to sketch the construction of the associator map $\alpha_{A,B,C} : (A \wedge B) \wedge C \xrightarrow{\sim} A \wedge (B \wedge C)$. In Section 4, we make a small detour and discuss induction principles for the smash product. It is here that we introduce the heuristic mentioned above. In Section 5, we apply our heuristic and sketch the proof of the symmetric monoidality of the smash product. Finally, in Section 6, we discuss the translation of our heuristic into a formal result. We provide one suggestion and prove it correct.

This paper is written in the informal flavour of type theory employed in, for example, The Univalent Foundations Program (2013). Nevertheless, all key results have been formalised in the proof assistant Cubical Agda (Vezzosi et al. 2021), a cubical extension of Agda which, in particular, enjoys native support for HITs. A file summarising the relevant formalisations can be found in the agda/cubical library at https://github.com/agda/cubical/blob/master/Cubical/Papers/SmashProducts.agda (and, alternatively, on a frozen branch at https://github.com/aljungstrom/cubical/blob/smashpaper/Cubical/Papers/SmashProducts.agda).

2. Background

Let us briefly introduce the key concepts of this paper: symmetric monoidal wild categories and smash products. We assume familiarity with HoTT and refer to the HoTT Book (The Univalent Foundations Program 2013) for the basic constructions and definitions used here. For the remainder of this paper, we adopt the convention that square-shaped diagrams denote (i) commutative squares of functions whenever their source is in the top-left corner and (ii) commutative squares of paths whenever their source is in the bottom-left corner. For instance, the left diagram below expresses the identity (of functions) $g \circ f = i \circ h$, whereas the right diagram below expresses the identity (of paths) $p \cdot q = r \cdot s$.

$$\begin{array}{ccc} A & \xrightarrow{h} & B \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle i} \\ C & \xrightarrow[g]{} & D \end{array} \qquad \begin{array}{ccc} y & \xrightarrow{q} & w \\ {\scriptstyle p}\uparrow & & \uparrow{\scriptstyle s} \\ x & \xrightarrow[r]{} & z \end{array}$$

2.1 Symmetric monoidal wild categories

To make the statements in this paper somewhat more compact, we employ the framework of *wild categories*. These are defined similarly to categories but without the condition that the morphisms form a set (Capriotti and Kraus 2017):

Definition 1 (Wild categories). *A wild category is a category with a type of objects and types of morphisms.*

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

The difference between a wild category and a category is that the latter asks for *sets* of morphisms. In this paper, the wild category of interest is that of pointed types (at some universe level which is left implicit in this paper).

Proposition 2. *Let* $\mathsf{Type}_\star$ *denote the universe of pointed types (at some universe level). This universe forms a wild category with* $\mathsf{Type}_\star[A, B] :\equiv (A \to_\star B)$*, that is, with pointed functions as morphisms.*

The main goal of this paper is to show that $\mathsf{Type}_\star$ is not only a wild category but also a *symmetric monoidal* wild category, so let us define this.

Definition 3 (Monoidal wild categories). *A monoidal wild category is a wild category* M *with*

- *a functor* $\otimes : M \times M \to M$,
- *a unit, that is, an element* $I : M$ *with natural isomorphisms* $\lambda_A : I \otimes A \cong A$ *and* $\rho_A : A \otimes I \cong A$,
- *a family of isomorphisms* $\alpha_{A,B,C} : ((A \otimes B) \otimes C) \cong (A \otimes (B \otimes C))$ *natural in all arguments such that*
  - *the triangle identity holds, that is, the following diagram commutes,*

$$\begin{array}{ccc} (A \otimes I) \otimes B & \xrightarrow{\alpha_{A,I,B}} & A \otimes (I \otimes B) \\ & {\scriptstyle \rho_A \otimes 1_B} \searrow \quad \swarrow {\scriptstyle 1_A \otimes \lambda_B} & \\ & A \otimes B & \end{array}$$

  - *the pentagon identity holds, that is, the following diagram commutes.*

$$\begin{array}{ccc} & ((A \otimes B) \otimes C) \otimes D & \\ {\scriptstyle \alpha_{A,B,C} \otimes 1_D} \swarrow & & \searrow {\scriptstyle \alpha_{A \otimes B,C,D}} \\ (A \otimes (B \otimes C)) \otimes D & & (A \otimes B) \otimes (C \otimes D) \\ {\scriptstyle \alpha_{A,B \otimes C,D}} \downarrow & & \downarrow {\scriptstyle \alpha_{A,B,C \otimes D}} \\ A \otimes ((B \otimes C) \otimes D) & \xrightarrow{1_A \otimes \alpha_{B,C,D}} & A \otimes (B \otimes (C \otimes D)) \end{array}$$

Definition 4 (Symmetric monoidal wild categories). *A symmetric monoidal wild category is a monoidal wild category equipped with a family of isomorphisms* $\beta_{A,B} : A \otimes B \cong B \otimes A$*, natural in both arguments, such that*

- $\beta_{B,A} \circ \beta_{A,B} = 1_{A \otimes B}$,
- *the hexagon identity holds, that is, the following diagram commutes.*

$$\begin{array}{ccc} (A \otimes B) \otimes C & \xrightarrow{\beta_{A,B} \otimes 1_C} & (B \otimes A) \otimes C \\ {\scriptstyle \alpha_{A,B,C}} \downarrow & & \downarrow {\scriptstyle \alpha_{B,A,C}} \\ A \otimes (B \otimes C) & & B \otimes (A \otimes C) \\ {\scriptstyle \beta_{A,B \otimes C}} \downarrow & & \downarrow {\scriptstyle 1_B \otimes \beta_{A,C}} \\ (B \otimes C) \otimes A & \xrightarrow[\alpha_{B,C,A}]{} & B \otimes (C \otimes A) \end{array}$$

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

2.2 Smash products

The model of the smash product we use here is given by the cofibre of the inclusion $A \vee B \hookrightarrow A \times B$, that is, the following (homotopy) pushout.

$$\begin{array}{ccc} A \vee B & \longrightarrow & A \times B \\ \downarrow & \ulcorner & \downarrow \\ 1 & \longrightarrow & A \wedge B \end{array}$$

For the sake of clarity, let us spell this out in detail by giving explicit names to all constructors of $A \wedge B$. We give the smash product the following self-contained definition.

Definition 5 (Smash products). *The smash product of two pointed types A and B is the HIT generated by:*

- *a point* $\star_\wedge : A \wedge B$,
- *points* $\langle a, b\rangle : A \wedge B$ *for every pair* $(a, b) : A \times B$,
- *paths* $\mathsf{push}_\mathsf{l}(a) : \langle a, \star_B\rangle = \star_\wedge$ *for every point* $a : A$,
- *paths* $\mathsf{push}_\mathsf{r}(b) : \langle \star_A, b\rangle = \star_\wedge$ *for every point* $b : B$,
- *a coherence* $\mathsf{push}_\mathsf{lr} : \mathsf{push}_\mathsf{l}(\star_A) = \mathsf{push}_\mathsf{r}(\star_B)$.

We always take $A \wedge B$ to be pointed by $\star_\wedge$.

Remark 6. *We remark that we could equivalently have defined the smash product by the following pushout.*

$$\begin{array}{ccc} A + B & \longrightarrow & A \times B \\ \downarrow & \ulcorner & \downarrow \\ 1 + 1 & \longrightarrow & A \wedge B \end{array}$$

This definition has the advantage of not having any 2-dimensional path constructors but has the disadvantage of having an additional point constructor. It turns out that Definition 5 suits our purposes better, so we stick with it. Alternatively, we could have defined the smash product to have $\langle \star_A, \star_B\rangle$ as its canonical basepoint. Such a definition is obtained by forming the HIT generated by:

- *points* $\langle a, b\rangle : A \wedge B$ *for every pair* $(a, b) : A \times B$,
- *paths* $\mathsf{push}_\mathsf{l}(a) : \langle a, \star_B\rangle = \langle \star_A, \star_B\rangle$ *for every point* $a : A$,
- *paths* $\mathsf{push}_\mathsf{r}(b) : \langle \star_A, b\rangle = \langle \star_A, \star_B\rangle$ *for every point* $b : B$,
- *a coherence* $\mathsf{push}_\mathsf{lr} : \mathsf{push}_\mathsf{l}(\star_A) = \mathsf{push}_\mathsf{r}(\star_B)$,
- *a coherence* $\mathsf{push}_{\mathsf{l}\star} : \mathsf{push}_\mathsf{l}(\star_A) = \mathsf{refl}$.

It was suggested to us by an anonymous reviewer that this definition could allow certain statements in this paper to be expressed in a slightly less convoluted way (in particular Lemmas 19 and 20). We stick to Definition 5 as it is more commonly used in the HoTT literature but remark that the interested reader may find it enlightening to reinterpret the results of this paper in terms of the definition above.

Let us verify that the smash product is functorial. In what follows, a pointed function $A \to_\star B$ is a function $f : A \to B$ equipped with a proof of pointedness $\star_f : f(\star_A) = \star_B$.

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

Definition 7. *For two pointed functions $f : A \to_\star C$ and $g : B \to_\star D$, there is an induced map $f \wedge g : A \wedge B \to_\star C \wedge D$ defined on point constructors by:*

$$(f \wedge g)\,(\star_\wedge) = \star_\wedge$$
$$(f \wedge g)\,\langle a, b\rangle = \langle f(a), g(b)\rangle$$

and on (1-dimensional) path constructors by the following compositions:

$$\mathsf{ap}_{f\wedge g}(\mathsf{push_l}(a)) \;=\; \langle f(a), g(\star_B)\rangle \xrightarrow{\mathsf{ap}_{\langle f(a),-\rangle}(\star_g)} \langle f(a), \star_D\rangle \xrightarrow{\mathsf{push_l}(f(a))} \star_\wedge$$
$$\mathsf{ap}_{f\wedge g}(\mathsf{push_r}(b)) \;=\; \langle f(\star_A), g(b)\rangle \xrightarrow{\mathsf{ap}_{\langle -,g(b)\rangle}(\star_f)} \langle \star_C, g(b)\rangle \xrightarrow{\mathsf{push_r}(g(b))} \star_\wedge$$

The final case of the definition, that is, $\mathsf{ap}_{\mathsf{ap}_{f\wedge g}}(\mathsf{push_{lr}})$, is a simple coherence which does not matter in the remainder of the paper. We take $f \wedge g$ to be pointed by refl.

We also take the opportunity to mention the commutativity of smash products. This is trivial since the definition of $A \wedge B$ is entirely symmetric in both arguments.

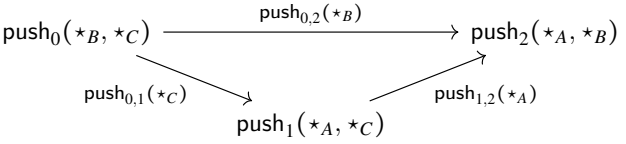
Proposition 8. *The swap map $A \times B \to B \times A$ induces a pointed equivalence $A \wedge B \simeq_\star B \wedge A$.*

3. Associativity

Proving that the smash product is associative is far less straightforward than proving that it is commutative. In fact, even the seemingly direct task of constructing the associator map is no mean feat. While associativity has already been verified by van Doorn (2018) and, using a computer-generated proof, by Brunerie (2018), let us give a direct construction of the equivalence. We do this because an explicit description makes the associator easier to trace when verifying, for example, the pentagon identity. For this purpose, let us introduce a new HIT capturing double smash products in a way which is neutral with respect to the distribution of parentheses.

Definition 9. *Given pointed types A, B and C, we define the type $\bigwedge(A, B, C)$ to be the HIT generated by:*

- *a point* $\star_{2\wedge}$,
- *points* $\langle a, b, c\rangle : \bigwedge(A, B, C)$ *for each triple of points* $(a, b, c) : A \times B \times C$,
- *paths* $\mathsf{push}_0(b, c) : \langle \star_A, b, c\rangle = \star_{2\wedge}$ *for each pair* $(b, c) : B \times C$,
- *paths* $\mathsf{push}_1(a, c) : \langle a, \star_B, c\rangle = \star_{2\wedge}$ *for each pair* $(a, c) : A \times C$,
- *paths* $\mathsf{push}_2(a, b) : \langle a, b, \star_C\rangle = \star_{2\wedge}$ *for each pair* $(a, b) : A \times B$,
- *paths* $\mathsf{push}_{1,2}(a) : \mathsf{push}_1(a, \star_C) = \mathsf{push}_2(a, \star_B)$ *for* $a : A$,
- *paths* $\mathsf{push}_{0,2}(b) : \mathsf{push}_0(b, \star_C) = \mathsf{push}_2(\star_A, b)$ *for* $b : B$,
- *paths* $\mathsf{push}_{0,1}(c) : \mathsf{push}_0(\star_B, c) = \mathsf{push}_1(\star_A, c)$ *for* $c : C$,
- *a coherence* $\mathsf{push}_{0,1,2}$ *filling the following triangle.*



https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

Table 1. $(A \wedge B) \wedge C$ vs. $\bigwedge (A, B, C)$

$(A \wedge B) \wedge C$	$\to$	$\bigwedge (A, B, C)$
$\star_\wedge$	$\mapsto$	$\star_{2\wedge}$
$\langle \star_\wedge, c \rangle$	$\mapsto$	$\star_{2\wedge}$
$\langle \langle a, b \rangle, c \rangle$	$\mapsto$	$\langle a, b, c \rangle$
$\mathrm{ap}_{\langle -, c \rangle}(\mathrm{push}_{\mathsf{l}}(a))$	$\mapsto$	$\mathrm{push}_{1}(a, c)$
$\mathrm{ap}_{\langle -, c \rangle}(\mathrm{push}_{\mathsf{r}}(b))$	$\mapsto$	$\mathrm{push}_{0}(b, c)$
$\mathrm{ap}_{\mathrm{ap}_{\langle -, c \rangle}}(\mathrm{push}_{\mathsf{lr}})$	$\mapsto$	$\mathrm{push}_{0,1}(c)$
$\mathrm{push}_{\mathsf{l}}(\star_\wedge)$	$\mapsto$	refl
$\mathrm{push}_{\mathsf{l}}\langle a, b \rangle$	$\mapsto$	$\mathrm{push}_{2}(a, b)$
$\mathrm{ap}_{\mathrm{push}_{\mathsf{l}}}(\mathrm{push}_{\mathsf{l}}(a))$	$\mapsto$	$push_{1,2}(a)$
$\mathrm{ap}_{\mathrm{push}_{\mathsf{l}}}(\mathrm{push}_{\mathsf{r}}(b))$	$\mapsto$	$push_{0,2}(b)$
$\mathrm{ap}_{\mathrm{ap}_{\mathrm{push}_{\mathsf{l}}}}(\mathrm{push}_{\mathsf{lr}})$	$\mapsto$	$\mathrm{push}_{0,1,2}$
$\mathrm{push}_{\mathsf{r}}(c)$	$\mapsto$	refl
$\mathrm{push}_{\mathsf{lr}}$	$\mapsto$	refl

Let us verify that this actually captures a double smash product. What we need is an equivalence $(A \wedge B) \wedge C \simeq \bigwedge (A, B, C)$. The underlying map of this equivalence is described in Table 1 with constructors of $(A \wedge B) \wedge C$ on the left and the corresponding constructors of $\bigwedge (A, B, C)$ on the right. We remark that this correspondence only serves as an informal sketch of the function – in practice, some simple coherences are needed for the higher constructors to make it well typed. Verifying that this map indeed defines an equivalence is somewhat laborious but direct; the interested reader is referred to the computer formalisation. We get the associativity of the smash product as a consequence.

Proposition 10. *There is a pointed equivalence* $\alpha_{A,B,C} : (A \wedge B) \wedge C \simeq_\star A \wedge (B \wedge C)$

Proof. Observe that $\bigwedge (A, B, C)$ is trivially invariant under permutation of the arguments in the sense that, for example, $\bigwedge (A, B, C) \simeq \bigwedge (B, C, A)$. This allows us to define $\alpha_{A,B,C}$ by the following composition of equivalences:

$$(A \wedge B) \wedge C \simeq \bigwedge (A, B, C) \simeq \bigwedge (B, C, A) \simeq (B \wedge C) \wedge A \simeq A \wedge (B \wedge C)$$

The fact that $\alpha_{A,B,C}$ is pointed holds by refl. □

4. The Heuristic

Reasoning about functions defined over iterated smash products quickly gets out of hand. For instance, in order to verify the pentagon axiom, we need to reason about functions on the form $((A \wedge B) \wedge C) \wedge D \to E$. To prove an equality of two such functions f and g, we have to construct, for instance, a dependent path:

$$\mathrm{ap}_{\mathrm{ap}_{\mathrm{ap}_f \circ \mathrm{push}_{\mathsf{l}}} \circ \mathrm{push}_{\mathsf{l}}}(\mathrm{push}_{\mathsf{l}}(a)) \;\rightsquigarrow\; \mathrm{ap}_{\mathrm{ap}_{\mathrm{ap}_g \circ \mathrm{push}_{\mathsf{l}}} \circ \mathrm{push}_{\mathsf{l}}}(\mathrm{push}_{\mathsf{l}}(a)) \tag{2}$$

which boils down to filling a 4-dimensional cube with highly non-trivial sides. This is often completely unmanageable in practice. The best thing we can hope for is that these types of coherence

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

problems are, in some sense, automatic. This was, in fact, suggested by Brunerie (2016) but was never proved nor in any way made formal. In this section, we will see that this, in fact, is the case.

The first troublesome part of verifying equalities of functions defined over smash products is the $\mathsf{push}_{\mathsf{lr}}$ constructor. Fortunately, we do not have to deal with it. Let us denote by $A \mathbin{\widetilde{\wedge}} B$ the exact same HIT as $A \wedge B$ but with the $\mathsf{push}_{\mathsf{lr}}$ constructor removed. In other words, it is the following pushout.

$$\begin{array}{ccc} A + B & \longrightarrow & A \times B \\ \downarrow & \ulcorner & \downarrow \\ 1 & \longrightarrow & A \mathbin{\widetilde{\wedge}} B \end{array}$$

Let i be the obvious map $A \mathbin{\widetilde{\wedge}} B \to A \wedge B$.

Lemma 11. *For any two maps $f, g : A \wedge B \to C$ satisfying $f \circ i = g \circ i$, we have that $f = g$.*

Proof. The antecedent of the statement provides us with

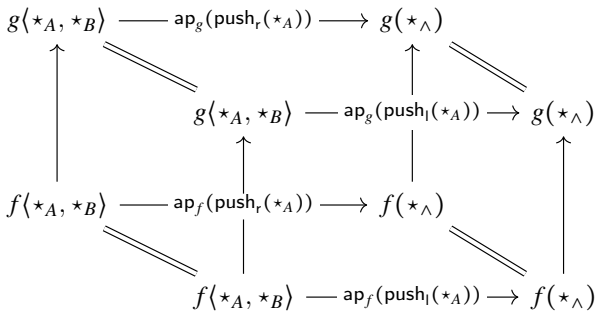
- a path $p : f(\star_\wedge) = g(\star_\wedge)$,
- a homotopy $h : ((a, b) : A \times B) \to f\langle a, b\rangle = g\langle a, b\rangle$,
- for each $a : A$, a filler $h_l(a)$ of the square

$$\begin{array}{ccc} g\langle a, \star_B\rangle & \xrightarrow{\mathsf{ap}_g(\mathsf{push}_{\mathsf{l}}(a))} & g(\star_\wedge) \\ {\scriptstyle h(a,\star_B)}\uparrow & & \uparrow{\scriptstyle p} \\ f\langle a, \star_B\rangle & \xrightarrow[\mathsf{ap}_f(\mathsf{push}_{\mathsf{l}}(a))]{} & f(\star_\wedge) \end{array}$$

- for each $b : B$, a filler $h_r(b)$ of the square

$$\begin{array}{ccc} g\langle \star_A, b\rangle & \xrightarrow{\mathsf{ap}_g(\mathsf{push}_{\mathsf{r}}(b))} & g(\star_\wedge) \\ {\scriptstyle h(\star_A,b)}\uparrow & & \uparrow{\scriptstyle p} \\ f\langle \star_A, b\rangle & \xrightarrow[\mathsf{ap}_f(\mathsf{push}_{\mathsf{r}}(b))]{} & f(\star_\wedge) \end{array}$$

To prove that $f = g$, we need to provide p', h', h'_l, h'_r of the same types as above, as well as a filler h'_{lr} of the cube



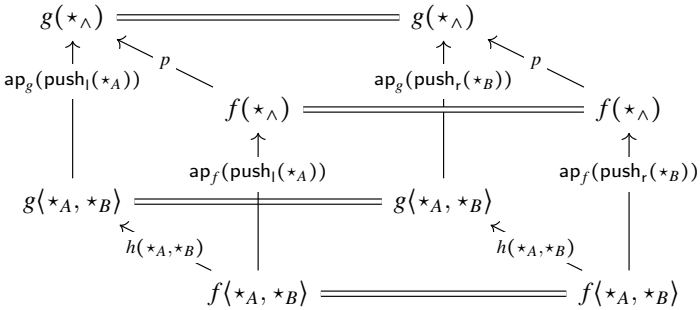
https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

where the top and bottom squares are given, respectively, by $\mathsf{ap}_{\mathsf{ap}_g}(\mathsf{push}_{\mathsf{lr}})$ and $\mathsf{ap}_{\mathsf{ap}_f}(\mathsf{push}_{\mathsf{lr}})$, the left- and right-hand side, respectively, by $\mathsf{refl}_{h'(\star_A,\star_B)}$ and $\mathsf{refl}_{p'}$ and the front and back, respectively, by $h'_l(\star_A)$ and $h'_r(\star_B)$.

We set $p' := p$, $h' := h$ and $h'_l := h_l$. For h'_r, however, we need to make an alteration. We construct it as the following composite square

$$\begin{array}{ccccc} g\langle \star_A, b\rangle & \xrightarrow{\mathsf{ap}_g(\mathsf{push}_{\mathsf{r}}(b))} & g(\star_\wedge) & = & g(\star_\wedge) \\ {\scriptstyle h(\star_A,b)}\uparrow & & \uparrow{\scriptstyle p} & & \uparrow{\scriptstyle p} \\ f\langle \star_A, b\rangle & \xrightarrow[\mathsf{ap}_f(\mathsf{push}_{\mathsf{r}}(b))]{} & f(\star_\wedge) & = & f(\star_\wedge) \end{array}$$

where the square on the left is $h_r(b)$ and the square on the right is the lid of the cube



with $h_l(\star_A)$ and $h_r(\star_B)$ as left- and right-hand sides, the action of f and g on $\mathsf{push}_{\mathsf{lr}}$ on the front and back, and $\mathsf{refl}_{h(\star_A,\star_B)}$ on the bottom. One can now easily construct the filler h'_{lr} by generalising the cubes involved and applying path induction. □

Remark 12. *Lemma 11 is a special case of the more general statement that i is a retraction (this was called to our attention by Dan Petersen). In fact, there is an equivalence $A \mathbin{\widetilde{\wedge}} B \simeq S^1 \vee (A \wedge B)$ under which the map i becomes the canonical retraction $S^1 \vee (A \wedge B) \to A \wedge B$. To see this, consider the following commutative diagram.*

$$\begin{array}{ccccc} 1 & \longleftarrow & 2 & \xrightarrow{\mathsf{id}} & 2 \\ \uparrow & & {\scriptstyle \mathsf{id}}\uparrow & & \uparrow \\ 2 & \xleftarrow{\mathsf{id}} & 2 & \longrightarrow & A+B \\ \downarrow & & \downarrow & & \downarrow \\ 1 & \longleftarrow & 1 & \longrightarrow & A\times B \end{array}$$

By taking pushouts of the rows, we produce the span $1 \leftarrow A + B \to A \times B$, the pushout of which is $A \mathbin{\widetilde{\wedge}} B$. On the other hand, by taking pushouts of the columns, we produce the span $S^1 \leftarrow 1 \to A \wedge B$, the pushout of which is $S^1 \vee (A \wedge B)$. This yields the desired equivalence $A \mathbin{\widetilde{\wedge}} B \simeq S^1 \vee (A \wedge B)$ by the 3×3-lemma (introduced by Brunerie 2016, *Lemma 1.8.3 with computer formalisation by* Pujet and Mörtberg 2020). *The fact that i factors in the appropriate manner follows by construction. Thus, we have given an alternative proof of Lemma 11.*

Lemma 11 is useful but does not get us all the way. Complicated paths like (2) still need to be constructed, regardless of what happens with the $\mathsf{push}_{\mathsf{lr}}$ constructor. To strengthen the principle,

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

we will need to introduce the concept of *homogeneous types* which, to the best of our knowledge, was first introduced (in HoTT) by Kraus (2013).

Definition 13. *A pointed type A is homogeneous if for any $a : A$, there is a pointed equivalence $(A, \star_A) \simeq_\star (A, a)$.*

The usefulness of homogeneous types is showcased by the following lemma which was first conjectured for Eilenberg–MacLane spaces in work leading up to Brunerie et al. (2022) and later proved and generalised by Cavallo (2021b) (and later further generalised by Buchholtz et al. 2023).

Lemma 14 (Cavallo). *Let $f, g : A \to_\star B$ and let B be homogeneous. If $f = g$ as plain functions, then $f = g$ as pointed functions.*

The same lemma holds for bi-pointed functions $f, g : A \to_\star (B \to_\star C)$, since the type $(B \to_\star C)$ is homogeneous if C is. This gives a corresponding principle for maps defined over smash products via the adjunction $(A \wedge B \to_\star C) \simeq (A \to_\star (B \to_\star C))$.

Lemma 15. *Let $f, g : A \wedge B \to_\star C$ and let C be homogeneous. If $f\langle a, b\rangle = g\langle a, b\rangle$ for all $a : A$ and $b : B$, we obtain an equality of pointed functions $f = g$.*

The following form is useful when dealing with non-pointed functions.

Lemma 16. *Let C be an arbitrary type and suppose that we have two functions $f, g : A \wedge B \to C$ with $(C, f(\star_\wedge))$ homogeneous. If $f\langle a, b\rangle = g\langle a, b\rangle$ for all $a : A$ and $b : B$, then $f = g$.*

Proof. Let $h(a, b) : f\langle a, b\rangle = g\langle a, b\rangle$. We know that $g(\star_\wedge) = f(\star_\wedge)$ by the composite path

$$g(\star_\wedge) \xrightarrow{\mathsf{ap}_g(\mathsf{push}_\mathsf{l}(\star_A))^{-1}} g\langle \star_A, \star_B\rangle \xrightarrow{h(\star_A,\star_B)^{-1}} f\langle \star_A, \star_B\rangle \xrightarrow{\mathsf{ap}_f(\mathsf{push}_\mathsf{l}(\star_A))} f(\star_\wedge)$$

Hence, we may view both f and g as pointed functions $A \wedge B \to_\star (C, f(\star_\wedge))$. Now, Lemma 15 applies and, in particular, $f = g$ as plain functions. □

If we could apply Lemma 15 or 16 when proving the pentagon identity, we would be done immediately. Unfortunately, none of the types showing up in the statement of the pentagon identity are necessarily homogeneous. There is, however, some use for the lemmas. Let us first make the following observation: given two pointed functions $f, g : A \wedge B \to_\star C$ and a homotopy $h : ((a, b) : A \times B) \to f\langle a, b\rangle = g\langle a, b\rangle$, we can define two functions $L_h : A \to \Omega(C)$ and $R_h : B \to \Omega(C)$ in terms of h and $\mathsf{push}_\mathsf{l}/\mathsf{push}_\mathsf{r}$. The obvious definition of these maps (which we will spell out in Definition 17) will give us a version of Lemma 11 which tells us that if L_h and R_h are constant, then $f = g$ as plain functions. Now, if either A or B is another smash product, this *would* be a situation where Lemma 16 applies since $\Omega(C)$ (and indeed any path type) is homogeneous. This suggests that Lemma 16 may be used our advantage when dealing with iterated smash products. Let us try to spell this out more clearly. We can state the idea without any pointedness assumptions by simply replacing $\Omega(C)$ with the path type $f(\star_\wedge) = g(\star_\wedge)$ which is also homogeneous (and agrees with $\Omega(C)$ whenever f and g are pointed).

Definition 17. *Let $f, g : A \wedge B \to C$ and let $h : ((a, b) : A \times B) \to f\langle a, b\rangle = g\langle a, b\rangle$. This induces functions $L_h : A \to f(\star_\wedge) = g(\star_\wedge)$ and $R_h : B \to f(\star_\wedge) = g(\star_\wedge)$ defined by:*

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

$$L_h(a) = \mathsf{ap}_f(\mathsf{push}_{\mathsf{l}}(a))^{-1} \cdot h(a, \star_B) \cdot \mathsf{ap}_g(\mathsf{push}_{\mathsf{l}}(a))$$
$$R_h(b) = \mathsf{ap}_f(\mathsf{push}_{\mathsf{r}}(b))^{-1} \cdot h(\star_A, b) \cdot \mathsf{ap}_g(\mathsf{push}_{\mathsf{r}}(b))$$

We may use L_h and R_h to give a compact induction principle for identities $f = g$. The following lemma is a direct rewriting of Lemma 11.

Lemma 18. *Let $f, g : A \wedge B \to C$. The following data yields an equality $f = g$.*

- *A homotopy $h : ((a, b) : A \times B) \to f\langle a, b\rangle = g\langle a, b\rangle$,*
- *paths $L_h = \mathsf{const}_{L_h(\star_A)}$ and $R_h = \mathsf{const}_{R_h(\star_B)}$.*

where const_y denotes the constant function, that is, $\mathsf{const}_y(x) = y$. If C is a pointed type, f and g are pointed functions and an equality $f = g$ of pointed functions is desired, a further coherence $\star_f = L_h(\star_A) \cdot \star_g$ is required.

The key idea now is, we stress again, to use Lemmas 18 and 15/16 iteratively to prove equalities of functions defined over iterated smash products. As a first example, let us consider the problem of proving an equality of functions $f = g$ where $f, g : (A \wedge B) \wedge C \to D$. The following lemma provides a good estimate of the minimal amount of data needed to construct such an equality.

Lemma 19. *For two functions $f, g : (A \wedge B) \wedge C \to D$, the following data gives an equality $f = g$.*

(i) A homotopy $h : ((a, b, c) : A \times B \times C) \to f\langle a, b, c\rangle = g\langle a, b, c\rangle$,

(ii) for every pair $(a, c) : A \times C$, a filler of the square

$$\begin{array}{ccc}
f\langle \star_A, \star_B, c\rangle & \xrightarrow{h(\star_A, \star_B, c)} & g\langle \star_A, \star_B, c\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -, c\rangle}(\mathsf{push}_{\mathsf{l}}(\star_A))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -, c\rangle}(\mathsf{push}_{\mathsf{l}}(\star_A))^{-1}} \\
f\langle \star_\wedge, c\rangle & & g\langle \star_\wedge, c\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -, c\rangle}(\mathsf{push}_{\mathsf{l}}(a))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -, c\rangle}(\mathsf{push}_{\mathsf{l}}(a))} \\
f\langle a, \star_B, c\rangle & \xrightarrow[h(a, \star_B, c)]{} & g\langle a, \star_B, c\rangle
\end{array}$$

(iii) for every pair $(b, c) : B \times C$, a filler of the square

$$\begin{array}{ccc}
f\langle \star_A, \star_B, c\rangle & \xrightarrow{h(\star_A, \star_B, c)} & g\langle \star_A, \star_B, c\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -, c\rangle}(\mathsf{push}_{\mathsf{r}}(\star_B))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -, c\rangle}(\mathsf{push}_{\mathsf{r}}(\star_B))^{-1}} \\
f\langle \star_\wedge, c\rangle & & g\langle \star_\wedge, c\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -, c\rangle}(\mathsf{push}_{\mathsf{r}}(b))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -, c\rangle}(\mathsf{push}_{\mathsf{r}}(b))} \\
f\langle \star_A, b, c\rangle & \xrightarrow[h(\star_A, b, c)]{} & g\langle \star_A, b, c\rangle
\end{array}$$

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

(iv) for every pair $(a, b) : A \times B$*, a filler of the square*

$$\begin{array}{ccc}
f\langle \star_A, \star_B, \star_C\rangle & \xrightarrow{h(\star_A,\star_B,\star_C)} & g\langle \star_A, \star_B, \star_C\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,\star_C\rangle}(\mathsf{push}_r(\star_B)^{-1}))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,\star_C\rangle}(\mathsf{push}_r(\star_B)^{-1}))} \\
f\langle \star_\wedge, \star_C\rangle & & g\langle \star_\wedge, \star_C\rangle \\
{\scriptstyle \mathsf{ap}_f(\mathsf{push}_l(\star_\wedge)^{-1})}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_g(\mathsf{push}_l(\star_\wedge)^{-1})} \\
f(\star_\wedge) & & g(\star_\wedge) \\
{\scriptstyle \mathsf{ap}_f(\mathsf{push}_l\langle a,b\rangle)}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_g(\mathsf{push}_l\langle a,b\rangle)} \\
f\langle a, b, \star_C\rangle & \xrightarrow[h(a,b,\star_C)]{} & g\langle a, b, \star_C\rangle
\end{array}$$

(v) for every point $c : C$*, a filler of the square*

$$\begin{array}{ccc}
f\langle \star_A, \star_B, \star_C\rangle & \xrightarrow{h(\star_A,\star_B,\star_C)} & g\langle \star_A, \star_B, \star_C\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,\star_C\rangle}(\mathsf{push}_r(\star_B))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,\star_C\rangle}(\mathsf{push}_r(\star_B))^{-1}} \\
f\langle \star_\wedge, \star_C\rangle & & g\langle \star_\wedge, \star_C\rangle \\
{\scriptstyle \mathsf{ap}_f(\mathsf{push}_l(\star_\wedge))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_g(\mathsf{push}_l(\star_\wedge))^{-1}} \\
f(\star_\wedge) & & g(\star_\wedge) \\
{\scriptstyle \mathsf{ap}_f(\mathsf{push}_r(c))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_g(\mathsf{push}_r(c))} \\
f\langle \star_\wedge, c\rangle & & g\langle \star_\wedge, c\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,c\rangle}(\mathsf{push}_r(\star_B))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,c\rangle}(\mathsf{push}_r(\star_B))} \\
f\langle \star_A, \star_B, c\rangle & \xrightarrow[h(\star_A,\star_B,c)]{} & g\langle \star_A, \star_B, c\rangle
\end{array}$$

Proof sketch. Suppose we have the given data. We apply Lemma 18 to f and g. This breaks the proof up into two 2 subgoals.

- First, we need to provide a homotopy $k : ((x, c) : (A \wedge B) \times C) \to f\langle x, c\rangle = g\langle x, c\rangle$. To construct k, we fix $c : C$ and note that we may now apply Lemma 18 again: this time to the functions $f(-, c)$ and $g(-, c)$. This gives us two new subgoals.
  - First, we need a homotopy $((a, b, c) : A \times B \times C) \to f\langle a, b, c\rangle = g\langle a, b, c\rangle$. This is given by h.
  - We finally need to show that $L_{h(-,c)}$ and $R_{h(-,c)}$ are constant. This boils down to providing fillers of the squares which we assumed in (ii) and (iii).
- We then need to show that L_k and R_k are constant. To show that L_k is constant, we apply Lemma 16. This is justified since the codomain of L_k is homogeneous. Hence, we only need to verify that $L_k\langle a, b\rangle = L_k(\star_\wedge)$. Unfolding the definitions, we see that this is given by assumption (iv). Note that this is where the explosion of complexity would normally happen but, thanks to Lemma 16, we completely avoid having to verify any higher coherences. For R_k, we again unfold the definitions to see that the path required is provided by (v). □

For completeness, let us state the corresponding lemma for functions $f, g : ((A \wedge B) \wedge C) \wedge D \to E$, as these appear in the pentagon identity. The proof is by Lemmas 18,16 and 19, following

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

the exact same line of attack as before. We acknowledge that the following result is highly technical but stress that the interesting aspect of it is not the exact details of the statement; rather, we include it to showcase the fact that only squares are involved, as opposed to the (many) high-dimensional cubes of coherences which would appear in a naive inductive proof.

Lemma 20. *For any two functions $f, g : ((A \wedge B) \wedge C) \wedge D \to E$, the following data gives an equality $f = g$:*

(i) A homotopy $h : ((a, b, c, d) : A \times B \times C \times D) \to f\langle a, b, c, d\rangle = g\langle a, b, c, d\rangle$.

(ii) For every triple $(a, b, c) : A \times B \times C$, a filler of the square

$$\begin{array}{ccc}
f\langle \star_A, \star_B, \star_C \star_D\rangle & \xrightarrow{h(\star_A,\star_B,\star_C\star_D)} & g\langle \star_A, \star_B, \star_C \star_D\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,\star_C,\star_D\rangle}(\mathsf{push}_{\mathsf{l}}(\star_A)))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,\star_C,\star_D\rangle}(\mathsf{push}_{\mathsf{l}}(\star_A)))^{-1}} \\
f\langle \star_\wedge, \star_C, \star_D\rangle & & g\langle \star_\wedge, \star_C, \star_D\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,\star_D\rangle}(\mathsf{push}_{\mathsf{l}}(\star_\wedge)))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,\star_D\rangle}(\mathsf{push}_{\mathsf{l}}(\star_\wedge)))^{-1}} \\
f\langle \star_\wedge, \star_D\rangle & & g\langle \star_\wedge, \star_D\rangle \\
{\scriptstyle \mathsf{ap}_{f}(\mathsf{push}_{\mathsf{l}}(\star_\wedge)))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g}(\mathsf{push}_{\mathsf{l}}(\star_\wedge)))^{-1}} \\
f(\star_\wedge) & & g(\star_\wedge) \\
{\scriptstyle \mathsf{ap}_{f}(\mathsf{push}_{\mathsf{l}}\langle a,b,c\rangle)}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g}(\mathsf{push}_{\mathsf{l}}\langle a,b,c\rangle)} \\
f\langle a, b, c, \star_D\rangle & \xrightarrow[h(a,b,c,\star_D)]{} & g\langle a, b, c, \star_D\rangle
\end{array}$$

(iii) For every triple $(a, b, d) : A \times B \times D$, a filler of the square

$$\begin{array}{ccc}
f\langle \star_A, \star_B, \star_C, d\rangle & \xrightarrow{h(\star_A,\star_B,\star_C)} & g\langle \star_A, \star_B, \star_C, d\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,\star_C,d\rangle}(\mathsf{push}_{\mathsf{r}}(\star_B)^{-1}))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,\star_C,d\rangle}(\mathsf{push}_{\mathsf{r}}(\star_B)^{-1}))} \\
f\langle \star_\wedge, \star_C, d\rangle & & g\langle \star_\wedge, \star_C, d\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,d\rangle}(\mathsf{push}_{\mathsf{l}}(\star_\wedge)^{-1})}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,d\rangle}(\mathsf{push}_{\mathsf{l}}(\star_\wedge)^{-1})} \\
f\langle \star_\wedge, d\rangle & & g\langle \star_\wedge, d\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,d\rangle}(\mathsf{push}_{\mathsf{l}}\langle a,b\rangle)}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,d\rangle}(\mathsf{push}_{\mathsf{l}}\langle a,b\rangle)} \\
f\langle a, b, \star_C, d\rangle & \xrightarrow[h(a,b,\star_C)]{} & g\langle a, b, \star_C, d\rangle
\end{array}$$

(iv) For every triple $(a, c, d) : A \times C \times D$, a filler of the square

$$\begin{array}{ccc}
f\langle \star_A, \star_B, c, d\rangle & \xrightarrow{h(\star_A,\star_B,c,d)} & g\langle \star_A, \star_B, c, d\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,c,d\rangle}(\mathsf{push}_{\mathsf{l}}(\star_A))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,c,d\rangle}(\mathsf{push}_{\mathsf{l}}(\star_A))^{-1}} \\
f\langle \star_\wedge, c, d\rangle & & g\langle \star_\wedge, c, d\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,c,d\rangle}(\mathsf{push}_{\mathsf{l}}(a))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,c,d\rangle}(\mathsf{push}_{\mathsf{l}}(a))} \\
f\langle a, \star_B, c, d\rangle & \xrightarrow[h(a,\star_B,c,d)]{} & g\langle a, \star_B, c, d\rangle
\end{array}$$

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

(v) For every triple $(b, c, d) : B \times C \times D$*, a filler of the square*

$$\begin{array}{ccc}
f\langle \star_A, \star_B, c, d\rangle & \xrightarrow{h(\star_A,\star_B,c,d)} & g\langle \star_A, \star_B, c, d\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,c,d\rangle}(\mathsf{push_r}(\star_B))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,c,d\rangle}(\mathsf{push_r}(\star_B))^{-1}} \\
f\langle \star_\wedge, c, d\rangle & & g\langle \star_\wedge, c, d\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,c,d\rangle}(\mathsf{push_r}(b))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,c,d\rangle}(\mathsf{push_r}(b))} \\
f\langle \star_A, b, c, d\rangle & \xrightarrow[h(\star_A,b,c,d)]{} & g\langle \star_A, b, c, d\rangle
\end{array}$$

(vi) For every pair $(c, d) : C \times D$*, a filler of the square*

$$\begin{array}{ccc}
f\langle \star_A, \star_B, \star_C, d\rangle & \xrightarrow{h(\star_A,\star_B,\star_C,d)} & g\langle \star_A, \star_B, \star_C\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,\star_C,d\rangle}(\mathsf{push_r}(\star_B))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,\star_C,d\rangle}(\mathsf{push_r}(\star_B))^{-1}} \\
f\langle \star_\wedge, \star_C, d\rangle & & g\langle \star_\wedge, \star_C\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,d\rangle}(\mathsf{push_l}(\star_\wedge))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,d\rangle}(\mathsf{push_l}(\star_\wedge))^{-1}} \\
f\langle \star_\wedge, d\rangle & & g\langle \star_\wedge, d\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,d\rangle}(\mathsf{push_r}(c))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,d\rangle}(\mathsf{push_r}(c))} \\
f\langle \star_\wedge, c, d\rangle & & g\langle \star_\wedge, c, d\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,c,d\rangle}(\mathsf{push_r}(\star_B))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,c,d\rangle}(\mathsf{push_r}(\star_B))} \\
f\langle \star_A, \star_B, c, d\rangle & \xrightarrow[h(\star_A,\star_B,c,d)]{} & g\langle \star_A, \star_B, c, d\rangle
\end{array}$$

(vii) For every $d : D$*, a filler of the square*

$$\begin{array}{ccc}
f\langle \star_A, \star_B, \star_C, \star_D\rangle & \xrightarrow{h(\star_A,\star_B,\star_C,\star_D)} & g\langle \star_A, \star_B, \star_C, \star_D\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,\star_C,\star_D\rangle}(\mathsf{push_l}(\star_A)))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,\star_C,\star_D\rangle}(\mathsf{push_l}(\star_A)))^{-1}} \\
f\langle \star_\wedge, \star_C, \star_D\rangle & & g\langle \star_\wedge, \star_C, \star_D\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,\star_D\rangle}(\mathsf{push_l}(\star_\wedge)))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,\star_D\rangle}(\mathsf{push_l}(\star_\wedge)))^{-1}} \\
f\langle \star_\wedge, \star_D\rangle & & g\langle \star_\wedge, \star_D\rangle \\
{\scriptstyle \mathsf{ap}_{f}(\mathsf{push_l}(\star_\wedge)))^{-1}}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g}(\mathsf{push_l}(\star_\wedge)))^{-1}} \\
f(\star_\wedge) & & g(\star_\wedge) \\
{\scriptstyle \mathsf{ap}_{f}(\mathsf{push_r}(d))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g}(\mathsf{push_r}(d))} \\
f\langle \star_\wedge, d\rangle & & g\langle \star_\wedge, d\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,d\rangle}(\mathsf{push_l}(\star_\wedge)))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,d\rangle}(\mathsf{push_l}(\star_\wedge)))} \\
f\langle \star_\wedge, \star_C, d\rangle & & g\langle \star_\wedge, \star_C, d\rangle \\
{\scriptstyle \mathsf{ap}_{f\langle -,\star_C,d\rangle}(\mathsf{push_l}(\star_A)))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_{g\langle -,\star_C,d\rangle}(\mathsf{push_l}(\star_A)))} \\
f\langle \star_A, \star_B, \star_C, d\rangle & \xrightarrow[h(\star_A,\star_B,\star_C,d)]{} & g\langle \star_A, \star_B, \star_C, d\rangle
\end{array}$$

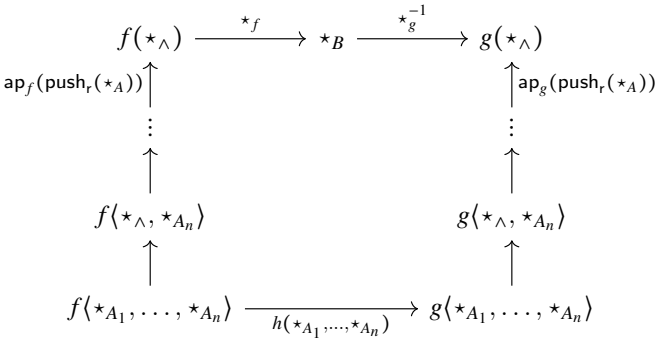
https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

Let us make three observations about Lemmas 19 and 20:

1. In both statements, the different pieces of data are almost completely mutually independent. The only meaningful choice we can make is that of the homotopy h. This means that we are free to provide any proofs we like for the remaining steps without having to worry about future coherences.
2. In many concrete cases (especially those relating to the symmetric monoidal structure of the smash product), the homotopy h will be defined by $h(a_1, \dots, a_n) := \mathsf{refl}$. This means that all other data we need to provide are equalities of composite paths defined entirely in terms of applications of f and g on push_l and push_r – something we can usually simply unfold to something (hopefully) simple.
3. Going from Lemmas 19–20, we see that only two additional assumptions are needed. If we were to increase the number of copies of smash products appearing in the domain by one, we would only need to provide two additional squares (and still no higher coherences). Hence, the complexity of such proofs grows linearly with the complexity of the domain. For comparison, if we were to resort to a naive proof by a deep smash product induction, the amount of data needed would grow exponentially.

While it is possible to generalise Lemmas 19 and 20 to a statement concerning n-fold smash products, let us, for now, only summarise the fundamental idea behind Lemmas 19 and 20 in terms of an informal heuristic.

Heuristic 21. *To show that two functions $f, g : \bigwedge_{i \leq n} A_i \to B$ are equal, it suffices, by iterative application Lemmas 18, 16 and 14, to provide a family of paths $h(x_1, \dots, x_n) : f\langle x_1, \dots, x_n\rangle = g\langle x_1, \dots, x_n\rangle$ for $x_i : A_i$ and to show that it is coherent with f and g on any single application of push_l or push_r (e.g., $\mathsf{ap}_{\langle -, x_{i+1}, \dots, x_n\rangle}(\mathsf{push}_\mathsf{l}\langle x_1, \dots, x_{i-1}\rangle)$). Furthermore, if an equality of pointed functions is required, we need to provide a filler of the following square:*



The reader who is looking for a precise mathematical statement of the above is referred to Section 6 where we also introduce some additional machinery required to make the idea formal. For now, we will settle for this informal heuristic – in practice, when working with functions over a small fixed number of smash products, we do not quite need the full strength of a general theorem.

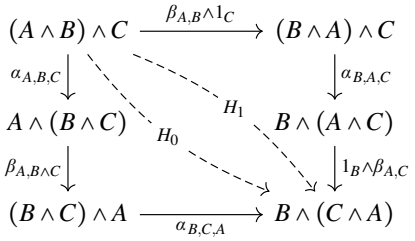
5. The Symmetric Monoidal Structure

Let us reap the fruits of our labour and show that the smash product defines a (1-coherent) symmetric monoidal product on the universe of pointed types. We will not verify all axioms

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

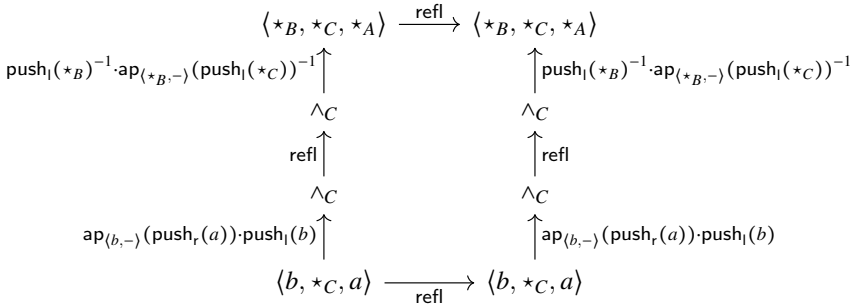
here, since this is neither very instructive nor very interesting. Instead, we sketch the proofs of the two most technical properties and refer the interested reader to the computer formalisation.

Proposition 22. *The smash product satisfies the hexagon identity, that is, we have an equality of pointed functions $H_0 = H_1$ where H_0 and H_1 are defined as the composites of each side of the following hexagon.*



Proof sketch. We show the statement by an application of our heuristic which, in this case, takes the form of Lemma 19. We provide the data as follows:

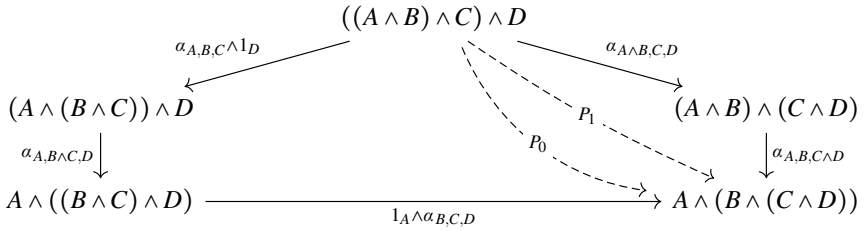
1. For the homotopy $h(a, b, c) : H_0\langle a, b, c\rangle = H_1\langle a, b, c\rangle$, we simply choose $h(a, b, c) := \mathsf{refl}$, since both sides compute to $\langle b, c, a\rangle$.
2. For example, the fourth datum in Lemma 19 computes to[1] the following square-filling problem:



   which is solved by refl.
3. The remaining squares are computed and solved in an identical manner.
4. For the pointedness, we need to fill the square outlined in end of the statement of Heuristic 21. This is equally direct since $\star_{H_0} = \star_{H_1} = \mathsf{refl}$, which holds because all functions involved in the definitions of H_0 and H_1 are pointed by refl. □

Proposition 23. *The pentagon identity holds for the smash product, that is, we have an equality of pointed functions $P_0 = P_1$ where P_0 and P_1 are defined as the composites of each side of the following pentagon.*

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press



Proof. The statement follows easily by the heuristic, which in this case corresponds to Lemma 20. The proof is identical to the proof of Proposition 22 and follows simply by evaluating P_0 and P_1 on the 1-dimensional path constructors involved and noting that all square-filling problems listed in Lemma 20 become trivial. □

All other axioms defining a symmetric monoidal wild category follow in the same direct manner and we, after some rather mechanical labour (see computer formalisation), easily arrive at the main result.

Theorem 24. *The universe of pointed types forms a symmetric monoidal wild category with the smash product as tensor product.*

5.1 Back to Brunerie's thesis

A first consequence of the fact that Theorem 24 finally has a complete and computer formalised proof is the correctness of Brunerie's PhD thesis (2016). While a computer formalisation of the main results of the thesis was presented by Ljungström and Mörtberg (2023), this formalisation did not stay completely true to Brunerie's original proof. Indeed, certain proofs and constructions were reworked in order not to rely on results concerning smash products. Most notably, the *cup product* was redefined using an alternative definition by Brunerie et al. (2022, Section 4.1). In Brunerie's thesis, the cup product (on integral Eilenberg–MacLane spaces) is defined as a map $\smile : K(\mathbb{Z}, n) \wedge K(\mathbb{Z}, m) \to K(\mathbb{Z}, n+m)$ where, for $n, m \geq 1$, we have $K(\mathbb{Z}, n) := \|S^n\|_n$, that is, the n-truncation of the n-sphere. Here, we define $S^n := \Sigma^{n+1}(\emptyset)$, that is, it is the $(n+1)$-fold suspension of the empty type, where, recall, the suspension of a type A, is the HIT $\Sigma(A)$ generated by

- two points $\mathsf{north}, \mathsf{south} : \Sigma(A)$,
- paths $\mathsf{merid}\,(a) : \mathsf{north} = \mathsf{south}$ for each $a : A$.

Brunerie's construction of the cup product is by means of a lift from the corresponding map on spheres.

$$\begin{array}{ccc} S^n \wedge S^m & \xrightarrow{\wedge_{n,m}} & S^{n+m} \\ {\scriptstyle |-|\wedge|-|}\downarrow & & \downarrow{\scriptstyle |-|} \\ K(\mathbb{Z}, n) \wedge K(\mathbb{Z}, m) & \dashrightarrow_{\smile} & K(\mathbb{Z}, n+m) \end{array}$$

Above, the map $\wedge_{n,m} : S^n \wedge S^m \to S^{n+m}$ is the canonical equivalence defined in, for example, Brunerie (2016, Proposition 4.2.2). With this construction, most properties of the cup product can be shown by showing that the corresponding properties hold for $\wedge_{n,m}$. In particular, graded-commutativity and associativity follow, respectively, from the following two propositions.

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

Proposition 25 (Brunerie 2016, Proposition 4.2.4). *The following diagram commutes.*

$$\begin{array}{ccc} S^n \wedge S^m & \longrightarrow & S^m \wedge S^n \\ {\scriptstyle \wedge_{n,m}}\downarrow & & \downarrow{\scriptstyle \wedge_{m,n}} \\ S^{n+m} & \xrightarrow[(-1)^{nm}]{} & S^{n+m} \end{array}$$

where $(-1) : S^n \to S^n$ *is defined by sending* merid (a) *to* merid $(a)^{-1}$.

Proposition 26 (Brunerie 2016, Proposition 4.2.2). *The following diagram commutes.*

$$\begin{array}{ccccc} (S^n \wedge S^m) \wedge S^k & \xrightarrow{(\wedge_{n,m})\wedge 1_{S^k}} & S^{n+m} \wedge S^k & & \\ & & & \searrow^{\wedge_{n+m,k}} & \\ {\scriptstyle \alpha_{S^n,S^m,S^k}}\downarrow & & & & S^{n+n+k} \\ & & & \nearrow_{\wedge_{n,m+k}} & \\ S^n \wedge (S^m \wedge S^k) & \xrightarrow[1_{S^n}\wedge(\wedge_{m,k})]{} & S^n \wedge S^{m+k} & & \end{array}$$

These results are proved by Brunerie using proofs which rely on the symmetric monoidal structure of the smash product. While Proposition 25 should be provable using the partial proof of symmetric monoidality due to van Doorn (2018, Section 4.3), Brunerie's proof of Proposition 26 relies on the pentagon identity which, until now, has been open. Thus, with Theorem 24 we can finally conclude the following.

Theorem 27. *The cup product, as constructed in* Brunerie (2016, *Definition 5.1.6), forms a graded-commutative and associative multiplication* $K(\mathbb{Z}, n) \wedge K(\mathbb{Z}, m) \to K(\mathbb{Z}, n + m)$.

We remark that we have not provided a computer formalisation of the above since there already exist well-developed computer formalisations of the cup product and cohomology rings (Brunerie et al. 2022; Lamiaux et al. 2023; Ljungström and Mörtberg 2024).

6. A Formal Statement of the Heuristic

In a classical setting, the iterated smash product $\bigwedge_{i\leq n} A_i$ can be described as the quotient space $(A_0 \times \cdots \times A_n)/\mathsf{FW}_{i\leq n}(A_i)$ where

$$\mathsf{FW}_{i\leq n}(A_i) := \{(a_0, \ldots, a_n) \mid a_i = \star_{A_i} \text{ for some } i\} \subset A_0 \times \cdots \times A_n$$

It is unclear how to mimic this construction in HoTT in a useful way, as the naive definition forces us to add, in a systematic way, an exponential number of higher coherences. However, the naive definition may still prove useful. Here, by 'naive definition', we mean the disjoint union

$$\mathsf{FS}_{i\leq n}(A_i) := \bigsqcup_{i\leq n} \left(A_0^{(i)} \times \cdots \times A_n^{(i)}\right) \qquad \text{where } A_j^{(i)} = \begin{cases} A_j & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

While the cofibre of the canonical map $\mathsf{FS}_{i\leq n}(A_i) \to A_0 \times \cdots \times A_n$ does not quite give us $\bigwedge_{i\leq n} A_i$, it *does*, however, provide a useful approximation. This is, in fact, precisely the content of our heuristic. We will soon make this precise. Before this, let us give a less set-theoretic definition of $\mathsf{FS}_{i\leq n}(A_i)$ which is easier to work with in HoTT.

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

Definition 28. *Given pointed types $A_0, \dots, A_n$, we define $\mathsf{FS}_{i\leq n}(A_i)$ by induction on n as follows.*

$$\mathsf{FS}_{i\leq n}(A_i) := \begin{cases} 1 & \text{if } n = 0 \\ (\mathsf{FS}_{i\leq n-1}(A_i) \times A_n) + (A_0 \times \cdots \times A_{n-1}) & \text{if } n > 0 \end{cases}$$

Let us, for clarity, explicitly describe the canonical map $\gamma_n : \mathsf{FS}_{i\leq n}(A_i) \to A_0 \times \cdots \times A_n$. It is recursively defined with $\gamma_0 : 1 \to A_0$ picking out the basepoint $\star_{A_0}$ and

$$\gamma_n : (\mathsf{FS}_{i\leq n-1}(A_i) \times A_n) + (A_0 \times \cdots \times A_{n-1}) \to A_0 \times \cdots \times A_n$$

being defined by $\gamma_n = \gamma_n^{(1)} + \gamma_n^{(2)}$ where

$$\gamma_n^{(1)}(x, a_n) = (\gamma_{n-1}(x), a_n)$$
$$\gamma_n^{(2)}(a_0, \dots, a_{n-1}) = (a_0, \dots, a_{n-1}, \star A_n)$$

Definition 29. *Given pointed types $A_0, \dots, A_n$, we define $\widetilde{\bigwedge}_{i\leq n} A_i$ to be the cofibre of $\gamma_n : \mathsf{FS}_{i\leq n}(A_i) \to A_0 \times \cdots \times A_n$, that is, the following pushout.*

$$\begin{array}{ccc} \mathsf{FS}_{i\leq n}(A_i) & \xrightarrow{\gamma_n} & A_0 \times \cdots \times A_n \\ \downarrow & & \downarrow \\ 1 & \longrightarrow & \widetilde{\bigwedge}_{i\leq n} A_i \end{array}$$

We give its constructors explicit names and write

- $\star_{\widetilde{\wedge}}$ *for its basepoint,*
- $\langle a_0, \dots, a_n \rangle$ *for its underlying points,*
- $\mathsf{push}^{(1)}(x) : \langle \gamma_n^{(1)}(x) \rangle = \star_{\widetilde{\wedge}}$ *for the first coherence given by the pushout square,*
- $\mathsf{push}^{(2)}(x) : \langle \gamma_n^{(2)}(x) \rangle = \star_{\widetilde{\wedge}}$ *for the second coherence given by the pushout square.*

It is easy to see that the composition $\mathsf{FS}_{i\leq n}(A_i) \to A_0 \times \cdots \times A_n \to \bigwedge_{i\leq n} A_i$ is null-homotopic. Hence, there is a basepoint preserving inclusion (of constructors) $\iota : \widetilde{\bigwedge}_{i\leq n} A_i \to \bigwedge_{i\leq n} A_i$ s.t. $\iota\langle a_0, \dots, a_n \rangle = \langle a_0, \dots, a_n \rangle$.

We also remark that there is an inclusion $\uparrow : \left(\widetilde{\bigwedge}_{i\leq n-1} A_i\right) \times A_n \to \widetilde{\bigwedge}_{i\leq n} A_i$. It is defined by:

$$\uparrow (\star_{\widetilde{\wedge}}, a_n) := \star_{\widetilde{\wedge}}$$
$$\uparrow (\langle a_0, \dots, a_{n-1} \rangle, a_n) := \langle a_0, \dots, a_n \rangle$$
$$\mathsf{ap}_{\uparrow(-,a_n)}(\mathsf{push}(x)) := \mathsf{push}^{(1)}(x, a_n)$$

Furthermore, $\uparrow$ commutes with ι in the sense that, for $x : \widetilde{\bigwedge}_{i\leq n-1} A_i$ and $a_n : A_n$, we have the following path (in $\bigwedge_{i\leq n} A_i$):

$$\uparrow^{\mathsf{coh}} (x, a_n) \; : \iota\,(\uparrow (x, a_n)) = \langle \iota(x), a_n \rangle$$

One constructs $\uparrow^{\mathsf{coh}} (x, a_n)$ very directly by induction on x. Let us give the construction when x is a point constructor. When $x := \langle a_0, \dots, a_{n-1} \rangle$, the statement holds by refl. When $x := \star_{\widetilde{\wedge}}$, the

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

type of $\uparrow^{\mathsf{coh}}(x, a_n)$ reduces to $\star_\wedge = \langle \star_\wedge, a_n \rangle$ which holds by $\mathsf{push}_\mathsf{r}(a_n)^{-1}$. Hence, we define

$$\uparrow^{\mathsf{coh}}(\langle a_0, \ldots, a_{n-1} \rangle, a_n) := \mathsf{refl} \tag{3}$$

$$\uparrow^{\mathsf{coh}}(\star_{\widetilde{\wedge}}, a_n) := \mathsf{push}_\mathsf{r}(a_n)^{-1} \tag{4}$$

The higher coherence is very straightforward and we omit it for the sake of readability. We are now ready for the key result of the section. It makes precise to what extent $\widetilde{\bigwedge}_{i\leq n} A_i$ approximates $\bigwedge_{i\leq n} A_i$ and may be seen as a formal version of our heuristic. We will later restate the result in a less technical and more self-contained form.

Theorem 30. *Let $f, g : \wedge_{i\leq n} A_i \to B$. Given a homotopy $\widetilde{p}_n : (x : \widetilde{\bigwedge}_{i\leq n} A_i) \to f(\iota(x)) = g(\iota(x))$, there is a homotopy $p_n : (x : \wedge_{i\leq n} A_i) \to f(x) = g(x)$. Furthermore, p_n satisfies:*

$$p_n(\star_\wedge) = \widetilde{p}_n(\star_{\widetilde{\wedge}}) \tag{5}$$

$$p_n\langle a_0, \ldots, a_n \rangle = \widetilde{p}_n\langle a_0, \ldots, a_n \rangle \tag{6}$$

Proof. We proceed by induction on n. When $n = 0$, the map ι is simply the canonical equivalence between the cofibre of the unique pointed map $1 \to_\star A_0$ and A_0 itself, and the theorem is trivial.

Let $n \geq 1$ and let f, g and $\widetilde{p}_n$ be as in the theorem statement. Let us, for clarity, rewrite the domain of f and g as the binary smash product $\left(\bigwedge_{i\leq n-1} A_i\right) \wedge A_n$. In order to construct $p_n : (x : \left(\bigwedge_{i\leq n-1} A_i\right) \wedge A_n) \to f(x) = g(x)$, it suffices, by Lemma 11, to define

(a) a path $p_n(\star_\wedge) : f(\star_\wedge) = g(\star_\wedge)$,
(b) for $a_n : A_n$, homotopies $p_n\langle -, a_n \rangle : (x : \bigwedge_{i\leq n-1} A_i) \to f\langle x, y \rangle = g\langle x, y \rangle$,
(c) For each $a_n : A_n$, a filler of the square

$$\begin{array}{ccc} f(\star_\wedge) & \xrightarrow{p_n(\star_\wedge)} & g(\star_\wedge) \\ {\scriptstyle \mathsf{ap}_f(\mathsf{push}_\mathsf{r}(a_n))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_g(\mathsf{push}_\mathsf{r}(a_n))} \\ f\langle \star_\wedge, a_n \rangle & \xrightarrow[p_n\langle \star_\wedge, a_n \rangle]{} & g\langle \star_\wedge, a_n \rangle \end{array}$$

(d) For each $x : \bigwedge_{i\leq n-1} A_i$, a filler of the square

$$\begin{array}{ccc} f(\star_\wedge) & \xrightarrow{p_n(\star_\wedge)} & g(\star_\wedge) \\ {\scriptstyle \mathsf{ap}_f(\mathsf{push}_\mathsf{l}(x))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_g(\mathsf{push}_\mathsf{l}(x))} \\ f\langle x, \star_{A_n} \rangle & \xrightarrow[p_n\langle x, \star_{A_n} \rangle]{} & g\langle x, \star_{A_n} \rangle \end{array}$$

For (a), we know that $\iota(\star_{\widetilde{\wedge}}) := \star_\wedge$ and thus $\widetilde{p}_n(\star_{\widetilde{\wedge}}) : f(\star_\wedge) = g(\star_\wedge)$. Consequently, we may simply set $p_n(\star_\wedge) := \widetilde{p}_n(\star_{\widetilde{\wedge}})$. Doing so, (5) holds definitionally. For (b), we fix $a_n : A_n$. By the induction hypothesis, it suffices to construct $p_n\langle \iota(x), a_n \rangle : f\langle \iota(x), y \rangle = g\langle \iota(x), y \rangle$ for $x : \widetilde{\bigwedge}_{i\leq n-1} A_i$. We do this by the composite path

$$f\langle \iota(x), a_n \rangle \xrightarrow{\mathsf{ap}_f(\uparrow^{\mathsf{coh}}(x,a_n))^{-1}} f(\iota(\uparrow(x, a_n))) \xrightarrow{\widetilde{p}_n(\uparrow(x,a_n))} g(\iota(\uparrow(x, a_n))) \xrightarrow{\mathsf{ap}_g(\uparrow^{\mathsf{coh}}(x,a_n))} g\langle \iota(x), a_n \rangle$$

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

This completes the construction of $p_n\langle -, a_n\rangle$. Note that, since $p_n\langle -, a_n\rangle$ was constructed using the induction hypothesis, we also get the following from (5):

$$\begin{aligned} p_n\langle \star_\wedge, a_n\rangle &= \mathsf{ap}_f(\uparrow^{\mathsf{coh}}(\star_\wedge, a_n))^{-1} \cdot \widetilde{p}_n(\uparrow(\star_\wedge, a_n)) \cdot \mathsf{ap}_g(\uparrow^{\mathsf{coh}}(\star_\wedge, a_n)) && \text{by (5)} \\ &= \mathsf{ap}_f(\mathsf{push}_\mathsf{r}(a_n)) \cdot p_n(\star_\wedge) \cdot \mathsf{ap}_g(\mathsf{push}_\mathsf{r}(a_n))^{-1} && \text{by (4)} \end{aligned}$$

and the following from (6):

$$\begin{aligned} p_n\langle \vec{a}_{n-1}, a_n\rangle &= \mathsf{ap}_f(\uparrow^{\mathsf{coh}}(\vec{a}_{n-1}, a_n))^{-1} \cdot \widetilde{p}_n(\uparrow(\vec{a}_{n-1}, a_n)) \cdot \mathsf{ap}_g(\uparrow^{\mathsf{coh}}(\vec{a}_{n-1}, a_n)) && \text{by (6)} \\ &= \mathsf{refl} \cdot \widetilde{p}_n(\vec{a}_{n-1}, a_n) \cdot \mathsf{refl} && \text{by (3)} \\ &= \widetilde{p}_n(\vec{a}_{n-1}, a_n) \end{aligned}$$

where $\vec{a}_{n-1}$ is short for $\langle a_0, \dots, a_{n-1}\rangle$. So, to summarise, we have the following two identities for $p\langle -, a_n\rangle$:

$$p_n\langle \star_\wedge, a_n\rangle = \mathsf{ap}_f(\mathsf{push}_\mathsf{r}(a_n)) \cdot p_n(\star_\wedge) \cdot \mathsf{ap}_g(\mathsf{push}_\mathsf{r}(a_n))^{-1} \tag{7}$$

$$p_n\langle \langle a_0, \dots, a_{n-1}\rangle, a_n\rangle = \widetilde{p}_n\langle a_0, \dots, a_{n-1}, a_n\rangle \tag{8}$$

Note that, in particular, the latter identity verifies (6). Let us now turn to (c). By (7), we simply need to fill the square

$$\begin{array}{ccc} f(\star_\wedge) & \xrightarrow{\quad p_n(\star_\wedge) \quad} & g(\star_\wedge) \\ {\scriptstyle \mathsf{ap}_f(\mathsf{push}_\mathsf{r}(a_n))}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_g(\mathsf{push}_\mathsf{r}(a_n))} \\ f\langle \star_\wedge, a_n\rangle & \xrightarrow[\mathsf{ap}_f(\mathsf{push}_\mathsf{r}(a_n)) \cdot p_n(\star_\wedge) \cdot \mathsf{ap}_g(\mathsf{push}_\mathsf{r}(a_n))^{-1}]{} & g\langle \star_\wedge, a_n\rangle \end{array}$$

which is entirely trivial by definition of path composition.

Finally, let us provide the data asked for in (d). Filling the square is equivalent to constructing, a homotopy $(x : \bigwedge_{i\le n-1} A_i) \to \mathsf{left}(x) = \mathsf{right}(x)$ where $\mathsf{left}, \mathsf{right} : \bigwedge_{i\le n-1} A_i \to f(\star_\wedge) = g(\star_\wedge)$ are defined by:

$$\begin{aligned} \mathsf{left}(x) &:= p_n(\star_\wedge) \\ \mathsf{right}(x) &:= \mathsf{ap}_f(\mathsf{push}_\mathsf{l}(x))^{-1} \cdot p_n\langle x, \star_{A_n}\rangle \cdot \mathsf{ap}_g(\mathsf{push}_\mathsf{l}(x)) \end{aligned}$$

The codomain $f(\star_\wedge) = g(\star_\wedge)$ is homogeneous (with $\mathsf{left}(\star_\wedge)$ as basepoint) and thus Lemma 16 applies. Hence, we only need to show that

$$\mathsf{left}\langle a_0, \dots, a_{n-1}\rangle = \mathsf{right}\langle a_0, \dots, a_{n-1}\rangle$$

Let us rewrite $p_n\langle -, \star_{A_n}\rangle$ according to (8) and transform the above back into square form. The problem is to fill the square

$$\begin{array}{ccc} f(\star_\wedge) & \xrightarrow{\quad \widetilde{p}_n(\star_\wedge) \quad} & g(\star_\wedge) \\ {\scriptstyle \mathsf{ap}_f(\mathsf{push}_\mathsf{l}\langle a_0,\dots,a_{n-1}\rangle)}\uparrow & & \uparrow{\scriptstyle \mathsf{ap}_g(\mathsf{push}_\mathsf{l}\langle a_0,\dots,a_{n-1}\rangle)} \\ f\langle\langle a_0, \dots, a_{n-1}\rangle, \star_{A_n}\rangle & \xrightarrow[\widetilde{p}_n\langle a_0,\dots,a_{n-1},\star_{A_n}\rangle]{} & g\langle\langle a_0, \dots, a_{n-1}\rangle, \star_{A_n}\rangle \end{array}$$

Such a square is given precisely by $\mathsf{ap}_{\widetilde{p}_n}(\mathsf{push}^{(2)}(a_0, \dots, a_{n-1}))$, and we are done. □

Let us restate the result in a more compact and self-contained form.

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

Corollary 31. *Let $f, g : \bigwedge_{i\leq n} A_i \to B$. If f and g agree on $\iota : \widetilde{\bigwedge}_{i\leq n} A_i \to \bigwedge_{i\leq n} A_i$, that is if $f \circ \iota = g \circ \iota$, then $f = g$.*

For completeness, let us also state the analogous result for pointed maps.

Corollary 32. *Let $f, g : \bigwedge_{i\leq n} A_i \to_\star B$. If $f \circ \iota = g \circ \iota$ as pointed map, then we obtain an equality of pointed maps $f = g$.*

7. Conclusions and Future Work

Let us summarise the key contributions of this paper. First and foremost, we have shown in Sections 3 through 5 that the smash product forms a 1-coherent symmetric monoidal product on the wild category of pointed types (Theorem 24). This fills a long-standing and rather troublesome gap in the HoTT literature. In particular, it implies the correctness of previous work which relies on this result – perhaps most notably, that of Brunerie (2016) and van Doorn (2018).

Second, we have presented a new method for reasoning about smash products in HoTT. This was first done by introducing, in Section 4, an informal heuristic which was used heavily leading up to Theorem 24. We later provided an attempt at a formal version of the heuristic in Corollaries 31 and 32. The message of these results is simple: functions defined over smash products in HoTT are not much harder to deal with than those defined over ordinary Cartesian products. This contradicts a commonly held view that smash products in HoTT become unworkable in higher dimensions.

In addition to salvaging already existing work relying on unproved results about smash products, we also hope that the results presented here will be useful in the further development of synthetic homotopy theory in HoTT. In particular, Theorem 24 is not the only consequence of Propositions 25 and 26. These propositions capture two important properties of the equivalence $\wedge_{n,m} : S^n \wedge S^m \simeq S^{n+m}$. For instance, the special case when $n = m = 1$ was implicitly used by Ljungström and Mörtberg (2023, Section VI.) in order to provide a simplified version of Brunerie's proof of $\pi_4(S^3) \cong \mathbb{Z}/2\mathbb{Z}$. More generally, $\wedge_{n,m} : S^n \wedge S^m \simeq S^{n+m}$ gives rise to the *Whitehead product* (Whitehead, 1941). This operation was originally defined in HoTT by Brunerie (2016, Definition 3.3.3.) using joins of spheres, but we can also give it a rather compact construction in terms of $\wedge_{n,m}$: the Whitehead product $[f, g]$ of two maps $f : S^{n+1} \to_\star A$ and $g : S^{m+1} \to_\star A$ can be viewed as the map defined by the composition:

$$S^{n+m} \xrightarrow{\wedge_{n,m}^{-1}} (S^n \wedge S^m) \xrightarrow{\mathsf{comm}} \Omega(S^{n+1} \vee S^{m+1}) \xrightarrow{\Omega(f\vee g)} \Omega A$$

where comm sends $\langle x, y\rangle$ to $\sigma_l^{-1}(x) \cdot \sigma_r(y) \cdot \sigma_l(x) \cdot \sigma_r^{-1}(y)$, where $\sigma : S^k \to \Omega S^{k+1}$ is defined by $\sigma(x) = \mathsf{merid}(x) \cdot \mathsf{merid}(\mathsf{north})^{-1}$ and the subscripts l and r indicate in which component of $S^{n+1} \vee S^{m+1}$ the loop takes place. This induces a map on homotopy groups

$$\pi_{n+1}(A) \times \pi_{m+1}(A) \to \pi_{n+m}(\Omega(A)) \xrightarrow{\sim} \pi_{n+m+1}(A)$$

which, classically, turns $\pi_\bullet(A)$ into a graded quasi-Lie algebra (Uehara and Massey 1957). This fact is still open in HoTT, but we conjecture that Propositions 25 and 26 will play a crucial role in a prospective proof. More concretely, as $\wedge_{n,m}$ is used in the construction of this map, Proposition 25 and 26 should, respectively, play important roles in prospective proofs of graded symmetry and the Jacobi identity.

On a related note, the heuristic presented in Section 4 and the related results in Section 6 are reminiscent of results from the study of polyhedral products (see e.g., Bahri et al. 2019; Theriault 2018). This is a field of mathematics which, very coarsely speaking, explores and generalises the mediation between iterated wedge sums and iterated Cartesian products. It is unclear whether

https://doi.org/10.1017/S0960129524000318 Published online by Cambridge University Press

this topic can be studied in a meaningful way through the lens of HoTT, but exploring this would certainly amount to an interesting continuation of the work we have presented here.

Another interesting question which we have not covered here is whether our heuristic can be used to show the primary technical gap in the thesis of van Doorn (2018), namely that the equivalence (1) is a pointed natural equivalence. This would show that the universe of pointed types is a closed monoidal (wild) category. It is not obvious that the heuristic, as it is phrased in this paper, is suitable for this problem. Nevertheless, it is not unlikely that techniques similar to those used here still apply. We leave this for future work.

Acknowledgements. The author would like to thank Evan Cavallo for several fruitful discussions, for his prior unpublished formalisations of smash products in Cubical Agda which have been useful as a source of inspiration for the formalisation of this paper, and, of course, for his contribution of the incredibly useful Lemma 14 to the homotopy type theory literature. The author would also like to thank Dan Petersen and an anonymous reviewer for their insightful comments on earlier versions of this paper.

Funding statement. This paper is based on research supported by the Swedish Research Council (Vetenskapsrådet) under Grant No. 2019-04545.

Competing interests. The author declares none.

Note

1 By 'computes to' we do not mean 'normalises in Agda to'. We mean 'compute' in the manual sense, that is, by tracing H_0 and H_1 on the point and path constructors involved. Direct normalisation in Agda produces rather large and unmanageable terms. However, using Agda to normalise the terms in a more controlled manner (i.e., step-by-step) is very useful, as a sanity check, for inspecting the action of H_0 and H_1 on the path constructors involved.

References

Bahri, A., Bendersky, M. and Cohen, F. R. (2019). Polyhedral products and features of their homotopy theory. In Miller, H. (ed.) *Handbook of Homotopy Theory* (1st ed.). New York: CRC Press, pp. 103–144.

Brunerie, G. (2016). *On the Homotopy Groups of Spheres in Homotopy Type Theory*. PhD thesis, Université Nice Sophia Antipolis.

Brunerie, G. (2018). Computer-generated proofs for the monoidal structure of the smash product. *Homotopy Type Theory Electronic Seminar Talks*.

Brunerie, G., Ljungström, A. and Mörtberg, A. (2022). Synthetic integral cohomology in cubical agda. In Manea, F. and Simpson, A.(eds.) *30th EACSL Annual Conference on Computer Science Logic (CSL 2022), Dagstuhl, Germany*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, vol. **216**, 11:1–11:19, *Leibniz International Proceedings in Informatics (LIPIcs)*.

Buchholtz, U., Christensen, J. D., Flaten, J. G. T. and Rijke, E. (2023). Central h-spaces and banded types, arXiv: 2301.02636.

Capriotti, P. and Kraus, N. (2017). Univalent higher categories via complete semi-segal types. *Proceedings of the ACM on Programming Languages* **2** (POPL) 44:1–44:29.

Cavallo, E. (2021a). *Higher Inductive Types and Internal Parametricity for Cubical Type Theory*. Phd thesis, Carnegie Mellon University.

Cavallo, E. (2021b). Pointed functions into a homogeneous type are equal as soon as they are equal as unpointed functions. Agda formalization, part of the cubical library. Available at https://agda.github.io/cubical/Cubical. Foundations.Pointed. Homogeneous.html#1616.

Cavallo, E. and Harper, R. (2020). Internal parametricity for cubical type theory. In Fernández, M. and Muscholl, A.(eds.) *Leibniz International Proceedings in Informatics (LIPIcs). 28th EACSL Annual Conference on Computer Science Logic (CSL 2020), Dagstuhl, Germany*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, vol. **152**, 13:1–13:17.

Kraus, N. (2013). The Truncation Map $|\_|:N\to\|N\|$ is nearly Invertible. Blog post at https://homotopytypetheory.org/2013/10/28/

Lamiaux, T., Ljungström, A. and Mörtberg, A. (2023). Computing cohomology rings in Cubical Agda. In: *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2023, New York, NY, USA*. Association for Computing Machinery, 239–252.

Ljungström, A. and Mörtberg, A. (2023). Formalizing $\pi_4(S^3) \cong \mathbb{Z}/2\mathbb{Z}$ and computing a Brunerie number in Cubical Agda. In: *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), Los Alamitos, CA, USA*. IEEE Computer Society, 1–13.

Ljungström, A. and Mörtberg, A. (2024). Computational synthetic cohomology theory in homotopy type theory, arXiv: 2401.16336.

Pujet, L. and Mörtberg, A. (2020). Cubical synthetic homotopy theory. In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New York, NY, USA*. Association for Computing Machinery, 158–171.

The Univalent Foundations Program. (2013). Homotopy Type Theory: Univalent Foundations of Mathematics. Self-published, Institute for Advanced Study.

Theriault, S. (2018). The dual polyhedral product, cocategory and nilpotence. *Advances in Mathematics* **340** 138–192.

Uehara, H. and Massey, W. S. (1957). *The Jacobi Identity for Whitehead Products*, Princeton, Princeton University Press, 361–377.

van Doorn, F. (2018). *On the Formalization of Higher Inductive Types and Synthetic Homotopy Theory*. PhD thesis, Carnegie Mellon University.

Vezzosi, A., Mörtberg, A. and Abel, A. (2021). Cubical agda: a dependently typed programming language with univalence and higher inductive types. *Journal of Functional Programming* **31** e8.

Whitehead, J. H. C. (1941). On adding relations to homotopy groups. *Annals of Mathematics* **42** (2) 409–428.

Cite this article: Ljungström A. (2024). Symmetric monoidal smash products in homotopy type theory. *Mathematical Structures in Computer Science*. https://doi.org/10.1017/S0960129524000318

Paper IV

Cellular Methods in Homotopy Type Theory

Axel Ljungström and Loïc Pujet
Department of Mathematics
Stockholm University
Email: axel.ljungstrom@math.su.se and loic@pujet.fr

***Abstract*—In classical mathematics, a CW complex is a topological space which can be built up inductively by gluing together cells of increasing dimension. Due to their good categorical properties, CW complexes have become the main object of interest in algebraic topology. Although their quasi-combinatorial nature suggests that a constructive treatment is possible, there seems to be little literature on the subject – perhaps because of the important role played by the axiom of choice in the classical theory of CW complexes.**

In this paper, we present a *synthetic* and *constructive* account of the theory of CW complexes in homotopy type theory. Most notably, we prove a finitary version of the cellular approximation theorem, which allows us to construct a theory of cellular homology without needing the axiom of choice or relying on a pre-existing notion of homology. We prove that our cellular homology is functorial and that it satisfies a finitary version of the Eilenberg-Steenrod axioms. Last but not least, we give a constructive proof of the Hurewicz theorem for CW complexes, which relates the first non-vanishing homotopy group of a CW complex with the corresponding homology group. All key theorems presented in this paper have been mechanised in Cubical Agda.

I. Introduction

Homotopy type theory (HoTT) is an extension of intensional type theory which treats types and equalities from a homotopical perspective [1]. It provides a synthetic framework for reasoning about spaces and their homotopy invariants, and has been successfully used to formalise a number of results from algebraic topology [2]. In this paper, we present a development of the theory of CW complexes in HoTT, including cornerstone results such as the cellular approximation theorem, cellular homology, and the Hurewicz theorem.

A remarkable aspect of homotopy type theory is that it is fully constructive by default. While it is possible to postulate classical reasoning principles such as the law of excluded middle or the axiom of choice, working in plain constructive HoTT results in theorems that are strictly more general – in particular, they become valid in every ∞-topos [3], [4]. Thus, the developments presented in this paper effectively show that cellular methods are available in this very general setting.

A. Outline and contributions

In Section II, we outline the basic definitions from HoTT which we will need for our development, with special emphasis on homotopy pushouts and truncations.

In Section III, we develop our constructive theory of CW complexes. Our main contributions are a construction of the pushout of two cellular maps (Definition 10 and Proposition 11), and a constructive treatment of the cellular approximation theorem for maps and homotopies (Theorems 15 and 22). We also define various categories which, to a varying degree, capture the idea of a cellular space in HoTT, and we study the relations between these categories.

In Section IV, we apply our results to the construction of homology theories for our categories of cellular spaces. We associate homology groups to every CW complex, and we show that this association is functorial and homotopy invariant using our freshly proved cellular approximation theorems. Finally, we verify that our homology functors satisfy the Eilenberg–Steenrod axioms. We remark that most proofs and constructions in this section can be interpreted in the setting of cellular *co*homology. We simply chose to focus on homology because it constituted an open problem.

In Section V, we prove the Hurewicz theorem for our homology theory. To this end, we prove a special case of the so-called *CW approximation theorem* which shows that, for CW complexes, the usual definition of an n-connected type in HoTT coincides with the classical definition of an n-connected CW complex (Corollary 48). We emphasise that the approximation theorems that underlie this work are results whose classical proofs tend to be inherently non-constructive. We hope that our constructive proofs will interest also the logician who is not necessarily well-versed in HoTT.

The main theorems and constructions in this paper have been mechanised in Cubical Agda; the proofs are available at https://github.com/caripoulet974/cellular_methods.

B. Related work

Our definition of CW complexes is based on the definition given by Buchholtz and Favonia in their work on cellular cohomology [5]. We develop the theory quite a bit further: we define cellular maps and cellular homotopies, and we prove their appurtenant approximation theorems. This lets us prove that cellular (co)homology is functorial without having to rely on a pre-existing (co)homology theory.[1] This is especially valuable for the construction of cellular *homology*, as there is no pre-existing notion of homology that has been developed to the same extent as Eilenberg–MacLane cohomology.

Nevertheless, there is work by Graham on developing synthetic homology in HoTT using the Eilenberg–MacLane

[1]Functoriality and homotopy invariance are not proved explicitly by Buchholtz and Favonia, but they follow from their comparison between cellular and Eilenberg–MacLane cohomology.

979-8-3503-3587-3/23/$31.00

prespectrum [6]. The resulting functor is expected to satisfy the Eilenberg-Steenrod axioms, but the additivity axiom remains an open question. Additionally, Christensen and Scoccola gave a proof of the Hurewicz theorem for this definition of homology [7]. We emphasise, however, that the proof given of the Hurewicz theorem in this paper is vastly different in its approach and that it concerns a different definition of homology compared to the one used by Christensen and Scoccola.

II. Background

In this section, we will give a brief introduction to the elementary constructions and facts from HoTT which are used in this paper. We assume some level of familiarity with HoTT and refer the reader to the *HoTT Book* [1] whose notation we, for the most part, stay consistent with in this paper. Another excellent introduction is [8].

a) Π-types: we borrow `Agda` notation and often write $(a : A) \to B\,a$ instead of $\Pi_{x:a} B\,a$. Non-dependent Π-types are simply denoted $A \to B$. We may still use the traditional Π-notation when convenient.

b) Path types: given $x, y : A$, we write $x = y$ for their identity type. We refer to elements of this type as *paths*, and we write $\mathsf{refl}_x : x = x$ for the constant path. The *path induction* rule states that dependent functions $((y, p) : \Sigma_{y:A}(x = y)) \to B(y, p)$ are determined by their action on (x, refl_A).

c) Universes and pointed types: we write Type for the universe of types (at some implicit universe level) and $\mathsf{Type}_\star$ for the universe of pointed types, i.e. the type of pairs $(A, \star_A)$ where $A : \mathsf{Type}$ and $\star_A : A$. For simplicity, we generally write 'A is a pointed type' and leave the basepoint implicit. We always use the notation $\star_A$ for basepoints.

d) Pointed functions: given two pointed types A and B, the type of pointed functions $A \to_\star B$ is the type of pairs $(f, \star_f)$ where $f : A \to B$ is a function and $\star_f : f\,\star_A = \star_B$. We often simply write $f : A \to_\star B$ and leave $\star_f$ implicit.

e) Fibres, equivalences and univalence: we write $\mathsf{fib}_f(b)$ for the fibre of a function $f : A \to B$ over a point $b : B$. That is, $\mathsf{fib}_f(b) := \Sigma_{a:A}(f\,a = b)$. A function $f : A \to B$ whose fibres are contractible (i.e. pointed by a unique point) is called an equivalence. We write $f : A \simeq B$, and $f^{-1} : B \simeq A$ for the induced inverse. The identity $\mathsf{id} : A \to A$ is always an equivalence; the univalence axiom says precisely that the function $A = B \to A \simeq B$ defined by path induction by sending refl_A to the identity equivalence is itself an equivalence.

f) The Unit type and the empty type: we write $\mathbb{1}$ for the unit type, i.e. the inductive type with one unique constructor $\star_\mathbb{1} : \mathbb{1}$, and $\bot$ for the empty type.

A. Pushouts

Besides inductive types, we will also make heavy use of *higher inductive types* (HITs), which include path constructors in addition to point constructors. One of the arguably most important HITs in HoTT is the *pushout* of a span.

Definition 1 (Pushouts)**.** *Given a span $Y \xleftarrow{f} X \xrightarrow{g} Z$, we define its pushout (as indicated in the diagram below) to be the HIT generated by point constructors $\mathsf{inl} : Y \to Y \sqcup^X Z$ and $\mathsf{inr} : Z \to Y \sqcup^X Z$, as well as a higher constructor $\mathsf{push} : (x : X) \to \mathsf{inl}\,(f\,x) = \mathsf{inr}\,(g\,x)$.*

$$\begin{array}{ccc} X & \xrightarrow{g} & Z \\ {\scriptstyle f}\downarrow & \ulcorner & \downarrow \\ Y & \to & Y \sqcup^X Z \end{array}$$

Given a span S, we may also write $\mathsf{PO}\,S$ for its pushout. We always take $Y \sqcup^X Z$ to be pointed by $\mathsf{inl}\,\star_Y$ (assuming Y is pointed). Pushouts will allow us to define most spaces of interest in this paper. The following three instances of pushouts are especially important for us.

a) Cofibres: we define the *cofibre* of a map $f : X \to Y$, denoted C_f, by $C_f := \mathbb{1} \sqcup^X Y$. To stay consistent with the existing literature, we write cfcod instead of $\mathsf{inr} : Y \to C_f$.

b) Wedge sums: given a dependent family of pointed types $A : I \to \mathsf{Type}_\star$, we define its *wedge sum*, denoted $\bigvee_{i:I}(A\,i)$, to be the cofibre of the obvious map $I \to \Sigma_{i:I}(A\,i)$. When we specifically wish to reason about the binary wedge sum of two pointed types A and B, we may alternatively define these by $A \vee B = A \sqcup^\mathbb{1} B$.

c) Suspensions: we define the *suspension* of a type X, denoted ΣX, by $\Sigma X := \mathbb{1} \sqcup^X \mathbb{1}$. As is standard practice, we use north and south to refer to $\mathsf{inl}\star$ and $\mathsf{inr}\star$ respectively, and we write $\mathsf{merid} : X \to \mathsf{north} = \mathsf{south}$ instead of push. Suspensions allow us to define spheres inductively by setting $\mathbb{S}^{-1} := \bot$, i.e the empty type, and $\mathbb{S}^n := \Sigma\mathbb{S}^{n-1}$ for $n > -1$.

Let us state two elementary lemmas concerning pushouts which will be useful later. The following lemma is proved using standard pushout-pasting arguments.

Lemma 2. *Let $f : A \to B$ with A and B. We have*

1) $C_{(\mathsf{cfcod}:B\to C_f)} \simeq \Sigma A$,
2) $C_f \simeq \Sigma A \vee B$ *if B is pointed and f is constant at $\star_B$.*

We will also need the 3×3-*lemma* – an incredibly useful result which was first introduced in the HoTT literature by Brunerie [2, Lemma 1.8.3] whose notation we also borrow. Let A_{ij} be a commutative grid of types indexed by $I = \{0, 2, 4\}$ as in the diagram to the right. The 3×3-lemma says that *taking pushouts over rows and then columns is equivalent to taking pushouts over columns and then rows.* Let us unwrap this statement. Let $A_{\bullet i}$ and $A_{i\bullet}$ denote, respectively, the pushout along column i and the pushout along row i. That is, let $A_{\bullet i} := A_{0i} \sqcup^{A_{2i}} A_{4i}$ and $A_{i\bullet} := A_{i0} \sqcup^{A_{i2}} A_{i4}$. We produce a span $(A_{\bullet i})_{i\in I} := (A_{\bullet 0} \xleftarrow{f_{01} \sqcup^{f_{21}} f_{41}} A_{\bullet 2} \xrightarrow{f_{03} \sqcup^{f_{23}} f_{43}} A_{\bullet 4})$. We define $(A_{i\bullet})_{i\in I}$ similarly. Let $A_{\square\bullet} := \mathsf{PO}\,(A_{\bullet i})_{i\in I}$ and $A_{\bullet\square} := \mathsf{PO}\,(A_{i\bullet})_{i\in I}$.

$$\begin{array}{ccccc} A_{00} & \xleftarrow{f_{01}} & A_{02} & \xrightarrow{f_{03}} & A_{04} \\ {\scriptstyle f_{10}}\uparrow & & {\scriptstyle f_{12}}\uparrow & & \uparrow{\scriptstyle f_{14}} \\ A_{20} & \xleftarrow[f_{21}]{} & A_{22} & \xrightarrow{f_{23}} & A_{24} \\ {\scriptstyle f_{30}}\downarrow & & \downarrow{\scriptstyle f_{32}} & & \downarrow{\scriptstyle f_{34}} \\ A_{40} & \xleftarrow[f_{41}]{} & A_{42} & \xrightarrow[f_{43}]{} & A_{44} \end{array}$$

Lemma 3 (3×3-lemma)**.** $A_{\square\bullet} \simeq A_{\bullet\square}$

B. Truncations

We say that a type A is a (-2)-type if it is contractible (i.e. if it consists of a unique element) and, inductively, that it is an $(n+1)$-type if any identity type $x =_A y$ over A is an n-type.

We refer to (-1)-types (i.e. types with at most one element) as *propositions* and 0-types (i.e. types which satisfy UIP) as *sets*. In HoTT, any type A can be turned into an n-type by forming its n-*truncation*, denoted $\|A\|_n$. This type is defined as a HIT with a point constructor $|-| : A \to \|A\|_n$ and a few additional constructors forcing $\|A\|_n$ to be an n-type. A detailed implementation can be found in [1, Section 7.3] but will not be needed here; all we shall need is the elimination property of the n-truncation which says that any (possibly dependent) function $f : (x : \|A\|_n) \to B\,x$ is uniquely determined by its action on canonical elements whenever B is a family of n-types. That is, the map $((x : \|A\|_n) \to B\,x) \to ((a : A) \to B\,|a|)$ is an equivalence. The philosophy of the elimination principle is that whenever we are trying to construct an element of a n-type, we may use $\|A\|_n$ and A interchangeably.

Truncations are crucial for internalising several notions and constructions from traditional mathematics in HoTT:

a) Choice: We say that a set A satisfies choice, or equivalently that A is a projective set, if the canonical map $\|(a : A) \to B\,a\|_{-1} \to ((a : A) \to \|B\,a\|_{-1})$ is an equivalence. The statement that every set is projective is a strong form of the axiom of choice, which is not available in constructive HoTT. On the other hand, the set $\mathsf{Fin}(n) := \Sigma_{i:\mathbb{N}}(i < n)$ is constructively projective, meaning that we do not need any axiom to get choice for families indexed over a finite set. We write pSet for the type of all projective sets.

b) Existence: we define $\exists_{a:A}(B\,a) := \|\Sigma_{a:A}(B\,a)\|_{-1}$ to encode the notion of propositional existence, which is also called mere existence. When this type is inhabited, we say that there *merely* exists an element $a : A$ such that $B\,a$ holds.

c) Homotopy groups: given a pointed type A and an integer $n \geq 1$, we define the nth homotopy group of A by $\pi_n(A) := \|\mathbb{S}^n \to_\star A\|_0$. This type turns out to have a group structure, which is abelian for $n \geq 2$. The construction is functorial via post-composition; for a map $f : A \to_\star B$, we write $\pi_n(f) : \pi_n(A) \to \pi_n(B)$ for the functorial action. The construction is also invariant under n-truncation: the canonical map $\pi_n(A) \to \pi_n(\|A\|_n)$ is an isomorphism of groups.

d) Connectedness: we say that a type A is n-connected if $\|A\|_n$ is contractible. A function $f : A \to B$ is said to be n-connected when all of its fibres are. It is an easy fact that if f is n-connected, it induces an equivalence on truncations $\|A\|_n \simeq \|B\|_n$ (and thus also on π_n).

Another important fact about n-truncations is that they commute with path types, in the sense that for any $x, y : A$, the canonical map $\|x = y\|_n \to |x| =_{\|A\|_{n+1}} |y|$ is an equivalence [1, Theorem 7.3.12]. This principle, together with the elimination principle for $\mathbb{S}^n$, gives rise to the following elementary 'choice principle' for $\mathbb{S}^n$.

Lemma 4. *Given a dependent type $A : \mathbb{S}^n \to \mathsf{Type}$, there exists a function*

$$\mathsf{choose}_{\mathbb{S}^n} : ((x : \mathbb{S}^n) \to \|A\,x\|_{n-1}) \to \|(x : \mathbb{S}^n) \to A\,x\|_{-1}$$

III. CW Complexes in HoTT

A *CW complex* is a space which is constructed by an iterative process of attaching cells: start with a collection of points (0-dimensional cells), then connect some of them using 1-dimensional line segments to obtain a multigraph, then glue a collection of 2-dimensional discs to the multigraph, then 3-dimensional cells, and so on. This iterative construction is captured by the following definition in type theory:

Definition 5. *A projective **CW structure** is a sequence of types $(X_{-1} \xrightarrow{\iota_{-1}} X_0 \xrightarrow{\iota_0} X_1 \xrightarrow{\iota_1} \dots)$ together with a **cardinality** function $c^X_{(-)} : \mathbb{N} \to \mathsf{pSet}$ and accompanying **attaching maps** $\alpha^X_i : \mathbb{S}^i \times c^X_{i+1} \to X_i$ satisfying the following two conditions.*

(A1) $X_{-1} \simeq \emptyset$,

(A2) *for each $i \geq -1$, the square to the right is a pushout.*

$$\begin{array}{ccc} \mathbb{S}^i \times c^X_{i+1} & \xrightarrow{\mathsf{snd}} & c^X_{i+1} \\ {\scriptstyle \alpha^X_i}\downarrow & \ulcorner & \downarrow \\ X_i & \xrightarrow[\iota_i]{} & X_{i+1} \end{array}$$

The cardinality function indicates 'how many' cells should be added at every stage, and the attaching maps α_i explain how the boundary of each $(i+1)$-dimensional cell is attached to the i-skeleton X_i. Finally, the pushout condition states that X_{i+1} is obtained from X_i by gluing cones along these boundaries, as in the 'hub and spokes' construction from [1, Section 6.7]. Since we will be using them a lot throughout the paper, we introduce special notation for the pushout constructors of X_{i+1}: given $x : X_i$, $y : c^X_{i+1}$ and $s : \mathbb{S}^i$, we write

- $\iota_i\,x : X_{i+1}$ (as indicated in the diagram),
- $\mathsf{cell}\,y : X_{i+1}$, and
- $\mathsf{glue}(s, y)$ for the path $\iota_i\,(\alpha^X_i\,(s, y)) = \mathsf{cell}\,y$.

We will often denote CW structures simply by $X_* : \mathsf{CWstr}$ and leave ι_*, c^X_* and α^X_* implicit. We remark that we assume our sets of cells to be projective, i.e. we assume that they all satisfy choice. In constructive HoTT, this includes at least the CW structures that have a finite number of cells in each dimension, which are already sufficient to represent many spaces of interest. In presence of the full axiom of choice ($\mathsf{AC}_{\infty,-1}$ in the notation of [1]), all sets are projective and thus our definition works with arbitrary sets of cells.

Disclaimer 6. *In this paper, we will restrict ourselves to CW structures of finite type, i.e. we take their cardinality function to be $c^X_n := \mathsf{Fin}(k_n)$ for some $k_n : \mathbb{N}$. We make this restriction for three reasons:*

- *it is the definition used in the closely related paper on cellular cohomology by Buchholtz and Favonia [5],*
- *it makes the presentation of some results more direct, as we can avoid the general theory of projective sets, and*
- *for computational reasons, it is the version used in our formalisation.*

We remark, however, that all of our methods and results (apart from Proposition 43, which has to be reformulated slightly) can be straightforwardly generalised to the projective version of CW structures presented in Definition 5. Although we leave this generalisation for a future extended version of this paper, we wish to thank an anonymous reviewer for calling our attention to it.

We will write $\left|c_n^X\right|$ for the cardinality of c_n^X and may abuse notation by writing e.g. $(c_n^X - 1)$ for the type $\mathsf{Fin}(\left|c_n^X\right| - 1)$.

Definition 7. *Given a CW structure X_*, we define its **sequential colimit** to be the HIT X_∞ which consists of*

- *for every $x : X_n$, a point $[x]_n : X_\infty$,*
- *for every $x : X_n$, a path $\mathsf{push}\, x : [x]_n = [\iota_n\, x]_{n+1}$.*

We sometimes write $\iota_\infty\, x$ for $[x]_n$ when n is clear from context.

Definition 8. *We say that a type A is a **CW complex** if there merely exists some CW structure X_* such that A is the sequential colimit of X_*. Formally, we define*

$$\mathsf{CW} := \Sigma(A : \mathsf{Type})\,.\,\exists(X_* : \mathsf{CWstr})\,.\,X_\infty \simeq A.$$

In the rest of this paper, we will develop the theory of CW complexes, building up to a definition of cellular homology and a proof of the Hurewicz theorem. In doing so, we will take advantage of the fact that every CW complex is presented by a CW structure, which allows us to construct most properties and objects by induction on the dimension. Thus, our first endeavour shall be the development of a working theory of CW structures, starting with their natural notion of maps.

Definition 9. *A cellular map from X_* to Y_* is a pair (f_*, h_*) where $f_i : X_i \to Y_i$ and $h_i : (x : X_i) \to f_{i+1}(\iota_i\, x) = \iota_i(f_i\, x)$, as depicted on the diagram below:*

$$\begin{array}{ccccccc} X_{-1} & \longrightarrow & X_0 & \longrightarrow & X_1 & \longrightarrow & \dots \\ {\scriptstyle f_{-1}}\downarrow & \swarrow^{h_0} & \downarrow{\scriptstyle f_0} & \swarrow^{h_1} & \downarrow{\scriptstyle f_1} & & \\ Y_{-1} & \longrightarrow & Y_0 & \longrightarrow & Y_1 & \longrightarrow & \dots \end{array}$$

In simpler terms, a cellular map is a map which respects the dimensions, in the sense that it sends the n-dimensional skeleton of the source to the n-dimensional skeleton of the target. For simplicity, we generally write $f_* : X_* \to Y_*$ for a cellular map, leaving h_* implicit. Every cellular map from X_* to Y_* gives rise to a function f_∞ between their colimits:

$$\begin{array}{lcl} f_\infty : X_\infty \to Y_\infty & & \\ f_\infty\, [x]_n & := & [f_n\, x]_n \\ \mathsf{ap}_{f_\infty}(\mathsf{push}\, x) & := & \mathsf{push}\,(f_n\, x) \cdot \mathsf{ap}_{[-]_{n+1}}(h_n\, x). \end{array}$$

The identity can be presented as a cellular map, and the obvious composition of two cellular maps yields the composition of the colimits. Furthermore, this composition operation is associative and unital. All this data assembles into a category CWstr whose objects are CW structures, and whose mapping sets are given by (the set truncation of) cellular maps. The colimit operation then defines a functor from CWstr to the category of CW complexes and set-truncated ordinary maps, which we denote by $\mathsf{Ho}(\mathsf{CW})$.[2]

The interplay between $\mathsf{Ho}(\mathsf{CW})$ and CWstr will be a recurring theme of this paper: our main object of interest is the category $\mathsf{Ho}(\mathsf{CW})$, but we find that it does not offer sufficient control over the objects and the morphisms. Instead, we define all of our constructions in CWstr, taking advantage of the inductive description of spaces and maps, before transporting them to $\mathsf{Ho}(\mathsf{CW})$. Our main tool for this transport step will be the *cellular approximation theorem*, which provides a partial inverse to the colimit functor.

[2]We reserve the name CW for the wild ∞-category of CW complexes, which we will not use in this paper. All of our categories are 1-categories, and we do not assume them to be univalent by default.

A. Pushouts of CW structures

Before embarking on the proof of the cellular approximation theorem, it might be good to look at a concrete example of a CW structure, to help the reader build intuition. For this purpose, we shall explain how to construct the homotopy pushout of two cellular maps. This construction will play an important role later down the line, as the definition of a homology theory requires our category of CW complexes to be equipped with pushouts.

Definition 10. *Let X_*, Y_*, Z_* be three CW structures, and $(f_*, h_*) : X_* \to Y_*$ and $(g_*, k_*) : X_* \to Z_*$ be two cellular maps. We define the pushout of the span $Y_* \xleftarrow{f_*} X_* \xrightarrow{g_*} Z_*$ to be the CW structure $(Y \sqcup^X Z)_*$ defined by letting $(Y \sqcup^X Z)_i$ be the pushout $Y_i \sqcup^{X_{i-1}} Z_i$, i.e. the pushout of the span $Y_i \xleftarrow{\iota_{i-1} \circ f_{i-1}} X_{i-1} \xrightarrow{\iota_{i-1} \circ g_{i-1}} Z_i$.*

***Inclusions:** The inclusions $(Y \sqcup^X Z)_i \to (Y \sqcup^X Z)_{i+1}$ are the obvious maps induced by the corresponding inclusions for X_*, Y_* and Z_*.*

***Cells:** We define the cell cardinalities $c_*^{Y \sqcup^X Z}$ in terms of those of X_*, Y_* and Z_* by letting $c_i^{Y \sqcup^X Z} = c_i^Y + c_i^Z + c_{i-1}^X$.*

***Attaching maps:** Finally, we define the attaching maps as*

$$\begin{array}{l} \alpha_i^{Y \sqcup^X Z} \;:\; \sum_{c \in \{c_{i+1}^Y, c_{i+1}^Z, c_i^X\}} \mathbb{S}^i \times c \;\to\; Y_i \sqcup^{X_{i-1}} Z_i \\ \alpha_i^{Y \sqcup^X Z} := \upsilon_i + \zeta_i + \chi_i \end{array}$$

where we define $\upsilon_i := \mathsf{inl} \circ \alpha_i^Y$, $\zeta_i := \mathsf{inr} \circ \alpha_i^Z$, and $\chi_i : \mathbb{S}^i \times c_i^X \to P_i$ is defined by $\mathbb{S}^i$-induction: on point constructors by setting $\chi_i(\mathsf{north}, y) := \mathsf{inl}\,(f_{i+1}(\mathsf{cell}\, y))$ and $\chi_i(\mathsf{south}, y) := \mathsf{inr}\,(g_{i+1}(\mathsf{cell}\, y))$, and on the path constructor by letting $\mathsf{ap}_{\chi_i(-,y)}(\mathsf{merid}\; x)$ be the composite path

$$\mathsf{inl}\,(\dots) \xrightarrow{\mathsf{ap}_{\mathsf{inl}}\, l} \mathsf{inl}\,(\dots) \xrightarrow{\mathsf{push}\,(\alpha_{i-1}^X(x,y))} \mathsf{inr}\,(\dots) \xrightarrow{\mathsf{ap}_{\mathsf{inr}}\, r^{-1}} \mathsf{inr}\,(\dots)$$

where $l : f_i(\mathsf{cell}\, y) = \iota_{i-1}(f_{i-1}(\alpha_{i-1}^X(x,y)))$ and is defined by $l := \mathsf{ap}_{f_i}(\mathsf{glue}(x,y)^{-1}) \cdot h_{i-1}(\alpha_{i-1}^X(x,y))$, and similarly for $r : g_i(\mathsf{cell}\, y) = \iota_{i-1}(g_{i-1}(\alpha_{i-1}^X(x,y)))$.

Proposition 11. *Definition 10 satisfies (A1) and (A2).*

Proof. The proof mostly follows from the 3×3 lemma. The interested reader can consult the formalised version. □

Note that the colimit of this definition is the expected pushout:

$$\operatorname*{colim}_{i \to \infty}\left(Y \sqcup^X Z\right)_i = \operatorname*{colim}_{i \to \infty}\left(Y_i \sqcup^{X_{i-1}} Z_i\right) \simeq Y_\infty \sqcup^{X_\infty} Z_\infty$$

and thus Definition 10 does indeed provide a CW structure for the pushout of a span in CWstr. Now, if we want to extend this construction to the category $\mathsf{Ho}(\mathsf{CW})$, we need to work with arbitrary maps between the colimits instead of cellular maps. This is one out of a handful places where cellular approximation is needed.

B. Finite structures and the cellular approximation theorem

In classical algebraic topology, the cellular approximation theorem is a cornerstone result which states that any continuous function between two CW complexes is homotopic to a cellular map. This seems perfect for extending our constructions of pushouts to $\mathsf{Ho}(\mathsf{CW})$, but unfortunately this theorem appears to be out of reach in our constructive framework: the standard proof involves considerations of point-set topology and the use of the axiom of choice. However, what we can prove is a *synthetic* and *finitary* version of the theorem, which informally states that the cellular approximation theorem holds when the domain is finite dimensional. This will be the main result of this subsection.

Before providing the precise statement for our constructive cellular approximation theorem, let us start with a brief digression about finite subcomplexes and substructures – this will allow us to formulate a statement that is somewhat more flexible than the one suggested above. We say that a CW structure is *finite* (of dimension n) if the maps in its underlying sequence of types are equivalences starting from dimension n. Given any CW structure X_*, there is a canonical way to restrict it to a finite CW structure $X_*^{(n)}$ with the following definitions:

$$X_i^{(n)} := \begin{cases} X_i & \text{if } i < n \\ X_n & \text{otherwise} \end{cases} \qquad c_i^{(n)} := \begin{cases} c_i & \text{if } i \leq n \\ \bot & \text{otherwise.} \end{cases}$$

The structure $X_*^{(n)}$ trivially satisfies $X_\infty^{(n)} \simeq X_n$. We will use the same notation for cellular maps, writing $f_*^{(n)}$ for the restrictions of a cellular map f_* to the n-skeleton of the domain. For ease of notation, we also define $X_*^{(\infty)} := X_*$ (and similarly for $f_*^{(\infty)}$).

Definition 12. *Let X_* and Y_* be two CW structures, and let $f : X_\infty \to Y_\infty$ be an arbitrary map between their colimits. A **cellular n-approximation** of f is the data of a cellular map $(f_*, h_*) : X_*^{(n)} \to Y_*$ along with a homotopy*

$$t : (x : X_n) \to f(\iota_\infty x) = \iota_\infty(f_n\, x).$$

Our first cellular approximation states that n-approximations always exist for n finite. To get there, we will need the help of two easy lemmas.

Lemma 13. *For any CW structure X_*, the inclusion map $\iota_i : X_i \to X_{i+1}$ is $(i-1)$-connected.*

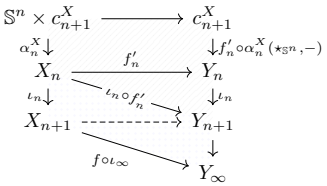
Proof. It is a general fact that given any span $B \xleftarrow{f} A \xrightarrow{g} C$, the map $\mathsf{inl} : B \to B \sqcup^A C$ is as connected as g [2, Proposition 2.3.10]. In our case, X_{i+1} is defined as the pushout of the span $X_i \leftarrow \mathbb{S}^i \times c_{i+1} \xrightarrow{\mathsf{snd}} c_{i+1}$, and thus it suffices to show that the projection $\mathsf{snd} : \mathbb{S}^i \times c_{i+1} \to c_{i+1}$ is $(i-1)$-connected. Indeed, its fibres are equivalent to $\mathbb{S}^i$, which is $(i-1)$-connected. □

Lemma 14. *For any CW structure X_*, the inclusion map $\iota_\infty : X_i \to X_\infty$ is $(i-1)$-connected.*

Proof. It follows immediately from Lemma 13 that all of the maps $X_{i+k} \xrightarrow{\iota_{i+k}} X_{i+k+1}$ are at least $(i-1)$ connected. As a consequence, their transfinite composition $\iota_\infty : X_i \to X_\infty$ is also $(i-1)$-connected [9, Corollary 7.7]. □

Theorem 15 (First cellular approximation theorem)**.** *Let X_* and Y_* be CW structures. For any map $f : X_\infty \to Y_\infty$ and $n : \mathbb{N}$, there merely exists a cellular n-approximation of f.*

Proof. The proof proceeds by induction on n. The base case, $n = -1$, is trivial. For the inductive step, assume that we have an n-approximation f'_* of f. We will use f'_* to *merely* construct an $(n+1)$-approximation $f_* : X_*^{(n+1)} \to Y_*$ (the fact that we are only aiming for mere existence allows us to use the elimination rule for propositional truncations a finite number of times). We define $f_i := f'_i$ for all $i \leq n$. It remains to define f_{n+1} and its associated homotopies. Consider the following (not necessarily commutative) diagram:



If we can construct f_{n+1} as the dashed map above in a way that makes all triangles commute, we are done. By the elimination principle of pushouts, the dashed map exists if we can fill the shaded area. In other words, we need to construct an element of type $\|((x,y) : \mathbb{S}^n \times c_{n+1}) \to F(x,y) = F(\star,y)\|_{-1}$ for $F := \iota_n \circ f'_n \circ \alpha_n^X$. Using the projectivity of c_{n+1} and Lemma 4, this corresponds to constructing, for every $y : c_{n+1}$, a family of paths $(x : \mathbb{S}^n) \to \|F(x,y) = F(\star,y)\|_{n-1}$. By Lemma 14, the map $\iota_\infty : Y_{n+1} \to Y_\infty$ is n-connected and therefore its action on path spaces, $\mathsf{ap}_{\iota_\infty}$, is $(n-1)$-connected. Thus $\|F(x,y) = F(\star,y)\|_{n-1}$ is equivalent to $\|\iota_\infty(F(x,y)) = \iota_\infty(F(\star,y))\|_{n-1}$. Since the dotted area of the diagram commutes, it suffices to show that the outermost diagram commutes, which is a consequence of the identity $\iota_\infty \circ f'_n = f \circ \iota_\infty$ and the fact that X_* is a CW structure. The remaining homotopy involved in the definition of a cellular approximation holds by construction. □

Corollary 16. *For any span of CW complexes $Y \xleftarrow{f} X \xrightarrow{g} Z$ with X finite, the pushout $Y \sqcup^X Z$ is a CW complex.*

Unfortunately, our finitary approximation theorem is not quite strong enough to prove the existence of all pushouts in $\mathsf{Ho}(\mathsf{CW})$. One option to remedy this would be to assume the axiom of countable choice, which allows us to deduce the mere existence of an ∞-approximation from the mere existence of an n-approximation for every n. This would, however, limit the generality of our theorems (countable choice does not hold in arbitrary infinity toposes), so we will refrain from doing so.

Question 17. *Can we prove that every map between CW complexes merely has an ∞-approximation without using the axiom of countable choice?*

Since we do not know the answer to this question, we will have to work with finite cellular approximations for the rest of this paper. For this purpose, we introduce the notion of n-truncated complexes, which can be faithfully captured by finite cellular approximations:

Definition 18. *An n-truncated CW complex is an n-truncated type A for which there merely exists a CW structure X_* of dimension $n+1$ such that $A \simeq \|X_{n+1}\|_n$.*

We write $\mathsf{Ho}(\mathsf{CW}^{(n)})$ for the category of n-truncated CW complexes and ordinary maps. Note the mismatch between the dimension of the structure and the truncation level in Definition 18. This mismatch is here so that we may define a truncation functor trunc_n from $\mathsf{Ho}(\mathsf{CW})$ to $\mathsf{Ho}(\mathsf{CW}^{(n)})$: we can send the pair $(A, |X_*|)$ to $(\|A\|_n, |X_*^{(n+1)}|)$, and the isomorphism condition holds because $\|X_\infty\|_n \simeq \|X_{n+1}\|_n$. We also introduce a corresponding category $\mathsf{CWstr}^{(n)}$ whose objects are CW structures of dimension $n+1$, and whose morphisms are cellular maps of dimension n.

C. Cellular homotopies and the 2nd approximation theorem

In essence, the first cellular approximation tells us that there *merely* exists an inverse to the colimit operation for finite cellular maps. This already lets us transfer some constructions from CWstr to $\mathsf{Ho}(\mathsf{CW})$, but we would ideally like to get rid of that propositional truncation and define a proper approximation functor from $\mathsf{Ho}(\mathsf{CW})$ to CWstr – or at least from $\mathsf{Ho}(\mathsf{CW}^{(n)})$ to $\mathsf{CWstr}^{(n)}$, to avoid choice issues. Unfortunately, this turns out to be problematic, as the cellular approximation theorem is inconsistent without the propositional truncation.

Theorem 19. *The set-truncated version of Theorem 15 is false.*

Proof. Both $\mathbb{1}$ and $\mathbb{S}^1$ can be presented by finite CW structures (which we will denote by $\mathbb{1}_*$ and $\mathbb{S}^1{}_*$), with only a 0-cell for the former and a 0-cell plus a 1-cell for the latter. Given any $x : \mathbb{S}^1$, define $\widehat{x} : \mathbb{1} \to \mathbb{S}^1$ to be the corresponding function. A cellular approximation of $\widehat{x}$ means that we can factor it as $\mathbb{1} \xrightarrow{\sim} \mathbb{1}_0 \xrightarrow{\widehat{x}_0} \mathbb{S}^1{}_0 \to \mathbb{S}^1$, which in turn implies that x is equal to the basepoint $\star_{\mathbb{S}^1}$. Therefore, the set-truncated version of Theorem 15 provides a proof of $(x : \mathbb{S}^1) \to \|x = \star_{\mathbb{S}^1}\|_0$. By Lemma 4, this entails $\|(x : \mathbb{S}^1) \to x = \star_{\mathbb{S}^1}\|_{-1}$ which by truncation elimination implies that $\mathbb{S}^1$ is contractible. But this is provably false in HoTT [10]. □

This problem stems from a mismatch between the notion of equality for morphisms in $\mathsf{Ho}(\mathsf{CW})$ and the notion of equality for morphisms in CWstr. Since any homotopy gives rise to an equality between maps, the morphisms in $\mathsf{Ho}(\mathsf{CW})$ should be understood as maps *up to homotopy*, while the equality between morphisms in CWstr is much closer in spirit to a strict equality. Therefore, if we want to frame our approximation theorem as a functor, we need to quotient the morphisms of the target category by an adequate notion of homotopy.

Definition 20. *A **cellular homotopy** between cellular maps $f_*, g_* : X_* \to Y_*$ is a family*

$$p_i : (x : X_i) \to \iota_i(f_i(x)) =_{Y_{i+1}} \iota_i(g_i(x))$$

with fillers $q_i\, x$, for each $i \geq 0$ and $x : X_i$, of the following square.

$$\begin{array}{ccc} \iota_{i+1}(\iota_i\,(f_i\,x)) & \xrightarrow{\mathsf{ap}_{\iota_{i+1}}(p_i\,x)} & \iota_{i+1}(\iota_i\,(g_i\,x)) \\ \downarrow & \overset{q_i\,x}{=\!=} & \downarrow \\ \iota_{i+1}(f_{i+1}(\iota_i\,x)) & \xrightarrow[p_{i+1}(\iota_i\,x)]{} & \iota_{i+1}(g_{i+1}(\iota_i\,x)) \end{array}$$

We use the notation $(p_*, q_*) : f_* \sim g_*$ or simply $p_* : f_* \sim g_*$ when the q_i's are clear from context.

One can easily prove that composition of cellular maps is invariant with respect to cellular homotopy. This lets us define the category $\mathsf{Ho}(\mathsf{CWstr})$, whose objects are CW structures and whose morphisms are cellular maps up to cellular homotopy. Furthermore, the existence of a cellular homotopy between f_* and g_* implies that their colimits are homotopic, or in other words, that $f_\infty = g_\infty$. This means that the colimit functor factors through $\mathsf{Ho}(\mathsf{CWstr})$.

That new colimit functor *almost* induces an equivalence between the categories $\mathsf{Ho}(\mathsf{CW}^{(n)})$ and $\mathsf{Ho}(\mathsf{CWstr}^{(n)})$. In order to prove this, we will need to extend our approximation theorem to cellular homotopies. Because the caveats regarding the axiom of countable choice still apply, we start by introducing a notion of finite approximation for cellular homotopies.

Definition 21. *Let $f_*, g_* : X_* \to Y_*$ be two cellular maps, and let $p : (x : X_\infty) \to f_\infty(x) = g_\infty(x)$ be a homotopy between their colimits. A **cellular n-approximation** of p is a cellular homotopy $p_* : f_*^{(n)} \sim g_*^{(n)}$ equipped with a filler of the following square for each $x : X_n$.*

$$\begin{array}{ccc} f_\infty(\iota_\infty\,x) & \xrightarrow{p(\iota_\infty\,x)} & g_\infty(\iota_\infty\,x) \\ \downarrow & = & \downarrow \\ \iota_\infty(\iota_n(f_n(x))) & \xrightarrow[\mathsf{ap}_{\iota_\infty}(p_n(x))]{} & \iota_\infty(\iota_n(g_n(x))) \end{array}$$

We are now ready to state the second cellular approximation theorem. Its proof follows the same strategy as Theorem 15, so we omit it but remark that it has been mechanised.

Theorem 22 (Second cellular approximation theorem)**.** *Let $f_*, g_* : X_* \to Y_\infty$ be cellular maps and $p : f_\infty \sim g_\infty$. For any $n : \mathbb{N}$, there merely exists an n-approximation of p.*

Theorem 22 implies that taking the colimit of a cellular map between two CW structures in $\mathsf{Ho}(\mathsf{CWstr}^{(n)})$ is an injective operation. On the other hand, Theorem 15 implies that it is a surjective operation. Therefore, the colimit induces a fully faithful functor from $\mathsf{Ho}(\mathsf{CWstr}^{(n)})$ to $\mathsf{Ho}(\mathsf{CW}^{(n)})$. Since this functor is essentially surjective in the sense of [1, Chapter 9], we get the following result as a corollary.

Corollary 23. *The colimit functor induces a weak equivalence between $\mathsf{Ho}(\mathsf{CWstr}^{(n)})$ and $\mathsf{Ho}(\mathsf{CW}^{(n)})$. Equivalently, $\mathsf{Ho}(\mathsf{CW}^{(n)})$ is the Rezk completion of $\mathsf{Ho}(\mathsf{CWstr}^{(n)})$.*

The relations between the various categories defined so far are summarised in Figure 1. This diagram gives us a systematic way of lifting a functor F defined over CWstr to a functor defined over $\mathsf{Ho}(\mathsf{CW})$: first, if the functor F happens to use only a finite number of dimensions, it can

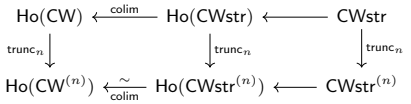


Fig. 1. The categories at play

be factored as $\overline{F} \circ \mathsf{trunc}_n$ for some functor $\overline{F}$ defined over $\mathsf{CWstr}^{(n)}$. Then, if we manage to prove that $\overline{F}$ is invariant under cellular homotopy, we can extend it to a functor $\widetilde{F}$ defined over $\mathsf{Ho}(\mathsf{CWstr}^{(n)})$. Finally, if the target is a univalent category, $\widetilde{F}$ can be extended to a functor defined over the Rezk completion of $\mathsf{Ho}(\mathsf{CWstr}^{(n)})$, which is $\mathsf{Ho}(\mathsf{CW}^{(n)})$. By composing the result with the truncation functor, we get a lift of F to $\mathsf{Ho}(\mathsf{CW})$.

IV. Cellular Homology

In algebraic topology, the homology groups of a space is a family of topological invariants which are somewhat similar to homotopy groups in that they intuitively measure the number of n-dimensional holes, but are much simpler to compute. There is a plethora of *homology theories* (roughly, different definitions for these homology groups) but among them, one is especially relevant to our interests: the theory of *cellular homology* is defined in terms of CW structures, and is particularly well suited for computation. Developing cellular homology in HoTT gives a new meaning to the adjective *computational*. Through the Curry–Howard correspondence, it provides formally verified computations of homology groups, facilitating the idea of 'proof by computation' – a central idea in computer formalisation of synthetic homotopy theory [2], [11], [12], [13].

In this section, we define a (reduced) homology functor $\widetilde{H}_i^{\mathsf{str}} : \mathsf{CWstr} \to \mathsf{AbGrp}$ which we then lift to a functor $\widetilde{H}_i^{\mathsf{cw}}$ over $\mathsf{Ho}(\mathsf{CW})$ using our freshly proved cellular approximation theorem. This provides the first complete definition of cellular homology in HoTT. We also state a finitary version of the Eilenberg–Steenrod axioms and prove that our functors satisfy them, thereby showing that they deserve the name of a 'homology theory'.

A. A crash course in homological algebra

The first step in the definition of homology groups is to approximate CW structures by *cellular chain complexes*. Before doing so, however, we need some preliminary background on chain complexes, as well as a definition of the homology groups of a chain complex.

Definition 24. *A **chain complex** is a sequence of abelian groups (called i-chains)*

$$\dots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} C_{-1} \xrightarrow{\partial_{-1}} \dots$$

where the maps ∂_i (called boundary maps) are group homomorphisms satisfying the equation $\partial_i \circ \partial_{i+1} = 0$.

Definition 25. *A **chain map** $\phi_* : C_* \to D_*$ is a collection of group homomorphisms $\phi_i : C_i \to D_i$ compatible with boundary maps in the sense that $\phi_i \circ \partial^C_{i+1} = \partial^D_{i+1} \circ \phi_{i+1}$.*

There are natural definitions of chain map composition (levelwise composition) and of the identity chain map (the levelwise identity). This lets us define the category Ch whose objects are chain complexes and whose morphisms are chain maps. We also have a natural notion of chain homotopy.

Definition 26. *A **chain homotopy** η_* between two chain maps $\phi_*, \psi_* : C_* \to D_*$ is a sequence of group homomorphisms $\eta_i : C_i \to D_{i+1}$ such that $\phi_i - \psi_i = \partial^D_{i+1} \circ \eta_i + \eta_{i-1} \circ \partial^C_i$.*

Chain homotopies are compatible with composition, which lets us define the homotopy category of chain complexes $\mathsf{Ho}(\mathsf{Ch})$ whose morphisms are chain maps up to chain homotopy. We finally arrive at the definition of homology groups, which is the natural analogue of homotopy groups in the category of chain complexes.

Definition 27 (Homology groups)**.** *We define the nth homology group of a chain complex (C_*, ∂_*) by $H_n(C_*) := \ker \partial_n / \operatorname{im} \partial_{n+1}$.*

We remark that the quotient in the definition above is well-defined since the boundary equation $\partial_i \circ \partial_{i+1} = 0$ ensures that $\operatorname{im} \partial_{i+1} \subseteq \ker \partial_i$. Furthermore, any chain map $\phi_* : C_* \to D_*$ induces a homomorphism $H_n(\phi_*) : H_n(C_*) \to H_n(D_*)$, and it does so in a functorial way. Thus, the definition of the nth homology group can be presented as a functor from Ch to the category of abelian groups AbGrp. Lastly, a standard argument shows that the existence of a chain homotopy between two chain maps ϕ_* and ψ_* implies that $H_n(\phi_*) \cong H_n(\psi_*)$. Therefore, the definition of H_n factors through the category $\mathsf{Ho}(\mathsf{Ch})$. This concludes our definition of the homology groups of a chain complex.

B. Sphere bouquets and reduced cellular homology

We are now in a position to define the cellular chain complex associated to a CW structure X_*. The definition for the abelian groups of n-chains is rather straightforward:

- when $n \geq 0$, we set $C_n := \mathbb{Z}[c^X_n]$, i.e. C_n is the free abelian group with a generator for each n-cell in X_*,
- when $n = -1$, we set $C_{-1} := \mathbb{Z}$ (in technical terms, this means that we are defining the *augmented* chain complex of X_*, but we will not go into detail here),
- when $n < -1$, we define C_n to be the trivial group.

The definition of the boundary maps is slightly more involved. In positive degrees, our goal is to construct a homomorphism of free abelian groups $\partial_{i+1} : \mathsf{Hom}(\mathbb{Z}[c^X_{i+1}], \mathbb{Z}[c^X_i])$. To do so, we will exploit the fact that free abelian groups are closely related to wedge sums of spheres, which we call *sphere bouquets*. This approach is essentially a reinterpretation of the definition used by May [14] and Buchholtz and Favonia [5]. In what follows, and for the remainder of the paper, we will use somewhat non-standard terminology and say that a type A is finite if $A \simeq \mathsf{Fin}(k)$ for some k.

Definition 28. *Given a finite type A and an integer $n \geq 0$, define the **sphere bouquet** of cardinality $|A|$ and dimension n to be the type $\bigvee_A \mathbb{S}^n$, i.e. the wedge sum of $|A|$ n-spheres.*

Before clarifying the relation between sphere bouquets and free abelian groups, we first need to recall some well-known facts about the *degree* of an endo-function of $\mathbb{S}^n$. For any $n > 0$, there is an isomorphism deg from $\pi_n(\mathbb{S}^n)$ to $\mathbb{Z}$. In fact, this definition extends to any (not necessarily pointed) map $f : \mathbb{S}^n \to \mathbb{S}^n$. This is done by noting that the 'forgetful map' $\|\mathsf{fst}\|_0 : \pi_n(\mathbb{S}^n) \to \|\mathbb{S}^n \to \mathbb{S}^n\|_0$ is an equivalence. This allows us to define a degree map by the composition

$$(\mathbb{S}^n \to \mathbb{S}^n) \xrightarrow{|-|} \|\mathbb{S}^n \to \mathbb{S}^n\|_0 \xrightarrow{\|\mathsf{fst}\|_0^{-1}} \|\mathbb{S}^n \to_\star \mathbb{S}^n\|_0 \xrightarrow{\mathsf{deg}} \mathbb{Z}$$

We allow some overloading of notation by also using deg to denote the above composition. In addition to inducing an isomorphism of groups, deg has a few useful properties.

Proposition 29. *The degree map commutes with suspensions, i.e. any $f : \mathbb{S}^n \to \mathbb{S}^n$ is of the same degree as its suspension $\Sigma f : \mathbb{S}^{n+1} \to \mathbb{S}^{n+1}$. Additionally,* deg *takes function composition to integer multiplication, i.e.* $\mathsf{deg}(f \circ g) = \mathsf{deg}\, f \cdot \mathsf{deg}\, g$.

As the degree map has been well studied in HoTT already [5], [15], we omit the proof of Proposition 29. This degree function has a natural generalisation to sphere bouquets, which we call the *bouquet degree* function (bdeg). It is defined by the following composition of arrows:

$$\begin{array}{ccc}
(\bigvee_A \mathbb{S}^n \to \bigvee_B \mathbb{S}^n) & \to & \Pi_A \Pi_B (\mathbb{S}^n \to \mathbb{S}^n) \\
\downarrow & & \downarrow{\scriptstyle (\mathsf{deg}^*)^*} \\
(\Pi_A \mathbb{S}^n \to \bigvee_B \mathbb{S}^n) & & \Pi_A \Pi_B \mathbb{Z} \\
{\scriptstyle \iota_\vee{}^*}\downarrow & & \downarrow{\scriptstyle \wr} \\
(\Pi_A \mathbb{S}^n \to \Pi_B \mathbb{S}^n) & & \mathsf{Hom}(\mathbb{Z}[A], \mathbb{Z}[B])
\end{array}$$

where the last equivalence is defined using the universal property of the free abelian group. The bouquet degree map immediately inherits properties corresponding to those listed in Proposition 29:

Proposition 30. *Let A, B and C be finite types, $n \geq 0$. The following facts hold.*

1) *The bouquet degree function induces a group homomorphism $\left\|\bigvee_A \mathbb{S}^{n+1} \to \bigvee_B \mathbb{S}^{n+1}\right\|_0 \to \mathsf{Hom}(\mathbb{Z}[A], \mathbb{Z}[B])$, where the group structure on the left-hand side is the natural extension of the group structure on $\pi_{n+1}(\mathbb{S}^{n+1})$.*
2) *The bouquet degree function commutes with suspension, i.e. any $f : \bigvee_A \mathbb{S}^n \to \bigvee_B \mathbb{S}^n$ is of the same degree as its suspension $\Sigma f : \Sigma(\bigvee_A \mathbb{S}^n) \to \Sigma(\bigvee_B \mathbb{S}^n)$, where the bouquet degree of the latter function is well-defined since $\Sigma(\bigvee_X \mathbb{S}^n) \simeq \bigvee_X \mathbb{S}^{n+1}$ for any X.*
3) *The bouquet degree function respects composition, i.e. for $f : \bigvee_A \mathbb{S}^n \to \bigvee_B \mathbb{S}^n$ and $g : \bigvee_B \mathbb{S}^n \to \bigvee_C \mathbb{S}^n$ we have $\mathsf{bdeg}(g \circ f) = \mathsf{bdeg}\, g \circ \mathsf{bdeg}\, f$.*

We can now return to the construction of the boundary maps: we would like to define, for any CW structure X_*, a homomorphism $\partial_{i+1} : \mathbb{Z}[c_{i+1}^X] \to \mathbb{Z}[c_i^X]$. By applying our bouquet degree function, it suffices to construct a function $d_i : \bigvee_{c_{i+1}^X} \mathbb{S}^{i+1} \to \bigvee_{c_i^X} \mathbb{S}^{i+1}$. We recall from [5] that there is an equivalence $e : X_i/X_{i-1} \simeq \bigvee_{c_i^X} \mathbb{S}^i$. When $i > 0$, we obtain it by considering the following diagram.

$$\begin{array}{ccccc}
\mathbb{S}^{i-1} \times c_i^X & \xrightarrow[\alpha_{i-1}^X]{} & X_{i-1} & \longrightarrow & \mathbb{1} \\
\downarrow & & \downarrow & & \downarrow \\
c_i^X & \longrightarrow & X_i & \longrightarrow & \Sigma(\bigvee_{c_i^X} \mathbb{S}^{i-1})
\end{array}$$

Since the outermost square is a pushout, we know, by pushout pasting, that so is the right square. The construction of e is completed by observing that suspension commutes with wedge sums. When $i = 0$, the equivalence is obtained by noting that $X_0/X_{-1} \simeq X_0 + \mathbb{1}$ which allows us to identify the appended point with the basepoint in $\bigvee_{c_0^X} \mathbb{S}^0$. We may now construct the desired map d_{i+1} by considering the composition

$$\bigvee_{c_{i+1}^X} \mathbb{S}^{i+1} \xrightarrow{\sim} X_{i+1}/X_i \xrightarrow{\mathsf{pinch}} \Sigma X_i \xrightarrow{\Sigma\mathsf{cfcod}} \Sigma(X_i/X_{i-1}) \xrightarrow{\sim} \bigvee_{c_i^X} \mathbb{S}^{i+1}$$

where pinch is the *pinch* map, i.e. the pointed map identifying inr with south and push with merid. Finally, we set $\partial_{i+1} = \mathsf{bdeg}\, d_{i+1}$. Note that this whole construction is only valid for $i > -1$. To complete the definition, we define $\partial_0 : \mathbb{Z}[c_0^X] \to \mathbb{Z}$ by sending every generator of $\mathbb{Z}[c_0^X]$ to 1, and lastly we let the maps in negative dimension be trivial.

Proposition 31. *The boundary maps satisfy $\partial_i \circ \partial_{i+1} = 0$.*

Proof. First, assume that $i > 0$. We compute:

$$\begin{aligned}
\partial_i \circ \partial_{i+1} = \mathsf{bdeg}\, d_i \circ \mathsf{bdeg}\, d_{i+1} &= \mathsf{bdeg}\,(\Sigma d_i) \circ \mathsf{bdeg}\,(d_{i+1}) \\
&= \mathsf{bdeg}\,(\Sigma d_i \circ d_{i+1})
\end{aligned}$$

We are done if we can show that $\Sigma d_i \circ d_{i+1} = 0$. This composition of maps is defined as follows.

$$\begin{array}{l}
\bigvee_{c_{i+1}^X} \mathbb{S}^{i+1} \longrightarrow X_{i+1}/X_i \longrightarrow \Sigma X_i \dashrightarrow \Sigma(X_i/X_{i-1}) \dashrightarrow \bigvee_{c_i^X} \mathbb{S}^{i+1} \\
\Sigma \bigvee_{c_i^X} \mathbb{S}^i \dashrightarrow \Sigma(X_i/X_{i-1}) \dashrightarrow \Sigma^2 X_{i-1} \to \Sigma^2(X_{i-1}/X_{i-2}) \to \Sigma \bigvee_{c_{i-1}^X} \mathbb{S}^i
\end{array}$$

It is enough to show that the dashed composition $\Sigma X_i \to \Sigma^2 X_{i-1}$ is trivial. By tracing the construction of the maps involved, it is easy to see that the map is given by

$$\Sigma X_i \xrightarrow{\Sigma\mathsf{cfcod}} \Sigma X_i/X_{i-1} \xrightarrow{\Sigma\mathsf{pinch}} \Sigma^2 X_{i-1}$$

which is equal to functorial action of Σ on pinch $\circ$ cfcod : $X_i \to \Sigma X_{i-1}$. This is constant by definition. The case $i = 0$ follows by an explicit computation of the maps ∂_1 and ∂_0. □

At this point, we have a proper definition for the cellular chain complex of a CW structure. It remains to show that this construction lifts to a functor from CWstr to Ch.

Let $f_* : X_* \to Y_*$ be a cellular map. Because f_* is cellular, it determines a map $X_i/X_{i-1} \to Y_i/Y_{i-1}$. With a bit of help from the equivalence e that we defined earlier, we can define a map of sphere bouquets $\widetilde{f}_i$ as follows:

$$\bigvee_{c_i^X} \mathbb{S}^i \xrightarrow{e^{-1}} X_i/X_{i-1} \xrightarrow{f_i/f_{i-1}} Y_i/Y_{i-1} \xrightarrow{e} \bigvee_{c_i^Y} \mathbb{S}^i.$$

We may thus define the functorial action of f on i-chains, $\overline{f}_i : \mathsf{Hom}(\mathbb{Z}[c_i^X], \mathbb{Z}[c_i^Y])$, by setting $\overline{f}_i = \mathsf{bdeg}\, \widetilde{f}_i$. Let us verify that it is a chain map, i.e. that $\partial_{i+1} \circ \overline{f}_{i+1} = \overline{f}_i \circ \partial_{i+1}$. Using the fact that bdeg respects suspension and composition, this is equivalent to $\mathsf{bdeg}\,(d_{i+1} \circ \widetilde{f}_{i+1}) = \mathsf{bdeg}\,(\Sigma \widetilde{f}_i \circ d_{i+1})$. Let us simply show that $d_{i+1} \circ \widetilde{f}_{i+1} = \Sigma \widetilde{f}_i \circ d_{i+1}$. That is, we will show that the outer square commutes in the diagram below:

$$\begin{array}{ccccccc} \bigvee_{c_{i+1}^X} \mathbb{S}^{i+1} & \longrightarrow & X_{i+1}/X_i & \longrightarrow & \Sigma X_i & \longrightarrow & \Sigma(X_i/X_{i-1}) \longrightarrow \bigvee_{c_i^X} \mathbb{S}^{i+1} \\ \downarrow{\scriptstyle \bigvee \widetilde{f}_{i+1}} & & \downarrow{\scriptstyle f_{i+1}/f_i} & & \downarrow{\scriptstyle \Sigma f_i} & & \downarrow{\scriptstyle \Sigma f_i/f_{i-1}} \quad \downarrow{\scriptstyle \bigvee \Sigma \widetilde{f}_i} \\ \bigvee_{c_{i+1}^Y} \mathbb{S}^{i+1} & \longrightarrow & Y_{i+1}/Y_i & \longrightarrow & \Sigma Y_i & \longrightarrow & \Sigma(Y_i/Y_{i-1}) \longrightarrow \bigvee_{c_i^Y} \mathbb{S}^{i+1} \end{array}$$

This is immediate: the leftmost and rightmost squares commute by construction of our functorial action, and the middle squares commute by definition.

Thus, we have shown that any cellular map $f_* : X_* \to Y_*$ gives rise to a chain map between the cellular chain complexes of X_* and Y_*. Due to space constraints, we omit the proofs that this operation satisfies the two functor axioms, but we note that they are very direct. This results in a functor $\mathsf{cellChain} : \mathsf{CWstr} \to \mathsf{Ch}$. If we compose this functor with the nth homology functor $H_n : \mathsf{Ch} \to \mathsf{AbGrp}$, we obtain a functorial definition of reduced cellular homology for CW structures. We denote the resulting functor by $\widetilde{H}_n^{\mathsf{str}} : \mathsf{CWstr} \to \mathsf{AbGrp}$.

C. The homology of a CW complex

Our end goal is to extend our cellular homology functor to the category of CW complexes. To do so, we follow the strategy laid out in Figure 1: first, we will need a lemma to show that cellular homology is homotopy invariant.

Proposition 32. *Let f_* and g_* be two parallel cellular maps. Every cellular homotopy between f_* and g_*, induces a chain homotopy between $\mathsf{cellChain}(f_*)$ and $\mathsf{cellChain}(g_*)$.*

The proof is standard but somewhat technical. Due to space constraints, we omit it and refer to the computer formalisation. Proposition 32 implies that cellChain descends to a functor from $\mathsf{Ho}(\mathsf{CWstr})$ to $\mathsf{Ho}(\mathsf{Ch})$. As we already saw, the chain homology functor H_n factors through $\mathsf{Ho}(\mathsf{Ch})$, meaning that we can compose it with cellChain to express cellular homology as a functor $\widetilde{H}_n^{\mathsf{str}} : \mathsf{Ho}(\mathsf{CWstr}) \to \mathsf{AbGrp}$. Therefore, we have established that cellular homology is homotopy invariant.

In fact, a quick glance at the definition of cellular homology makes it clear that $\widetilde{H}_n^{\mathsf{str}}(X_*)$ only depends on the $(n+1)$-skeleton of X_*, so $\widetilde{H}_n^{\mathsf{str}}$ can actually be defined as a functor from $\mathsf{Ho}(\mathsf{CWstr}^{(n+1)})$ to AbGrp. Since abelian groups form a univalent category, $\widetilde{H}_n^{\mathsf{str}}$ can even be extended to the Rezk completion of $\mathsf{Ho}(\mathsf{CWstr}^{(n+1)})$, which is $\mathsf{Ho}(\mathsf{CW}^{(n+1)})$. Composing the resulting functor with the truncation functor from $\mathsf{Ho}(\mathsf{CW})$ to $\mathsf{Ho}(\mathsf{CW}^{(n+1)})$ yields the desired definition of the cellular homology functor $\widetilde{H}_n^{\mathsf{cw}} : \mathsf{Ho}(\mathsf{CW}) \to \mathsf{AbGrp}$.

D. The Eilenberg–Steenrod axioms

To be deserving of the title of a *homology theory*, our definition should satisfy the Eilenberg–Steenrod axioms [16]. However, this raises yet another constructivity issue: the modern formulation of these axioms involves wedge sums indexed by arbitrary sets, which are not guaranteed to exist in our category of CW structures. To remedy this, we work with a *finitary* version of the axioms[3]. In what follows, $\mathsf{CWstr}_\star$ denotes the category of pointed CW structures.

Definition 33 (Eilenberg–Steenrod homology)**.** ***A reduced homology theory*** *is a $\mathbb{Z}$-indexed family of functors $\widetilde{E}_n : \mathsf{CWstr}_\star \to \mathsf{AbGrp}$ satisfying the following axioms.*

Homotopy: *For any n and cellular homotopy $f_* \sim g_*$, we have that $\widetilde{E}_n(f_*) = \widetilde{E}_n(g_*)$.*

Suspension: *For any n, there is an isomorphism $\widetilde{E}_n(X_*) \cong \widetilde{E}_{n+1}((\Sigma X)_*)$ which is natural in X.*

Exactness: *For any cellular map f_*, the sequence*

$$\widetilde{E}_n(X_*) \xrightarrow{\widetilde{E}_n(f_*)} \widetilde{E}_n(Y_*) \xrightarrow{\widetilde{E}_n(\mathsf{cfcod}_*)} \widetilde{E}_n((C_f)_*)$$

is exact, meaning that $\ker \widetilde{E}_n(\mathsf{cfcod}_*) = \mathrm{im}\, \widetilde{E}_n(f_*)$.

Dimension: *$\widetilde{E}_n(\mathbb{S}^0_*)$ is trivial for $n \neq 0$ and isomorphic to $\mathbb{Z}$ when $n = 0$.*

Binary additivity: *For any $X_*, Y_* : \mathsf{CWstr}_\star$, the canonical map $\widetilde{E}_n(X_*) \oplus \widetilde{E}_n(Y_*) \to \widetilde{E}_n((X \vee Y)_*)$ is an isomorphism.*

An important point is that we decided to define the Eilenberg–Steenrod axioms over $\mathsf{CWstr}_\star$ rather than $\mathsf{Ho}(\mathsf{CW}_\star)$. The reason for this is that the exactness axiom involves cofibres of arbitrary maps, which are not guaranteed to exist in the category $\mathsf{Ho}(\mathsf{CW}_\star)$ (see the discussion around Corollary 16). Nevertheless, we do get a restricted exactness axiom for $\widetilde{H}_n^{\mathsf{cw}}$ which involves only maps with a finite domain as a consequence of the exactness of $\widetilde{H}_n^{\mathsf{str}}$. Before we prove exactness, however, let us show that the suspension axiom is satisfied (skipping the homotopy axiom, since it is an immediate consequence of Proposition 32). We note that, unlike the other proofs in this paper, the following proofs regarding the Eilenberg–Steenrod axioms only have been partially formalised.

Proposition 34. *The suspension axiom is satisfied by $\widetilde{H}_n^{\mathsf{str}}$.*

Proof. Let (C_*^X, ∂_*^X) and $(C_*^\Sigma, \partial_*^\Sigma)$ be the augmented chain complexes associated to X_* and $(\Sigma X)_*$ respectively. Let $(\widehat{C}_*^\Sigma, \widehat{\partial}_*^\Sigma) := (C_{*+1}^\Sigma, \partial_{*+1}^\Sigma)$ be the latter complex shifted by 1, and denote its chain homology groups by $\widetilde{H}_n^\Sigma$. We have $\widetilde{H}_n^\Sigma = \widetilde{H}_{n+1}^{\mathsf{str}}((\Sigma X)_*)$ by construction. We construct a chain

[3] After generalising our work from finitary to projective CW structures, it should be possible to show the additivity axiom for any projective set of indices, using techniques similar to those of Cavallo [17] and Buchholtz and Favonia [5].

map $\varphi_* : \widehat{C}^\Sigma_* \to C^X_*$ as follows:

$$\begin{array}{ccccccccccc}
 & & \mathbb{Z}[c_n] & & & & \mathbb{Z}[c_0] & & \mathbb{Z}[2] & & \\
 & & \| & & & & \| & & \| & & \\
\dots & \xrightarrow{\widehat{\partial}_{n+1}} & \widehat{C}^\Sigma_n & \xrightarrow{\widehat{\partial}_n} & \dots & \xrightarrow{\widehat{\partial}_1} & \widehat{C}^\Sigma_0 & \xrightarrow{\widehat{\partial}_0} & \widehat{C}^\Sigma_{-1} & \xrightarrow{\widehat{\partial}_{-1}} & \mathbb{Z} \\
 & & \downarrow{\scriptstyle \varphi_n} & & & & \downarrow{\scriptstyle \varphi_0} & & \downarrow{\scriptstyle \varphi_{-1}} & & \downarrow{\scriptstyle \varphi_{-2}} \\
\dots & \xrightarrow[\partial_{n+1}]{} & C^X_n & \xrightarrow[\partial_n]{} & \dots & \xrightarrow[\partial_1]{} & C^X_0 & \xrightarrow[\partial_0]{} & C^X_{-1} & \longrightarrow & \mathbb{1} \\
 & & \| & & & & \| & & \| & & \\
 & & \mathbb{Z}[c_n] & & & & \mathbb{Z}[c_0] & & \mathbb{Z} & &
\end{array}$$

We simply set φ_n to be the identity when $n \geq 0$, and let φ_{-1} be the map forgetting the second generator. The fact that the squares commute is a direct consequence of Proposition 30, apart from the second square from the right, whose commutativity follows by construction of $\widehat{\partial}_0$. Thus φ_n induces an isomorphism $\phi_n : \widetilde{H}^{\mathsf{str}}_{n+1}((\Sigma X)_*) = \widetilde{H}^\Sigma_n \to \widetilde{H}^{\mathsf{str}}_n(X_*)$ on homology when $n \geq 1$. Naturality is immediate as the isomorphism is induced by the identity. When $n = 0$, we need to be somewhat more careful since $\widehat{\partial}_0$ and ∂_0 have different codomains. Nonetheless, their kernels trivially agree and so we still obtain the desired isomorphism on homology. The final non-trivial case we need to check is when $n = -1$. This case amounts to showing that $\widetilde{H}^\Sigma_{-1}$ is trivial which follows immediately by construction of $\widehat{\partial}_0$ and $\widehat{\partial}_{-1}$. □

Let us continue with the exactness axiom. For this, we need to characterise the behaviour of the boundary map on pushouts. The proof, which we have to omit here due to space constraints, proceeds by unfolding the definition of the attaching maps in Definition 10 and some direct but tedious computations.

Lemma 35. *Let P_* be the cellular pushout of some span $Y_* \xleftarrow{f_*} X_* \xrightarrow{g_*} Z_*$. The boundary map ∂^P_{n+1} factors as*

$$C^p_{n+1} \xrightarrow{\sim} C^X_n \oplus C^Y_{n+1} \oplus C^Z_{n+1} \xrightarrow{\partial'_{n+1}} C^X_{n-1} \oplus C^Y_n \oplus C^Z_n \xrightarrow{\sim} C^P_n$$

where $\partial'_{n+1}(x, y, z) := (-\partial^X_n\, x, \partial^Y_{n+1}\, y + \overline{f}_n\, x, \partial^Z_{n+1}\, z - \overline{g}_n\, x)$.

Proposition 36. *The exactness axiom is satisfied by $\widetilde{H}^{\mathsf{str}}_n$.*

Proof. The fact that $\operatorname{im}(\widetilde{H}^{\mathsf{str}}_n(f_*)) \subseteq \ker(\widetilde{H}^{\mathsf{str}}_n(\mathsf{cfcod}_*))$ follows from the functoriality of $\widetilde{H}^{\mathsf{str}}_n$ and the fact that $\mathsf{cfcod}_* \circ f_*$ is constant by definition of $(C_f)_*$. For the other direction, let $[y] : \widetilde{H}^{\mathsf{str}}_n(Y_*)$ be an equivalence class (where $y : C^Y_n$) and assume it is in the kernel of the composite map $C^Y_n \xrightarrow{\overline{\mathsf{cfcod}}_n} C^{\mathsf{cof}}_n \xrightarrow{q} C^{\mathsf{cof}}_n / \partial^{\mathsf{cof}}_{n+1}$. A quick computation reveals that the group C^{cof}_n is equal to $C^X_{n-1} \oplus C^Y_n \oplus \mathbb{1}_n$, and that $\overline{\mathsf{cfcod}}_n\, y$ is equal to $(0, y, 0)$. Therefore, our assumption is equivalent to $(0, y, 0)$ being in the image of $\partial^{\mathsf{cof}}_{n+1}$. Using Lemma 35, this means that $y = \partial^Y_{n+1}\, y_0 + \overline{f}_n\, x$ for some $y_0 : C^Y_{n+1}$ and $x : C^X_n$. Thus, $[y] = [\overline{f}_n\, x]$ in $\widetilde{H}^{\mathsf{str}}_n(Y_*)$. Since $[\overline{f}_n\, x] = \widetilde{H}^{\mathsf{str}}_n(f_*)[x]$, we are done. □

Proposition 37. *The dimension axiom is satisfied by $\widetilde{H}^{\mathsf{str}}_n$.*

Proof. The augmented chain complex associated to $\mathbb{S}^0$ is

$$\dots \xrightarrow{\partial_3} \mathbb{1} \xrightarrow{\partial_2} \mathbb{1} \xrightarrow{\partial_1} \mathbb{Z}[2] \xrightarrow{\partial_0} \mathbb{Z} \xrightarrow{\partial_{-1}} \mathbb{1} \xrightarrow{\partial_{-2}} \mathbb{1} \xrightarrow{\partial_{-3}} \dots$$

The homology of this complex is clearly concentrated in degree 0 with $\widetilde{H}^{\mathsf{str}}_n(\mathbb{S}^0) \cong \mathbb{Z}$. □

Proposition 38. *Binary additivity is satisfied by $\widetilde{H}^{\mathsf{str}}_n$.*

Proof. The direct sum $\widetilde{H}^{\mathsf{str}}_n(X_*) \oplus \widetilde{H}^{\mathsf{str}}_n(Y_*)$ can be viewed as the homology of the chain complex $(C^X_n \oplus C^Y_n, \partial^X_n \oplus \partial^Y_n)$. Under this identification, the map $\widetilde{H}^{\mathsf{str}}_n(X_*) \oplus \widetilde{H}^{\mathsf{str}}_n(Y_*) \to \widetilde{H}^{\mathsf{str}}_n((X \vee Y)_*)$ corresponds to the chain map

$$\begin{array}{ccccc}
\dots & \longrightarrow & C^Y_n \oplus C^Z_n & \xrightarrow{\partial^Y_n \oplus \partial^Z_n} & C^Y_n \oplus C^Z_n \longrightarrow \dots \\
 & & \downarrow & \nearrow_{0 \oplus \partial^Y_n \oplus \partial^Z_n} & \downarrow \\
\dots & \longrightarrow & C^{\mathbb{1}}_n \oplus C^Y_{n+1} \oplus C^Z_{n+1} & & C^{\mathbb{1}}_{n-1} \oplus C^Y_n \oplus C^Z_n \longrightarrow \dots
\end{array}$$

where the bottom row is the reduced cell complex associated to $(X \vee Y)_*$ using the cell structure for pushouts. The computation of the boundary map comes from Lemma 35. As $C^{\mathbb{1}}_n$ vanishes for $n \geq 2$, the vertical maps are isomorphisms in these dimensions and hence we obtained the desired isomorphism of homology in groups. The additional 0-cell in $(X \vee B)_0$ forces us to construct the inverse of the prospective isomorphism explicitly in dimensions $n = 1$ and $n = 0$. The construction is completely standard and we refer to May [14] for details. □

Theorem 39. *The functor $\widetilde{H}^{\mathsf{str}}_n : \mathsf{CWstr}_\star \to \mathsf{AbGrp}$ is a reduced homology theory.*

Finally, we arrive at the corresponding result for $\widetilde{H}^{\mathsf{cw}}_n$ (where the notion of exactness is restricted to mention only the pushouts which exist in $\mathsf{Ho}(\mathsf{CW}_\star)$, i.e. those of maps with finite CW structures as domains).

Corollary 40. *The functor $\widetilde{H}^{\mathsf{cw}}_n : \mathsf{Ho}(\mathsf{CW}_\star) \to \mathsf{AbGrp}$ is a reduced homology theory.*

Some care has to be taken when inferring Corollary 40 from Theorem 39. As $\widetilde{H}^{\mathsf{cw}}_n$ concerns the homology of arbitrary types *merely* equipped with a CW structure, we are only able to automatically infer the two axioms which happen to be propositions, namely exactness and binary additivity. The dimension axiom follows because it concerns $\mathbb{S}^0$, a closed type for which we have an explicit CW structure. Finally, we need to take care of the suspension axiom. Its statement is a set and not a proposition, which prevents us from using the usual elimination principle for truncations, but we can instead use the set elimination principle of Kraus [18, Chapter 8.1.1]. We need to prove the theorem whenever X has an explicit CW structure (using Theorem 39), and then show that the proof does not depend on the choice of CW structure, which is a direct consequence of naturality.

V. Part 4: The Hurewicz Theorem

As previously mentioned, homology groups are quite similar in spirit to homotopy groups, so one might hope that the two notions are connected in some way. The answer lies in the Hurewicz theorem, which states that if a space is n-connected, then its homology groups coincide with its homotopy groups up to dimension $n+1$ (up to abelianisation in the case $n = 0$).

A. Approximating n-connected spaces

The classical proof of the Hurewicz theorem for cellular homology takes an arbitrary n-connected CW complex, and replaces its CW structure with an alternative one with no non-trivial cells in dimension $< n+1$. This is done by defining the new set of $(n+1)$ cells to be the generators of the $(n+1)$-fst homotopy group of the space, from which the Hurewicz theorem will follow. Unfortunately, this approach will not work in our framework since the homotopy groups of a finite CW complex are not necessarily finitely generated – this applies to, for instance, $\pi_2(\mathbb{S}^1 \vee \mathbb{S}^2)$. Yet, perhaps surprisingly, we are able to give a constructive proof of the Hurewicz theorem by using a different construction for the alternative structure of n-connected CW complexes.

Definition 41. *We say that a CW structure X_* is **Hurewicz n-connected** if $\left|c_0^X\right| = 1$ and $\left|c_i^X\right| = 0$ for $0 < i \leq n$. We use the same terminology for CW complexes which merely have a Hurewicz n-connected CW structure.*

We remark that being Hurewicz n-connected is a property (i.e. a proposition). The following lemma gives a few elementary consequences of Hurewicz n-connectedness.

Lemma 42. *Let X_* be a CW structure. If X_* is Hurewicz n-connected, then*

1) *$X_i \simeq \mathbb{1}$ for $0 \leq i \leq n$,*
2) *$X_{n+1} \simeq \bigvee_{c_{n+1}^X} \mathbb{S}^{n+1}$,*
3) *X_i is n-connected for $i \in \mathbb{N} \cup \{\infty\}$.*

Item 3 tells us that Hurewicz n-connectedness implies the usual notion of n-connectedness. The other direction is much less obvious – especially constructively. Nonetheless, we can, in fact, prove it. As a warm-up, let us tackle the case $n = 0$.

Proposition 43. *For any 0-connected structure X_*, there is a Hurewicz 0-connected CW structure X'_* s.t. $X_i = X'_i$ for $i \geq 1$.*

Proof. We proceed by induction on $\left|c_0^X\right|$, i.e. the size of X_0 (indeed, we have $X_0 \simeq c_0^X$). If $\left|c_0^X\right| = 0$, this contradicts the 0-connectedness of X_*. If $\left|c_0^X\right| = 1$, then X_* is already of the right form and there is nothing to prove. Consider now the case $\left|c_0^X\right| > 1$. We will be done if we can show that X_1 may be obtained as the pushout of $c'_0 \xleftarrow{\alpha'} \mathbb{S}^0 \times c'_1 \xrightarrow{\mathsf{snd}} c'_1$ for some α' and some finite sets c'_0 and c'_1 satisfying $|c'_0| < \left|c_0^X\right|$. Let us carry out the construction. Some of the arguments may look non-constructive but we emphasise that they are justified as they concern finite sets.

First, note that there must be some $a_0 : c_1$ such that $\alpha_0(\mathsf{north}, a_0) \neq \alpha_0(\mathsf{south}, a_0)$. Indeed, if this were not the case, we would have that $\|X_1\|_0 \simeq X_0$. By combining this equation with $\|X_\infty\|_0 \simeq \|X_1\|_0$, we would obtain that $\|X_\infty\|_0$ is isomorphic to X_0, which is not contractible since $\left|c_0^X\right| > 1$. Now, by permuting the elements of c_1 and c_0, we may assume that the last element $a_0 : c_1$ satisfies $\alpha_0(\mathsf{north}, a_0) = \left|c_0^X\right| - 1$ and $\alpha_0(\mathsf{south}, a_0) = \left|c_0^X\right| - 2$. We define a new attaching map $\alpha'_0 : \mathbb{S}^0 \times (c_1^X - 1) \to (c_0^X - 1)$ by

$$\alpha'_0(x, y) = \begin{cases} \alpha_0(x, y) \text{ if } \alpha_0(x, y) < |c_0| - 1 \\ \left|c_0^X\right| - 2 \text{ otherwise} \end{cases}$$

The 1-skeleton X'_1 obtained by pushing out along α'_0 is easily identified with X_1, and thus we are done as we have decreased the cardinality of the codomain of the attaching map by 1. □

Before turning to higher dimensions, let us define a useful alteration of the notion of CW structure. In what follows, we abuse notation for the sake of convenience and interpret $\bigvee_A \mathbb{S}^{-1}$ as the empty type rather than than the unit type.

Definition 44. ***A good** CW structure is a pointed CW structure X_* whose attaching maps $\alpha_i : c_{i+1}^X \times \mathbb{S}^i \to X_i$ lift to maps defined over sphere bouquets, i.e. for all i there exists a matching $\alpha'_i : \bigvee_{c_{i+1}^X} \mathbb{S}^i \to_\star X_i$ such that $X_{i+1} \simeq C_{\alpha'_i}$.*

Lemma 45. *Let X_* be a good CW structure. X_* is Hurewicz n-connected iff $X_{n+1} \simeq \bigvee_B \mathbb{S}^{n+1}$ and $X_{n+2} \simeq C_f$ where A and B are finite types and $f : \bigvee_A \mathbb{S}^{n+1} \to \bigvee_B \mathbb{S}^{n+1}$.*

This lemma follows immediately from the definition of good structures and Lemma 42. We remark that good CW structures always are Hurewicz 0-connected. The converse also holds for connectedness reasons.

Proposition 46. *Any finite Hurewicz 0-connected CW structure is merely good.*

We are now ready to prove the main technical theorem of which states that the synthetic standard notion of connectedness coincides, for CW complexes, with the more analytic notion of Hurewicz connectedness.

Theorem 47. *Let X_* be an n-connected CW structure. There merely exists a Hurewicz n-connected CW structure X'_* such that $X_i = X'_i$ for $i > n$.*

Proof. We proceed by induction on n. The base case is given by Proposition 43. For the inductive step, let X_* be n-connected. In particular, X_* is $(n-1)$-connected, so by induction hypothesis we may assume that it is Hurewicz $(n-1)$-connected. Since $n > 0$, this structure is also Hurewicz 0-connected and we may assume that it is good (up to some fixed finite dimension $k \gg n$) by Proposition 46. Using Lemma 45, we know that $X_n \simeq \bigvee_A \mathbb{S}^n$ and $X_{n+1} \simeq C_f$ for $f : \bigvee_B \mathbb{S}^n \to \bigvee_A \mathbb{S}^n$ where A and B are some finite sets. Using Lemma 45 again, we are done if we can construct sets A', B' and $f' : \bigvee_{B'} \mathbb{S}^{n+1} \to \bigvee_{A'} \mathbb{S}^{n+1}$ s.t. $X_{n+2} \simeq C_{f'}$ (note that we are then implicitly setting $X'_{n+1} := \bigvee_{A'} \mathbb{S}^{n+1}$). Consider the following diagram where $C = c_{n+2}^X$.

$$\begin{array}{ccccc} & & \bigvee_C \mathbb{S}^{n+1} & \longrightarrow & \mathbb{1} \\ & & \downarrow{\scriptstyle \alpha_{n+1}} & & \downarrow \\ \bigvee_A \mathbb{S}^n & \xrightarrow{\mathsf{cfcod}} & C_f & \longrightarrow & X_{n+2} \\ \downarrow & & \downarrow & & \downarrow \\ \mathbb{1} & \longrightarrow & \bigvee_B \mathbb{S}^{n+1} & \longrightarrow & X_{n+2} \vee \bigvee_A \mathbb{S}^{n+1} \end{array}$$

The top square is a pushout square because $X_*^{(k)}$ is a good CW structure (we have identified X_{n+1} with C_f). The fact

that the bottom-left square is a pushout follows by the first part of Lemma 2. The bottom right-square is less evident. Consider the composite map $\bigvee_A \mathbb{S}^n \to X_{n+2}$ on the second row. Using the fact that X_∞ (and hence also X_{n+2}) is n-connected, it is an easy consequence of Lemma 4 that this map is merely constant. As we are proving a proposition, we may ignore the word 'merely' and assume that it is constant. This means that the composition of the two bottom squares is a pushout by the second part of Lemma 2. Consequently, the bottom-right square is also a pushout square. Let us write β for the map $\bigvee_C \mathbb{S}^{n+1} \to \bigvee_B \mathbb{S}^{n+1}$ that is described by the middle column of the diagram. We have shown that $C_\beta \simeq X_{n+2} \vee \bigvee_A \mathbb{S}^{n+1}$. Another way to interpret this equivalence is that we gave the space $X_{n+2} \vee \bigvee_A \mathbb{S}^{n+1}$ a good Hurewicz n-connected CW structure V, with $(n+1)$-skeleton $V_{n+1} = \bigvee_B \mathbb{S}^{n+1}$ and attaching map $\alpha_{n+1} = \beta$.

Consider the inclusion $\mathsf{inr} : \bigvee_A \mathbb{S}^{n+1} \to X_{n+2} \vee \bigvee_A \mathbb{S}^{n+1}$. This happens to be a map between CW complexes, so we may approximate it using Theorem 15. Doing so produces a map $\mathsf{inr}_{n+1} : \bigvee_A \mathbb{S}^{n+1} \to V_{n+1} = \bigvee_B \mathbb{S}^{n+1}$, which factors inr as $\bigvee_A \mathbb{S}^{n+1} \xrightarrow{\mathsf{inr}_{n+1}} \bigvee_B \mathbb{S}^{n+1} \xrightarrow{\iota_{n+1}} X_{n+2} \vee \bigvee_A \mathbb{S}^{n+1}$. Now consider the following diagram.

$$\begin{array}{ccccc} \bigvee_A \mathbb{S}^{n+1} & \xrightarrow{\mathsf{inr}_{n+1}} & \bigvee_B \mathbb{S}^{n+1} & \longrightarrow & X_{n+2} \vee \bigvee_A \mathbb{S}^{n+1} \\ \downarrow & & \downarrow & & \downarrow \\ \mathbb{1} & \longrightarrow & C_{\mathsf{inr}_{n+1}} & \longrightarrow & X_{n+2} \end{array}$$

The left square is a pushout by definition, and the total square is a pushout for elementary reasons. Thus, the right square is a pushout. Replacing $X_{n+2} \vee \bigvee_A \mathbb{S}^{n+1}$ with C_β, we conclude that X_{n+2} is obtained as the pushout of the span $C_{\mathsf{inr}_{n+1}} \leftarrow \bigvee_B \mathbb{S}^{n+1} \to C_\beta$. An application of the 3×3 lemma tells us that this is equivalent to cofibre of the map $\mathsf{inr}_{n+1} \vee \beta : \bigvee_{A+C} \mathbb{S}^{n+1} \to \bigvee_B \mathbb{S}^{n+1}$. Thus, we have shown that X_{n+2} is of the desired form and we are done. □

Corollary 48 (The Hurewicz Approximation Theorem)**.** *A CW complex is n-connected iff it is Hurewicz n-connected.*

B. From homotopy to homology

In order to state our final theorem, we will need the help of the Hurewicz homomorphism. We define it using $\widetilde{H}_n^{\mathsf{cw}}$, but remark that the construction carries over to $\widetilde{H}_n^{\mathsf{str}}$.

Definition 49. *Let X be a CW complex. Define the* ***Hurewicz homomorphism***[4] *$\eta : \pi_n(X) \to \widetilde{H}_n^{\mathsf{cw}}(X)$ on canonical elements $f : \mathbb{S}^n \to_\star X$ by letting $\eta(|f|) : \widetilde{H}_n^{\mathsf{cw}}(X)$ be the image of 1 under the composition $\mathbb{Z} \xrightarrow{\sim} \widetilde{H}_n^{\mathsf{cw}}(\mathbb{S}^n) \xrightarrow{f_*} \widetilde{H}_n^{\mathsf{cw}}(X)$.*

The Hurewicz theorem will provide us with a condition for when this homomorphism is an isomorphism. Before we state and prove it, let us try to understand the groups involved in the 'simple' special case when X is the cofibre C_f of some map of (finite) sphere bouquets $f : \bigvee_A \mathbb{S}^n \to \bigvee_B \mathbb{S}^n$. This special case will turn out to inform the proof for the general case. As C_f has an explicit CW structure, let us switch our homology theory to $\widetilde{H}_n^{\mathsf{str}}$. Now let us compute $\widetilde{H}_n^{\mathsf{str}}(C_f)$ using the exactness axiom: consider the sequence

$$\bigvee_A \mathbb{S}^n \xrightarrow{f} \bigvee_B \mathbb{S}^n \xrightarrow{\mathsf{cfcod}} C_f \xrightarrow{\mathsf{cfcod}} C_{(\mathsf{cfcod}:\bigvee_B \mathbb{S}^n \to C_f)} \simeq \bigvee_A \mathbb{S}^{n+1}$$

where the final equivalence is the usual characterisation of X_{n+1}/X_n using that C_f has a CW structure. This is a cofibre sequence, and so the following sequence is exact

$$\widetilde{H}_n^{\mathsf{str}}\left(\bigvee_A \mathbb{S}^n\right) \xrightarrow{f_*} \widetilde{H}_n^{\mathsf{str}}\left(\bigvee_B \mathbb{S}^n\right) \xrightarrow{\mathsf{cfcod}_*} \widetilde{H}_n^{\mathsf{str}}(C_f) \to 0 \quad (1)$$

where the final 0 comes from the fact that $\widetilde{H}_n^{\mathsf{str}}$ vanishes on $\bigvee_A \mathbb{S}^{n+1}$. We can compute the first two homology groups using additivity, and thus we see that $\widetilde{H}_n^{\mathsf{str}}(C_f) \cong \mathbb{Z}[B]/\mathbb{Z}[A]$. Let us now compute the domain of η, i.e. the group $\pi_n(C_f)$.

Proposition 50. *For any $f : \bigvee_A \mathbb{S}^n \to \bigvee_B \mathbb{S}^n$ where $n \geq 1$ and A and B are finite types, there is an exact sequence*

$$\pi_n\left(\bigvee_A \mathbb{S}^n\right) \xrightarrow{f_*} \pi_n\left(\bigvee_B \mathbb{S}^n\right) \xrightarrow{\mathsf{cfcod}_*} \pi_n(C_f).$$

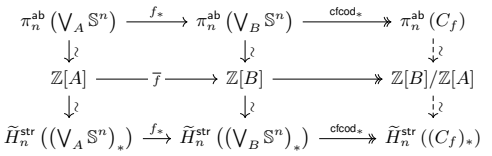
Proof sketch. This follows from the Seifert–Van Kampen theorem [1, Example 8.7.17] in the case $n = 1$, and from the Blakers–Massey theorem [19] in the case $n > 1$. □

We are now almost ready for the Hurewicz theorem. In order to state it, let us define π_n^{ab} to be the abelianisation of the homotopy group functor, i.e. $\pi_n^{\mathsf{ab}}(X) := \pi_n(X)/\operatorname{im}[-,-]$ where $[-,-] : \pi_n(X) \times \pi_n(X) \to \pi_n(X)$ is the commutator defined by $[x,y] = xyx^{-1}y^{-1}$. As higher homotopy groups are already abelian, the quotient map $\pi_n(X) \to \pi_n^{\mathsf{ab}}(X)$ is an isomorphism; in what follows, we will simply interpret π_n^{ab} as π_n when $n \geq 2$. We will, with some abuse of notation, view the Hurewicz homomorphism η as being defined over π_n^{ab}. This is justified as the codomain is an abelian group.

Theorem 51. *The Hurewicz homomorphism $\eta : \pi_n^{\mathsf{ab}}(X) \to \widetilde{H}_n^{\mathsf{cw}}(X)$ is an isomorphism for any $(n-1)$-connected CW complex X.*

Proof. Since we are proving a proposition, we can assume that we have a CW structure X_* and switch our homology theory to $\widetilde{H}_*^{\mathsf{str}}$. Since the map $X_{n+1} \to X_\infty$ is n-connected, the canonical map $\pi_n(X_{n+1}) \to \pi_n(X_\infty)$ is an equivalence. Similarly, $\widetilde{H}_n^{\mathsf{str}}(X_*) = \widetilde{H}_n^{\mathsf{str}}(X_*^{(n+1)})$ by definition. Thus, it suffices to show the theorem for the $(n+1)$-skeleton of X. As X is Hurewicz $(n-1)$-connected, we may assume that $X_n = \bigvee_B \mathbb{S}^n$ and that $X_{n+1} = C_f$ for some $\alpha : \bigvee_A \mathbb{S}^n \to \bigvee_B \mathbb{S}^n$. Elementary algebra tells us that abelianisation is right-exact and thus preserves the exact sequence in Proposition 50. Let us compare this sequence (top sequence below) to the corresponding one for homology in (1) (bottom sequence below).

[4]The fact that this map is a homomorphism boils down to the easy fact that the (group) addition of cellular maps $\mathbb{S}^n \to_\star X$ is again cellular.



The isomorphisms $\pi_n^{\mathsf{ab}}(\bigvee_C \mathbb{S}^n) \cong \mathbb{Z}[C]$ for $C \in \{A, B\}$ are easily constructed using the Seifert–Van Kampen theorem [1, Example 8.7.17] when $n = 1$ and the Blakers–Massey theorem [19] when $n > 1$. On homology, the isomorphism is a direct consequence of the Eilenberg–Steenrod axioms (but can also be obtained by simply inspecting the related chain complex). The fact that the two left-most squares commute holds almost by definition of the maps involved. Hence we obtain an isomorphism $\pi_n^{\mathsf{ab}}(C_f) \cong \widetilde{H}_n^{\mathsf{str}}((C_f)_*)$. We simply have to verify that this isomorphism is equal to η. It is enough to check this on the inclusion of generators from $\pi_n^{\mathsf{ab}}(\bigvee_B \mathbb{S}^n)$ – but here there is nothing to prove: simply unfolding the definitions involved, it is immediate that the desired equality holds. □

VI. Conclusions and Future Work

We hope the reader is now convinced that the theory of CW complexes and cellular homology has a home in HoTT. The fact that the results we have proved in this paper – in particular the approximation theorems – are at all provable without any form of choice was initially a surprise to us. The theory of CW complexes and cellular homology as it is developed classically often 'feels' constructive, with many constructions being inductive, but it makes heavy use of choice principles. An important takeaway is that this feeling is justified: a significant part of this theory *is* constructive.

However, the initial motivation behind this project was not to carry out a case study in constructive mathematics. Originally, our development was motivated by the recent proof of the Serre Finiteness theorem by Barton and Campion [20]. This proof relies on homology computations and the Hurewicz theorem, thus the formalisation that accompanies this paper should be helpful to the ongoing formalisation of the Serre Finiteness theorem (by, in particular, Milner [21]).

This paper also aims to be integrated into a larger project including Mörtberg, which seeks to use cellular (co)homology to reduce homological arguments in HoTT to concrete computations which we can run in proof assistants. The canonical example is the computation of the *Brunerie number* [2], a number whose value is given by a certain cohomology computation which, as it is constructively defined in HoTT, should simply be produced by evaluating it in a proof assistant, but whose evaluation is computationally infeasible. Our hope is that if these computations are ported to a cellular (co)homology theory, where many of them should become feasible, paving the way for *proofs by computation* in HoTT.

It would also be interesting to use our cellular approach to explore more advanced results and constructions such as the Steenrod squares and the (currently open) Künneth formula.

References

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

[2] 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

[3] P. L. Lumsdaine and M. Shulman, "Semantics of higher inductive types," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 169, no. 1, pp. 159–208, 2020.

[4] M. Shulman. (2019, April) All $(\infty, 1)$-toposes have strict univalent universes. Preprint. [Online]. Available: https://arxiv.org/abs/1904.07004

[5] 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

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

[7] D. C. Christensen and L. Scoccola, "The Hurewicz theorem in homotopy type theory," *Algebraic & Geometric Topology*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:220496177

[8] E. Rijke, "Introduction to homotopy type theory," 2022. [Online]. Available: https://arxiv.org/abs/2212.11082

[9] K. Sojakova, F. v. Doorn, and E. Rijke, "Sequential colimits in homotopy type theory," in *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, ser. LICS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 845–858. [Online]. Available: https://doi.org/10.1145/3373718.3394801

[10] 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.

[11] G. Brunerie, A. Ljungström, and A. Mörtberg, "Synthetic Integral Cohomology in Cubical Agda," in *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), F. Manea and A. Simpson, Eds., vol. 216. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, pp. 11:1–11:19. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2022/15731

[12] T. Lamiaux, A. Ljungström, and A. Mörtberg, "Computing cohomology rings in cubical agda," in *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs*, ser. CPP 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 239–252. [Online]. Available: https://doi.org/10.1145/3573105.3575677

[13] A. Ljungström and A. Mörtberg, " Formalizing $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$ and Computing a Brunerie Number in Cubical Agda ," in *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2023, pp. 1–13. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/LICS56636.2023.10175833

[14] J. May, *A Concise Course in Algebraic Topology*, ser. Chicago Lectures in Mathematics. University of Chicago Press, 1999.

[15] P. Cagne, U. T. Buchholtz, N. Kraus, and M. Bezem, "On symmetries of spheres in univalent foundations," in *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science*, ser. LICS '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3661814.3662115

[16] S. Eilenberg and N. E. Steenrod, "Axiomatic approach to homology theory," *Proceedings of the National Academy of Sciences*, vol. 31, no. 4, pp. 117–120, 1945. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.31.4.117

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

[18] N. Kraus, "Truncation levels in homotopy type theory," Ph.D. dissertation, University of Nottingham, 2015. [Online]. Available: https://eprints.nottingham.ac.uk/28986/1/thesis.pdf

[19] 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.

[20] R. Barton, "Finite presentability of homotopy groups of spheres," 2022, talk at the Seminar on Homotopy Type Theory at CMU, presenting joint work with Tim Campion. [Online]. Available: https://www.cmu.edu/dietrich/philosophy/hott/seminars/previous.html

[21] O. Milner, "Formalizing the whitehead tower in cubical agda," 2023, talk at the Seminar on Homotopy Type Theory at CMU. [Online]. Available: https://lc2023.unimi.it/wp-content/uploads/2023/05/milner.pdf

Paper V

The Steenrod squares via unordered joins

Axel Ljungström
Department of Mathematics
Stockholm University
Email: axel.ljungstrom@math.su.se

David Wärn
Department of Computer Science and Engineering
University of Gothenburg and Chalmers University of Technology
Email: warnd@chalmers.se

***Abstract*—The Steenrod squares are cohomology operations with important applications in algebraic topology. While these operations are well-understood classically, little is known about them in the setting of homotopy type theory. Although a definition of the Steenrod squares was put forward by Brunerie (2017), proofs of their characterising properties have remained elusive. In this paper, we revisit Brunerie's definition and provide proofs of these properties, including stability, Cartan's formula and the Adem relations. This is done by studying a higher inductive type called the unordered join. This approach is inherently synthetic and, consequently, many of our proofs differ significantly from their classical counterparts. Along the way, we discuss upshots and limitations of homotopy type theory as a synthetic language for homotopy theory. The paper is accompanied by a computer formalisation in Cubical Agda.**

I. Introduction

Homotopy type theory (HoTT) is an extension of Martin-Löf type theory based on the idea of treating *types* as ∞-groupoids, or *spaces*. While HoTT only gained attention as recently as 2012, ∞-groupoids themselves are important mathematical objects and have long been fruitfully studied using the many tools of algebraic topology and homotopy theory. A key question is to what extent these tools can be made to work with the language of HoTT, and whether HoTT can provide new insights going beyond classical homotopy theory. By now, there is an established line of research, dubbed *synthetic homotopy theory*, dedicated to answering these questions. The promises of synthetic homotopy theory include conceptual clarity, semantic generality (an argument expressed in HoTT automatically applies in many models, including arbitrary ∞-topoi [1]), and amenability to computer formalisation, but it also comes with its own set of limitations.

A fundamental tool in homotopy theory is that of *cohomology*, and a fundamental tool in making sense of cohomology is that of *cohomology operations*. These are ways of constructing new cohomology classes from old ones, and they give the cohomology of any space a rich structure. The purpose of this paper is to study an important family of cohomology operations, the *Steenrod squares*, in HoTT. Although a good deal of work has been done setting up the foundations of cohomology in synthetic homotopy theory, from Eilenberg–MacLane spaces [2], cohomology groups [3], cup products [4], [5], and cellular cohomology [6], to Gysin sequences [7] and spectral sequences [8], the Steenrod squares have so far only been *defined* in HoTT in a short text by Brunerie [9]. The Steenrod squares are classically known to satisfy a list of properties that are not easily read off from their definition but are important for applications, and these properties have remained elusive in synthetic homotopy theory. In this work, we will prove all these properties.

We have computer formalised a large part of the project (including all key technical results) in `Cubical Agda`, a proof assistant implementing a flavour of HoTT called cubical type theory. We emphasise, however, that this paper is agnostic with respect to HoTT flavour and is written in the implementation-agnostic informal style of the HoTT book [10].

Two themes feature prominently in this work. The first theme is that of constructions which depend on an arbitrary 2-element type [9], [11]. Here, a type is said to be a 2-element type if it *merely* is equivalent to the standard 2-element type 2 (i.e. $\{0,1\}$). We denote the type of all 2-element types by $\mathbb{R}P^\infty$. By univalence, any construction that depends on an arbitrary 2-element type (i.e. indexed by $\mathbb{R}P^\infty$) automatically respects automorphisms of 2-element types, and in this way we get a synthetic approach to what is classically known as equivariant homotopy theory. The second theme is that of higher inductive types; of particular importance to us are joins and smash products. The meat of our work consists of studying the interaction of these two themes: unordered joins and smash products and their properties.

To set the stage and state the main result of this paper, let us briefly revisit the work of Brunerie. Brunerie defines the nth Steenrod square, a map $H^m(X,\mathbb{Z}/2\mathbb{Z}) \to H^{m+n}(X,\mathbb{Z}/2\mathbb{Z})$, directly on Eilenberg–MacLane spaces: in HoTT, elements of $H^m(X,\mathbb{Z}/2\mathbb{Z})$ are represented by functions $X \to K_m$ where $K_m \coloneqq K(\mathbb{Z}/2\mathbb{Z},m)$ denotes the mth Eilenberg–MacLane space of $\mathbb{Z}/2\mathbb{Z}$, and so the Steenrod square is given by a map $\mathsf{Sq}^n : K_m \to K_{m+n}$. This map is defined as a composition:

$$K_m \to (\mathbb{R}P^\infty \to K_{2m}) \xrightarrow{\sim} \Pi_{i\leq 2m} K_i \xrightarrow{\mathsf{proj}_{m+n}} K_{m+n}. \quad (1)$$

Here, the first map is defined using *unordered smash products*, and the second equivalence comes from the Thom isomorphism theorem [7, Section 6.1]. While this construction is elegant, it has turned out to be difficult to analyse. In particular, one would like to know that the Steenrod squares satisfy the properties listed below; proving these is the primary contribution of this paper.

Theorem 1 (The Steenrod squares, axiomatically)**.** *There is a set of pointed maps* $\mathsf{Sq}^n_m : K_m \to_{\mathsf{pt}} K_{m+n}$ *for* $m,n \geq 0$,

called the Steenrod squares, usually written Sq^n *leaving the* m *implicit, which satisfy the following identities.*

(I1) $\mathsf{Sq}^0(x) = x$
(I2) $\mathsf{Sq}^n_m(x) = 0$ *if* $n > m$
(I3) $\mathsf{Sq}^n_n(x) = x \smile x$
(C) $\mathsf{Sq}^n(x \smile y) = \sum_{i+j=n} \mathsf{Sq}^i(x) \smile \mathsf{Sq}^j(y)$ *(the Cartan formula)*

In addition, the squares are stable and satisfy the Adem relations:

(Ω) *The* n*th square* $\mathsf{Sq}^n_m : K_m \to_{\mathsf{pt}} K_{m+n}$ *is also given by*

$$K_m \xrightarrow{\sim} \Omega(K_{m+1}) \xrightarrow{\Omega(\mathsf{Sq}^n_{m+1})} \Omega(K_{(m+1)+n}) \xrightarrow{\sim} K_{m+n}.$$

(A) *The Adem relations are satisfied: for* $n < 2k$*, we have*

$$\mathsf{Sq}^n \circ \mathsf{Sq}^k = \textstyle\sum_{i=0}^{\lfloor n/2 \rfloor} \binom{k-i-1}{n-2i} \mathsf{Sq}^{n+k-i} \circ \mathsf{Sq}^i.$$

In (I3) and (C) above, the symbol $\smile$ denotes the cup product map $K_m \to_{\mathsf{pt}} K_n \to_{\mathsf{pt}} K_{m+n}$ obtained from the ring structure on $\mathbb{Z}/2\mathbb{Z}$.

A. Contributions and outline

The main contribution of this paper is a proof of the properties of Steenrod squares listed in Theorem 1. Before discussing Steenrod squares, we discuss, in Section II, some background material about $\mathbb{R}P^\infty$ and unordered pairs. Then in Section III, we present a variant of Brunerie's definition of the Steenrod squares, via what we call unordered cup products. We prove all the properties of Steenrod squares listed in Theorem 1, modulo a technicality dealt with in the following section. Section IV concerns properties of unordered HITs and constitutes the technical core of this paper. In Section V we discuss how a good theory of E_∞-monoids in type theory would have significantly simplified our work. In Section VI we showcase an application of Steenrod squares toward analysing $\pi_4(\mathbb{S}^3)$. Finally, in Section VII, we mention some open questions.

B. Notation and basic definitions

Let us briefly introduce some notation while also recalling some basic constructions from HoTT which will be used in the paper. The reader familiar with HoTT should be able to safely skim or even skip this part.

a) Pi- and sigma-types: Let $B : A \to \mathcal{U}$ be a dependent type – here $\mathcal{U}$ denotes the universe of types (at some implicit universe level). We often use the notation $(a : A) \to B\,a$ and $(a : A) \times B\,a$ for $\Pi_{a:A} B(a)$ and $\Sigma_{a:A} B(a)$ respectively. We may sometimes write B^A for the non-dependent function type $A \to B$.

b) Equality: We write $x = y$ for the type of paths from x to y. We use $x :\equiv y$ for definitions. The constant path (reflexivity) is denoted by $\mathsf{refl}_x : x = x$. We use *path induction* to refer to the usual induction principle for identity types in MLTT.

c) Equivalences and univalence: A type X is said to be *contractible* if there is some $x : X$ s.t. for all $x' : X$, we have that $x' = x$. Given a map $f : A \to B$ we define its fibre over some point $b : B$ by $\mathsf{fib}_f(b) :\equiv (a : A) \times (f(a) = b)$. We say that f is an equivalence if $\mathsf{fib}_f(b)$ is contractible for each $b : B$. In this case, we simply write $f : A \simeq B$ and leave the contractibility proof implicit.

There is a canonical map $\mathsf{coe} : X = Y \to X \simeq Y$ defined by path induction, sending refl_X to the identity $\mathsf{id}_X : X \simeq X$. The univalence axiom says that coe itself is an equivalence. In particular, this means that if two types are equivalent, then they are equal.

d) Pointed structures: A pointed type (A, pt_A), i.e. a type A equipped with a basepoint $\mathsf{pt}_A : A$, will often simply be written A, i.e. with the basepoint left implicit. We use the same convention for pointed functions and simply write $f : A \to_{\mathsf{pt}} B$ to mean a pair (f, pt_f) where $f : A \to B$ is a plain function and $\mathsf{pt}_f : f(\mathsf{pt}_A) = \mathsf{pt}_B$ is a proof that f is basepoint preserving.

e) Loop spaces: We define the *loop space* of a pointed type A by $\Omega(A) :\equiv (\mathsf{pt}_A = \mathsf{pt}_A)$. This construction is itself pointed by $\mathsf{refl}_{\mathsf{pt}_A}$ and can thus be iterated by inductively defining $\Omega^{n+1}(A) :\equiv \Omega^n(\Omega(A))$.

f) H-levels: We say that a type A is a (-2)*-type* if it is contractible and, inductively, that it is an n-type if $x = y$ is an $(n-1)$-type for all $x, y : A$. Apart from (-2)-types, special names are also given to (-1)-types and 0-types; these are, respectively, called *propositions* and *sets*.

g) Truncations: We write $\|A\|_n$ for the n*-truncation of* A, the canonical way of forcing A to become an n-type. This is a type equipped with an inclusion of points $|-| : A \to \|A\|_n$ whose induction principle say that the map $((x : \|A\|_n) \to B(x)) \to ((a : A) \to B\,|a|)$ is an equivalence whenever $B : \|A\|_n \to \mathcal{U}$ is a family of n-types. It is implemented using a recursive HIT [10, Section 7.3], but we do not need the implementation details here.

h) Connectedness: We say that a type is n-connected if $\|A\|_n$ is contractible. We say that a function $f : A \to B$ is n-connected if all of its fibres are n-connected.

i) Pushouts: Given a span $Y \xleftarrow{f} X \xrightarrow{g} Z$, we define its (homotopy) pushout, $Y \sqcup^X Z$, to be the HIT generated by two point constructors $\mathsf{inl} : Y \to Y \sqcup^X Z$ and $\mathsf{inr} : Z \to Y \sqcup^X Z$, as well as one higher constructor $\mathsf{push} : (x : X) \to \mathsf{inl}(f\,x) = \mathsf{inr}(g, x)$. An important special case of pushouts is the *suspension* of a type X, written ΣX, which we define by $\Sigma X :\equiv \mathbb{1} \sqcup^X \mathbb{1}$. Another important construction is the *smash product* of two pointed types, denoted $X \wedge Y$. We define it here as the pushout $(\mathbb{1} + \mathbb{1}) \sqcup^{X+Y} X \times Y$. The *join* of types is another example of a pushout which plays a key role for us. The join of $X * Y$ of types X, Y is defined to be the pushout $X \sqcup^{X \times Y} Y$ of the span $X \xleftarrow{\mathsf{fst}} X \times Y \xrightarrow{\mathsf{snd}} Y$. We take all pushouts to be pointed, whenever possible, by $\mathsf{inl}(\mathsf{pt}_X)$.

j) Eilenberg–MacLane spaces and cohomology: Given an abelian group G and a natural number n, we denote by $K(G, n)$ the nth *Eilenberg–MacLane space*, or *delooping*,

of G [2]. It is characterised as the unique pointed $(n-1)$-connected n-type whose nth loop space $\Omega^n K(G,n)$ is isomorphic, as a group, to G. It follows that we have pointed equivalences $\sigma_n : K(G,n) \simeq_{\mathsf{pt}} \Omega K(G,n+1)$. In this way it is clear that $K(G,n)$ has an associative and commutative H-space structure (corresponding to path composition in $\Omega K(G,n+1)$), denoted $+ : K(G,n) \to K(G,n) \to K(G,n)$.

If moreover R is a ring, then we have a *cup product* $\smile$: $K(R,n) \to_{\mathsf{pt}} K(R,m) \to_{\mathsf{pt}} K(R,n+m)$ which is graded-commutative and associative. The defining property of the cup product is that in degree $(0,0)$, i.e. as a map $K(R,0) \to K(R,0) \to K(R,0)$, it simply corresponds to multiplication in R, and that the cup product respects looping in the following sense. For a fixed $a : K_n$, consider the pointed map given by cupping with a, i.e. $(-) \smile_m a : K_m \to_{\mathsf{pt}} K_{m+n}$. We have $\Omega((-) \smile_{m+1} a) \circ \sigma_m = \sigma_{m+n} \circ ((-) \smile_m a)$. This is proved in detail in [5, Lemma 29].

The nth cohomology group of a type X with coefficients in an abelian group G is given by $H^n(X,G) :\equiv \|X \to K(G,n)\|_0$. The types $H^\bullet(X,G)$ are abelian groups by pointwise addition in $K(G,n)$, and moreover form a graded ring if G has the structure of a ring.

II. Unordered pairs and commutativity structures

One of the main themes in this paper, crucial for the treatment of Steenrod squares, is that of constructions that depend on 2-element type. Recall that we write 2 for the standard 2-element type, with two distinct elements 0 and 1. A general type X is said to be a 2-element type if we have $\|X \simeq 2\|$. Thus any 2-element type is merely equivalent to 2, but it has no preferred enumeration. Recall also that we write $\mathbb{R}P^\infty$ for the type of all 2-element types, or more explicitly

$$\mathbb{R}P^\infty :\equiv (X : \mathcal{U}) \times \|X \simeq 2\|.$$

We treat $\mathbb{R}P^\infty$ as a pointed type with basepoint $(2, |\mathsf{id}_2|)$. Buchholtz and Rijke [12] have shown that $\mathbb{R}P^\infty$ is the sequential colimit of the finite-dimensional real projective spaces $\mathbb{R}P^n$, hence the notation. Given a term $X : \mathbb{R}P^\infty$ we will conflate X with its underlying type $\mathsf{fst}(X) : \mathcal{U}$.

For us, $\mathbb{R}P^\infty$ is significant because it captures the idea of commutativity in a synthetic and homotopy coherent manner. Consider, for example, the symmetry of cartesian products, $A_0 \times A_1 \simeq A_1 \times A_0$. This can be explained as follows using $\mathbb{R}P^\infty$. The binary cartesian product can be seen as a 2-indexed dependent product, $\Pi_{i:2} A(i)$ for $A : 2 \to \mathcal{U}$. More generally, for any $X : \mathbb{R}P^\infty$ and $A : X \to \mathcal{U}$, we can consider the product $\Pi_{i:X} A(i)$. Clearly this reduces to the binary cartesian product if X is 2. Now 2 has a self-equivalence $\neg : 2 \to 2$ which swaps 0 and 1, and by univalence this induces a loop $\mathsf{pt} = \mathsf{pt}$ of the basepoint of $\mathbb{R}P^\infty$. By action on paths, this induces an equivalence $\Pi_{i:2} A(i) \simeq \Pi_{i:2} A(\neg i)$ for any $A : 2 \to \mathcal{U}$, which reduces to commutativity of the cartesian product.

In general, our usage of $\mathbb{R}P^\infty$ follows the same pattern. The point is to first generalise some construction which normally would operate on ordered pairs – like the cartesian product $A_0 \times A_1$ computed from the pair $(A_0, A_1) : \mathcal{U} \times \mathcal{U}$ – to a construction that depends on an 'unordered pair'. By an unordered pair (of elements of A), we mean a map $a : X \to A$ where $X : \mathbb{R}P^\infty$ and A is some arbitrary type. We often write A^X to emphasise that it should be thought of as a generalisation of A^2 – one might also say that A^X is a 'twisted' version of A^2.

The upshot is that whenever we write down a construction indexed by $\mathbb{R}P^\infty$, we automatically gain information about the symmetry of said construction. In the case of the cartesian product, this gives rise to the equivalence $\mathsf{swap}_{A_0,A_1} : A_0 \times A_1 \simeq A_1 \times A_0$, but this is not all. The fact that the self-equivalence $\neg : 2 \simeq 2$ is involutive tells us that $\mathsf{swap}_{A_1,A_0} \circ \mathsf{swap}_{A_0,A_1} = \mathsf{id}$. This is only the start of an infinite tower of coherences, associated with the cell decomposition of $\mathbb{R}P^\infty$. The upshot of the synthetic approach is that we do not need to think explicitly about these coherences.

A. Basic facts about $\mathbb{R}P^\infty$

Let us now recall some elementary lemmas and constructions regarding $\mathbb{R}P^\infty$ and unordered pairs. The following lemma is a special case of a result due to Kraus [13, Proposition 8.1.2.] and the remaining ones can be found in Buchholtz and Rijke [12].

Lemma 2. *For $P : \mathbb{R}P^\infty \to \mathcal{U}$, the type of functions $(X : \mathbb{R}P^\infty) \to P(X)$ is equivalent to*

(a) $P(2)$ if P is proposition-valued.
(b) $(t : P(2)) \times (P(\neg)(t) = t)$ if P is set-valued.

Lemma 3. *All types $X : \mathbb{R}P^\infty$ are sets.*

Lemma 4. *For any $X : \mathbb{R}P^\infty$, there is an involution $\neg : X \simeq X$ agreeing with the usual involution of 2 whenever $X :\equiv 2$.*

Although this lemma/definition is well known, its proof illustrates a useful technique for defining operations over $\mathbb{R}P^\infty$, so we choose to include it.

Proof/construction of Lemma 4. For any $X : \mathbb{R}P^\infty$, let $P(X) :\equiv (e : X \to X) \times ((e \circ e = \mathsf{id}) \times e \neq \mathsf{id}_X)$. We claim that this type is contractible. Since this claim is a proposition, it suffices, by Lemma 2, to show that $P(2)$ is contractible. This is trivial, as $P(2)$ is the type of non-identity involutions on 2 – a type uniquely pointed by 2-involution. So, $P(X)$ is contractible for any $X : \mathbb{R}P^\infty$ and we define $\neg$ to be the centre of contraction. □

Using Lemma 4, we may construct, for any $X : \mathbb{R}P^\infty$ and $x : X$, an equivalence $e^x : 2 \simeq X$ defined by setting

$$e^x(0) :\equiv x \qquad e^x(1) :\equiv \neg x.$$

To prove that this indeed is an equivalence, we note that this statement is a proposition and thus, by Lemma 2, it suffices to do so when X is 2. By case-splitting on $x : 2$, we see that e^x is the identity when $x = 0$ and involution when $x = 1$. In

particular, it is an equivalence. In fact, not only is e^x always an equivalence – the map $e^{(-)}$ is one itself:

Lemma 5. *For any $X : \mathbb{R}P^\infty$, the map $e^{(-)} : X \to (\mathbb{2} \simeq X)$ is an equivalence.*

Proof. The statement is a proposition, and thus it suffices to show it when $X = \mathbb{2}$. In this case, $e^{(-)}$ is the map sending 0 to the identity on $\mathbb{2}$ and sending 1 to the involution. As these are precisely the (two) $\mathbb{2}$-automorphisms, $e^{(-)}$ is clearly invertible and thus an equivalence. □

By univalence, Lemma 5 gives a characterisation of the based path types on $\mathbb{R}P^\infty$: it tells us that any based path type $(\mathrm{pt}_{\mathbb{R}P^\infty} = X)$ is equivalent to the 'point' X itself. In particular, we get that unordered pairs (A^X) really corresponds to fibrations over the based path types of $\mathbb{R}P^\infty$, i.e. $(\mathrm{pt}_{\mathbb{R}P^\infty} = X \to A)$. This gives us a new way of interpreting path induction for $\mathbb{R}P^\infty$:

Lemma 6. *Let $A : (X : \mathbb{R}P^\infty) \times X \to \mathcal{U}$. The map*

$$(((X, x) : (\dots)) \to A(X, x)) \xrightarrow{f \mapsto f(\mathbb{2},0)} A(\mathbb{2}, 0)$$

is an equivalence.

Another way of understanding this is by the following induction rule for functions defined over $X : \mathbb{R}P^\infty$.

Lemma 7. *Let $X : \mathbb{R}P^\infty$, $B : X \to \mathcal{U}$ and $x : X$. Any pair of points $b_0 : B(x)$ and $b_1 : B(\neg x)$ induces a function*

$$\mathsf{Elim}^{x \mapsto b_0}_{\neg x \mapsto b_1} : (x : X) \to B(x)$$

satisfying $\mathsf{Elim}^{x \mapsto b_0}_{\neg x \mapsto b_1}(x) = b_0$ and $\mathsf{Elim}^{x \mapsto b_0}_{\neg x \mapsto b_1}(\neg x) = b_1$. In fact, the map $B(x) \times B(\neg x) \xrightarrow{(b_0, b_1) \mapsto \mathsf{Elim}^{x \mapsto b_0}_{\neg x \mapsto b_1}} \Pi_{x:X} B(x)$ is an equivalence.

B. Commutativity Structures

As discussed, the significance of unordered pairs is that they allow us to capture the idea of an operation being homotopy commutative in an 'infinitely coherent' manner. This is captured by the following definition.

Definition 8 (Brunerie [9])**.** A *commutativity structure* for a binary operation $\diamond : A \times A \to B$ is a family of maps $\diamond_X : A^X \to B$ for each $X : \mathbb{R}P^\infty$ agreeing with $\diamond$ if $X = \mathbb{2}$.

By letting A and B be Eilenberg-MacLane spaces in the above definition, a commutativity structure $\diamond_{(-)}$ will allow us to produce cohomology classes in $H^*(\mathbb{R}P^\infty)$. Brunerie's construction of the Steenrod squares boils down to showing that the cup product $\smile : K_n \times K_n \to K_{2n}$ has a commutativity structure. Before we get there, however, let us give the following example in order to illustrate the general idea of how commutativity structures can be constructed. In fact, the following construction will be useful in its own right.

Example 9. For any commutative monoid $(M, +, 0)$, addition $+ : M \times M \to M$ has a commutativity structure. Since M is a monoid, it is a set and thus the type of maps $M^X \to M$ is a set for any X. We will define the commutativity structure, denoted by $\Sigma : M^X \to M$ for $X : \mathbb{R}P^\infty$, using Lemma 2(b). For $X \coloneqq \mathbb{2}$, we define $\Sigma f \coloneqq f(0) + f(1)$. We then need to check that this definition is invariant under $\mathbb{2}$-inversion. This corresponds to checking that $f(0)+f(1) = f(1)+f(0)$ which of course follows from commutativity of M.

The construction in Example 9 crucially relied on M being a set; when constructing commutativity structures in general, there are not that many other methods than this at hand. Fortunately, this argument can sometimes still be used in cases when the h-level of the type of commutativity structures is not zero:

Example 10. Addition on Eilenberg–MacLane spaces $+ : K_n \times K_n \to K_n$ has a commutativity structure. To see why, we consider the family of dependent types $P_X : (K_n^X \to K_n) \to \mathcal{U}$ defined by $P_X(f) \coloneqq (f(\lambda x\,.0) = 0)$. We have

$$(f : K_n \times K_n \to K_n) \times (P_2(f)) = (K_n \times K_n \to_{\mathrm{pt}} K_n)$$

which is a set [14, Corollary 9]. This means that Lemma 2(b) applies. It thus suffices to provide an element of $P_2(+)$ and check that this choice is invariant w.r.t. involution of $\mathbb{2}$. This boils down to verifying the commutativity of $+$.

This idea of defining a predicate over the function type of interest which forces it to become a set is present also in Brunerie's original definition of a commutativity structure for the cup product. Although he does not state it exactly this way, Brunerie implicitly considers the following predicate.

Definition 11. Let $X : \mathbb{R}P^\infty$, $A : X \to \mathcal{U}_{\mathrm{pt}}$ and $B : \mathcal{U}_{\mathrm{pt}}$, and $f : (\prod_{x:X} A(X)) \to B$. We define $\mathsf{isBiHom}_X(f) : \mathcal{U}$ to be the following type expressing that f is 'pointed in each argument':

$$(f(\lambda\, x\,.\,\mathsf{pt}) = \mathsf{pt}) \times (\mathsf{pts} : B^X)$$
$$\times \left(\left(a : \prod_{x:X} A(x)\right)(x : X) \to (a(x) = \mathsf{pt}) \to f(a) = \mathsf{pts}(x)\right)$$

$$\text{Let } \mathsf{BiHom}_X(A, B) \coloneqq (f : \textstyle\prod_{x:X} A(x) \to B) \times \mathsf{isBiHom}_X(f).$$

A straightforward rewriting shows that, for any proof of $\mathsf{isBiHom}_X(f)$, its pts component is constantly $\mathrm{pt} : B$, and we have

$$(f : A_0 \times A_1 \to B) \times \mathsf{isBiHom}_2(f) \simeq (A_0 \wedge A_1 \to_{\mathrm{pt}} B).$$

By setting $A(x) = K_n$ and $B = K_{2n}$ in Definition 11, $\mathsf{isBiHom}_X$ is a predicate on the function type $K_n^X \to K_{2n}$. Let us construct such a function. The key observation is that the type of such functions is equivalent to $K_n \wedge K_n \to_{\mathrm{pt}} K_{2n}$ whenever $X = \mathbb{2}$. This turns out to be a set [14, Corollary 9], and thus the type of such function is a set for any $X : \mathbb{R}P^\infty$. Like in Examples 9 and 10, it is enough to give the construction when $X = \mathbb{2}$ and check that it is commutative. Since the cup with coefficients mod 2 is commutative, it has a commutativity structure.

This concludes (our take on) Brunerie's definition of the commutativity structure on the cup product. While it is certainly sufficient for constructing the Steenrod squares, it has turned out to be rather hard to reason about. One simple but crucial reason for this is that Brunerie's definition does not quite capture a key fact about the cup product, namely that it is *graded*. The main issue we have is that our notion of a commutativity structure does not allow for dependent types. To remedy this, we propose the following definition.

Definition 12. Let $A : I \to \mathcal{U}$ be a family of types where I is a commutative monoid (e.g. $I = \mathbb{N}$). A graded commutativity structure for a graded operation $\diamond : A_i \times A_j \to A_{i+j}$ is a family of maps $\diamond_{X,n} : (\Pi_{x:X} A_{n(x)}) \to A_{\Sigma n}$ for each $X : \mathbb{R}P^\infty$ and $n : X \to I$, which reduces to $\diamond$ for $X :\equiv \mathbb{2}$. We remind the reader of the definition of Σn from Example 9.

Remark 13. Since we work modulo 2 throughout, we have no reason to worry about the signs that normally show up when discussing graded commutativity of e.g the cup product, but let us make a comment about how they can be dealt with. For a group G and a finite type $\mathbf{n}$ of n elements, one can define a pointed type $K(G, \mathbf{n})$, which is like $K(G, n)$ but with a 'twist' relating odd permutations of $\mathbf{n}$ with the involution of $K(G, n)$ given by negation. If G then is a commutative ring, the corresponding graded commutativity structure on $K(G, -)$ would be given by maps $\Pi_{x:X} K(G, \mathbf{n}(x)) \to K(G, \sum_{x:X} \mathbf{n}(x))$ for $X : \mathbb{R}P^\infty$ and $\mathbf{n} : X \to \mathcal{U}$ a family of finite types. The key here is to index not by $\mathbb{N}$ but by the type of finite sets, or some other higher type which records information about twists.

Our construction of the commutativity structure on the cup product can be restated, word by word, to equip it with a graded commutativity structure. This slight generalisation of Brunerie's definition will be the one used in this paper. For this reason, let us finish this section by giving it a name.

Definition 14. The cup product has a graded commutativity structure which we will refer to as the **unordered cup product**. For $X : \mathbb{R}P^\infty$, $n : X \to \mathbb{N}$ and $f : (x : X) \to K_{n(x)}$, we denote it by $\smile_{x:X} f(x) : K_{\Sigma n}$.

III. The Steenrod squares

We are now well-prepared to define the Steenrod squares. We follow Brunerie's approach, as laid out in (1). That is, we need to define two maps: one of type $K_m \to (\mathbb{R}P^\infty \to K_{2m})$ and one (equivalence) of type $(\mathbb{R}P^\infty \to K_{2m}) \xrightarrow{\sim} \prod_{i \leq 2m} K_i$. Let us start with the first map. Suppose we are given $a : K_m$ and $X : \mathbb{R}P^\infty$. We let $n : X \to \mathbb{N}$ and $\widehat{a} : (x : X) \to K_{n(x)}$ be the constant functions $n(x) :\equiv m$ and $\widehat{a}(x) :\equiv a$. We may now define $a^X : K_{2m}$ by

$$a^X :\equiv \smile_{x:X} \widehat{a}(x). \tag{2}$$

The notation is meant to suggest that we think of a^X as the cup product of X-many copies of a; traditionally, this may also be written as $S(a, X)$.

The second map is given by the inverse equivalence in the following lemma.

Lemma 15. *For $n : \mathbb{N}$, we have an equivalence*

$$\mathsf{Gys}_n : \prod_{i \leq n} K_i \simeq (\mathbb{R}P^\infty \to K_n)$$

$$\mathsf{Gys}_n(b_0, \dots, b_n) :\equiv X \mapsto \sum_{i=0}^{n} b_i \smile t(X)^{n-i}.$$

Here, t denotes the unique pointed equivalence $\mathbb{R}P^\infty \xrightarrow{\sim}_{\text{pt}} K_1$, and $t(X)^{n-i}$ denotes the iterated cup product of $t(X)$ with itself. One can think of Lemma 15 as describing the mod 2 cohomology of $\mathbb{R}P^\infty$, but more directly it says that every map $\mathbb{R}P^\infty \to K_n$ has a unique 'polynomial' representation. Before proving Lemma 15, we first have to state two simpler lemmas.

Lemma 16. *For $n : \mathbb{N}$, we have an equivalence*

$$(\mathbb{R}P^\infty \to K_n) \simeq (\mathbb{R}P^\infty \to_{\text{pt}} K_{n+1})$$
$$f \mapsto X \mapsto t(X) \smile f(X)$$

Lemma 16 is proved using the Thom isomorphism [7, Section 6.1]. For details, see [5, Section 5.5].

Lemma 17. *For any invertible H-space B and pointed type A, we have an equivalence*

$$B \times (A \to_{\text{pt}} B) \simeq A \to B$$
$$(b, f) \mapsto a \mapsto b + f(a)$$

Proof. We have

$$\begin{aligned} B \times (A \to_{\text{pt}} B) &\simeq (b : B) \times (A \to_{\text{pt}} (B, b)) \\ &\simeq (f : A \to B) \times (b : B) \times (f(a) = b) \\ &\simeq (A \to B) \end{aligned}$$

where the first step comes from the fact that B is an invertible H-space (and hence homogeneous) and the second from the contractibility of singletons. This equivalence agrees with the proposed one by construction. □

We are now ready to prove Lemma 15.

Proof of Lemma 15. By induction. For $n = 0$, this is simply the statement that any map $\mathbb{R}P^\infty \to K_0$ is constant, which follows from connectedness of $\mathbb{R}P^\infty$. Now, suppose the lemma holds for some $n \geq 0$. Then

$$\begin{aligned} \Pi_{i \leq n+1} K_i &\simeq K_{n+1} \times \Pi_{i \leq n} K_i \\ &\simeq K_{n+1} \times (\mathbb{R}P^\infty \to K_n) \\ &\simeq K_{n+1} \times (\mathbb{R}P^\infty \to_{\text{pt}} K_{n+1}) \\ &\simeq \mathbb{R}P^\infty \to K_{n+1}. \end{aligned}$$

Here, the second line is by the inductive hypothesis, the third by Lemma 16, and the final line is by Lemma 17. It is direct to see that the forward composite is the desired one. □

Finally, we are ready to define the Steenrod squares.

Definition 18 (Steenrod squares)**.** We define the total square $\widehat{\mathsf{Sq}} : K_m \to \prod_{i \leq 2m} K_i$ by $\widehat{\mathsf{Sq}}(a) := \mathsf{Gys}_{2m}^{-1}(a^{(-)})$.We define the nth Steenrod square $\mathsf{Sq}^n : K_m \to K_{m+n}$ by

$$\mathsf{Sq}^n(a) := \begin{cases} \mathsf{proj}_{m+n}(\widehat{\mathsf{Sq}}(a)) & \text{if } n \leq m \\ 0 & \text{otherwise} \end{cases}$$

Unpacking the definition, we get the following characterisation of $\mathsf{Sq}^n(a)$ for a given $a : K_m$: they are the unique collection of terms such that for every $X : \mathbb{R}P^\infty$ we have[1]

$$a^X = \sum_{i=0}^{m} \mathsf{Sq}^m(a) \smile t(X)^{m-i}. \tag{3}$$

Note that (I2) holds by construction with this definition of Sq^n.

A. Proving the main theorem

Now that we have a definition of the Steenrod squares (following Brunerie), let us, in this section, work our way towards a proof of Theorem 1. The idea is to use Equation (3) to reduce properties of Sq^n to properties of $(-)^X$, and hence of the unordered cup product. The following is a simple example.

Lemma 19. *The Steenrod squares are pointed, i.e.* $\mathsf{Sq}^n(0) = 0$.

Proof. We have $0^X = 0$ for any $X : \mathbb{R}P^\infty$ since the unordered cup product is a bihom by construction. Thus Equation (3) gives $0 = \sum_{i=0}^{n} \mathsf{Sq}^n(a) \smile t(X)^{n-i}$ for all $X : \mathbb{R}P^\infty$. By Lemma 15, we must have $\mathsf{Sq}^n(a) = 0$ for all n. □

Perhaps more interestingly, the Cartan formula is equivalent to the following innocuous equation:

$$(a \smile b)^X = a^X \smile b^X. \tag{4}$$

We will prove the above equation via the following generalisation, which can be thought of as a type of Fubini interchange law and also will give rise to the Adem relations.

Theorem 20. *For any* $X, Y : \mathbb{R}P^\infty$, $n : X \times Y \to \mathbb{N}$ *and* $f : \prod_{x:X} \prod_{y:Y} K_{n(x,y)}$, *we have*

$$\smile_{x:X} \smile_{y:Y} f(x,y) = \smile_{y:Y} \smile_{x:X} f(x,y).$$

While easy to state, proving Theorem 20 is far more difficult than anything we have done so far. Its proof, which we assume for now but which will be discussed at length later in the paper, necessitates a development of the theory of unordered joins and constitutes the technical core of this paper. Before we are faced with reality, let us reap its fruits prematurely and prove the characterising properties of the Steenrod squares laid out in Theorem 1.

Proposition 21. *The Steenrod squares satisfy the Cartan formula (C).*

[1]Formally, we should include terms i from 0 to $2m$ in the equation. But the corresponding maps $\mathsf{Sq}^n : K_m \to K_{m+n}$ with $n < 0$ are zero for connectedness reasons; they are pointed by Lemma 19.

Proof. Consider Theorem 20 in these case where Y is $\mathbb{2}$, and n and f depend only their second arguments, so that they are given simply by $i, j : \mathbb{N}$ and $a : K_i$, $b : K_j$. In this case, the $\mathbb{2}$-indexed 'unordered' cup product reduces to the ordinary cup product, and the X-indexed cup product reduces to $(-)^X$, so that we end up with Equation (4), $(a \smile b)^X = a^X \smile b^X$. Combined with Equation (3), this gives the following:

$$\begin{aligned} &\sum_{k=0}^{i+j} \mathsf{Sq}^k(a \smile b) \smile t(X)^{i+j-k} \\ &= \left(\sum_{l=0}^{i} \mathsf{Sq}^l(a) \smile t(X)^{i-l} \right) \smile \left(\sum_{m=0}^{h} \mathsf{Sq}^m(b) \smile t(X)^{j-m} \right) \\ &= \sum_{l=0}^{i} \sum_{m=0}^{j} \mathsf{Sq}^l(a) \smile \mathsf{Sq}^m(b) \smile t(X)^{i+j-l-m}. \end{aligned}$$

Since, for given a, b, the above identity holds in $K_{2(i+j)}$ for *all* $X : \mathbb{R}P^\infty$, we may by Lemma 15 formally identify coefficients of the polynomials. This concludes the proof. □

Proposition 22. *The Steenrod squares satisfy (I3): for* $a : K_n$, *we have* $\mathsf{Sq}^n(a) = a \smile a$.

Proof. Taking X to be $\mathbb{2}$ in Equation (3), we have $t(X) = 0$ and so only one term in the sum remains: $a^{\mathbb{2}} = \mathsf{Sq}^n(a)$. We have $a^{\mathbb{2}} = a^2$ since the unordered cup product generalises the ordinary cup product. This concludes the proof. □

An important fact about Steenrod squares not listed in Theorem 1 is that they are additive: $\mathsf{Sq}^n(a+b) = \mathsf{Sq}^n(a) + \mathsf{Sq}^n(b)$. This is a consequence of (Ω), essentially because the action of any function on paths respects path composition. But there is also a more direct proof.

Lemma 23. *The Steenrod squares are additive: for* $a, b : K_m$ *we have* $\mathsf{Sq}^n(a+b) = \mathsf{Sq}^n(a) + \mathsf{Sq}^n(b)$.

Proof. By Equation (3), it suffices to prove that $(a+b)^X = a^X + b^X$. In fact, one can show a stronger statement, namely that $(-)^X : K_m \to K_{2m}$ has a unique delooping. Since this is a proposition, we may suppose that X is $\mathbb{2}$, i.e. it suffices to show that $(-)^2 : K_m \to K_{2m}$ has a unique delooping. By [15, Corollary 12], the delooping is unique if it exists, and it exists if and only if $(a+b)^2 = a^2 + b^2$. This holds by distributivity and commutativity since we are working mod 2. □

Remark 24. Related to Lemma 23, we remark that an alternative, simpler definition of Sq^n is possible. The stability axiom Theorem 1 tells us that the map $\mathsf{Sq}^n : K_m \to_{\mathsf{pt}} K_{m+n}$ should be a delooping of $\mathsf{Sq}^n : K_{m-1} \to_{\mathsf{pt}} K_{m+n-1}$. By [15, Corollary 12] and the fact that $(a+b)^2 = a^2 + b^2$, the delooping exists and is unique so we could take this as a recursive definition of Sq^n, starting from the definition of $\mathsf{Sq}^n : K_n \to K_{2n}$ as $x \mapsto x \smile x$. In this way, one would immediately get a stable cohomology operation, which is sufficient for some purposes, but this definition seems to give no insight toward proving the Cartan formula or Adem relations.

Before we continue with the remaining axioms, we will need the following lemma which will allow a computation of Sq^0 on K_1.

Lemma 25. *For $X : \mathbb{R}P^\infty$ we have $t(X)^X = 0$.*

Proof. We have to show that $\smile_{x:X} t(X) = 0$. Since the unordered cup product has the structure of a bihom, it suffices to prove that $t(X) = 0$ in K_1 for all $x : X$. This is direct; given $x : X$ we indeed have $X \simeq 2$ so that $t(X) = 0$. □

The above proof may seem curious: we argue that $t(X)^X = 0^X$, not by showing that $t(X) = 0$, but by showing $t(X) = 0$ for all $x : X$.

Lemma 26. *For $x : K_1$, we have $\mathsf{Sq}^0(x) = x$.*

Proof. We have $0 = t(X)^X = \mathsf{Sq}^0(t(X))t(X) + \mathsf{Sq}^1(t(X))$ by Equation (3). Here Sq^0 is a pointed map $K_1 \to_{\mathsf{pt}} K_1$ so it is given by multiplication by some element c of $\mathbb{Z}/2\mathbb{Z}$. We have that $\mathsf{Sq}^1(t(X)) = t(X)^2$ by Proposition 22. Thus $0 = c\,t(X)^2 + t(X)^2$. By formally identifying coefficients of polynomials, we get $c = 1$, so that Sq^0 is the identity map, as needed. □

In order to prove the stability axiom (Ω), it will be helpful to represent loop spaces in terms of maps from $\mathbb{S}^1$. In our setting, this is mediated by cup products and the pointed map $e : \mathbb{S}^1 \to_{\mathsf{pt}} K_1$ which sends the generating loop of $\mathbb{S}^1$ to the non-trivial loop of K_1, i.e. $\sigma_0(1)$, according to the following lemma.

Lemma 27. *With $e : \mathbb{S}^1 \to_{\mathsf{pt}} K_1$ as above and $n : \mathbb{N}$, the composite of $\sigma_n : K_n \to \Omega K_{n+1}$ with the canonical equivalence $\Omega K_{n+1} \to (\mathbb{S}^1 \to_{\mathsf{pt}} K_{n+1})$ is given by $a \mapsto (x \mapsto e(x) \smile a)$.*

Proof. Given $a : K_n$, it suffices to show that the action of $((-) \smile a) \circ e : \mathbb{S}^1 \to_{\mathsf{pt}} K_{n+1}$ on the generating loop, i.e. $\mathsf{loop} : \Omega\mathbb{S}^1$, is given by $\sigma_n(a) : \Omega K_{n+1}$. We have

$$\begin{aligned}\Omega(((-) \smile a) \circ e)(\mathsf{loop}) &= \Omega((-) \smile a)(\Omega(e)(\mathsf{loop})) \\ &= \Omega((-) \smile a)(\sigma_0(1)) \\ &= \sigma_n(1 \smile a) \\ &= \sigma_n(a)\end{aligned}$$

where, in the second-to-last step, we use the fact that the cup product respects looping. □

Proposition 28. *The Steenrod squares satisfy the stability axiom (Ω).*

Proof. Let Sq^n_m denote the Steenrod square $K_m \to K_{m+n}$, let $\sigma_m : K_k \xrightarrow{\sim}_{\mathsf{pt}} \Omega K_{m+1}$ and let τ_m denote the canonical equivalence $\Omega K_m \xrightarrow{\sim} (\mathbb{S}^1 \to_{\mathsf{pt}} K_m)$. Given $m, n : \mathbb{N}$, we wish to show that square (A) in the following diagram commutes.

$$\begin{array}{ccc} K_m & \xrightarrow{\mathsf{Sq}^n_m} & K_{m+n} \\ {\scriptstyle\sigma_m}\downarrow & (\mathrm{A}) & \downarrow{\scriptstyle\sigma_{m+n}} \\ \Omega K_{m+1} & \xrightarrow{\Omega\mathsf{Sq}^n_{m+1}} & \Omega K_{m+n+1} \\ {\scriptstyle\tau_{m+1}}\downarrow & (\mathrm{B}) & \downarrow{\scriptstyle\tau_{m+n+1}} \\ (\mathbb{S}^1 \to_{\mathsf{pt}} K_{m+1}) & \xrightarrow[\mathsf{Sq}^n_{m+1} \circ (-)]{} & (\mathbb{S}^1 \to_{\mathsf{pt}} K_{m+n+1}) \end{array}$$

To this end, we note that it is easy to see that square (B) commutes and that all vertical maps are equivalences. Hence, it suffices to show that the outer square commutes. By Lemma 27, the vertical composites are given by $a \mapsto (x \mapsto e(x) \smile a)$. Thus it suffices to show that for every $a : K_m$ and $x : \mathbb{S}^1$, we have

$$\mathsf{Sq}^n(e(x) \smile a) = e(x) \smile \mathsf{Sq}^n(a).$$

By the Cartan formula, the left hand side computes to $\mathsf{Sq}^0(e(x)) \smile \mathsf{Sq}^n(a) + \mathsf{Sq}^1(e(x)) \smile \mathsf{Sq}^{n-1}(a)$. By Lemma 26 we have $\mathsf{Sq}^0(e(x)) = e(x)$ so it suffices to show that $\mathsf{Sq}^1(e(x)) = 0$, i.e. that $e(x) \smile e(x) = 0$.

To see why this holds, one can simply note that the map $x \mapsto e(x) \smile e(x)$ factors as

$$\mathbb{S}^1 \xrightarrow{\Delta} \mathbb{S}^1 \wedge \mathbb{S}^1 \xrightarrow{\smile} K_1 \wedge K_1.$$

The diagonal map $\Delta : A \to A \wedge A$ vanishes whenever A is a suspension (so, in particular when $A = \mathbb{S}^1$). This follows by straightforward suspension induction. □

Proposition 29. *The Steenrod squares satisfy axiom (I1), i.e. $\mathsf{Sq}^0 = \mathsf{id}$.*

Proof. The zeroth square lives in the type of pointed functions $K_n \to_{\mathsf{pt}} K_n$ – a type which we understand well: looping $(K_n \to_{\mathsf{pt}} K_n) \to (K_{n-1} \to_{\mathsf{pt}} K_{n-1})$ is an equivalence for each $n \geq 0$. Since looping preserves the identity function, it is thus, by (Ω), enough to show that $\mathsf{Sq}^0_0 : K_0 \to K_0$ is the identity. By (I3), we have $\mathsf{Sq}^0_0(x) = x \smile x$. However, since $K_0 := \mathbb{Z}/2\mathbb{Z}$, the cup product here is simply multiplication in $\mathbb{Z}/2\mathbb{Z}$ and thus $\mathsf{Sq}^0_0(x) = x \smile x = x$. □

Proposition 30. *The Steenrod squares satisfy the Adem relations (A).*

Proof. Given $m : \mathbb{N}$ and $a : K_m$, consider Theorem 20 in the case where n and f are constantly m and a. In this case, we have for any $X, Y : \mathbb{R}P^\infty$ that

$$(a^X)^Y = (a^Y)^X.$$

The idea is now simply to expand each side using Equation (3). We also make use of Lemma 23, the Cartan formula, and the

fact that $t(X)^Y = t(X)^2 + t(X) \smile t(Y)$, which follows from (I3). In this way, we get the following.

$$\begin{aligned}(a^X)^Y &= \left(\sum_i \mathsf{Sq}^i(a) \smile t(X)^{n-i}\right)^Y \\ &= \sum_i \mathsf{Sq}^i(a)^Y \smile (t(X)^2 + t(X)t(Y))^{n-i} \\ &= \sum_{i,j,k} \binom{n-i}{k} \mathsf{Sq}^j\mathsf{Sq}^i(a) t(Y)^{2n-j-k} t(X)^{n+k-i}\end{aligned}$$

In the same way, one can express $(a^Y)^X$ as a polynomial in $t(X)$ and $t(Y)$. Since we have $(a^X)^Y = (a^Y)^X$, we can formally identify coefficients in these polynomials, by repeated application of Lemma 15. The steps required to go from here to the Adem relations are the same as in the classical case; see [16, Page 345] for details. □

This concludes the proof of Theorem 1. A natural question to ask after seeing the short and snappy proofs in this section is whether, perhaps, the setting of HoTT makes working with cohomology operations like the Steenrod squares 'easier' than in a more traditional setting. Although there are aspects of our setup here which certainly seem to speak in favour of HoTT, we wish to use this section to emphasise that *a lot* of the heavy lifting is done by Theorem 20 which we have, thus far, only assumed. Thus, this question entirely hinges on the difficulty of proving this statement. For this reason, we will now devote the remainder of the paper to proofs. They are interesting in their own right, as they force us to develop a fair bit of novel machinery surrounding unordered joins.

IV. Unordered joins and their Fubini theorem

We now set out to prove Theorem 20. This theorem states that our unordered cup product satisfies a certain 'Fubini theorem'. Recall from the definition of the unordered cup product $\smile_X \colon \Pi_{x:X} K_{n(x)} \to K_{\Sigma n}$ that it has the structure of a bihom, i.e. it is pointed in each argument, and that it is non-trivial. This characterises $\smile_X$ up to contractible choice, and so anything we prove about $\smile_X$ should come from this characterisation. In order to prove Theorem 20, we would like to have a similar characterisation of the iterated cup product

$$\smile_X \smile_Y \colon \Pi_{x:X} \Pi_{y:Y} K_{n(x,y)} \to K_{\Sigma n}.$$

In other words, we would like to find a way to uniquely characterise $\smile_X \smile_Y$, that is symmetric in X and Y. Morally, this characterisation is that $\smile_X \smile_Y$ is coherently pointed in each of its $X \times Y$-many arguments. Thus we are lead to consider a generalisation of $\mathsf{isBiHom}$ to the 4-element type $X \times Y$. Already defining such a generalisation is rather complicated; it involves the combinatorics of all non-empty (decidable) subsets of $X \times Y$ and their inclusions. So it will be helpful to have another perspective on $\mathsf{isBiHom}$. In fact, Brunerie [9] never considered $\mathsf{isBiHom}$, and instead worked with its corepresenting object, the unordered smash product:

Definition 31. Let $X : \mathbb{R}P^\infty$ and $A : X \to \mathcal{U}_{\mathsf{pt}}$ be a family of pointed types. We define the unordered smash product of A, denoted $\bigwedge_{x:X} A(x)$, by the following pushout

$$\begin{array}{ccc}(x : X) \times A(x) & \xrightarrow{(x,a) \mapsto \mathsf{Elim}^{x \mapsto a}_{-x \mapsto \mathsf{pt}}} & \prod_{x:X} A(x) \\ {\scriptstyle \mathsf{fst}}\downarrow & \ulcorner & \downarrow \\ X & \longrightarrow & \bigwedge_{x:X} A(x)\end{array}$$

We take this type to be pointed by $\mathsf{inr}(\lambda x\,.\mathsf{pt})$.

It is easy to see that $\mathsf{isBiHom}(f)$ is equivalent to asking that f factors through $\mathsf{inr} : \Pi_{x:X} A(x) \to \bigwedge_{x:X} A(x)$ via a pointed map $\bigwedge_{x:X} A(x) \to_{\mathsf{pt}} B$, and that $\mathsf{BiHom}_X(A, B)$ is equivalent to the type of all such pointed maps $\bigwedge_{x:X} A(x) \to_{\mathsf{pt}} B$. In this way, one can see that $\smile_X \smile_Y$ is given by the composite of the map $\Pi_{x:X} \Pi_y K_{n(x,y)} \to \bigwedge_{x:X} \bigwedge_{y:Y} K_{n(x,y)}$, given by $f \mapsto \mathsf{inr}(x \mapsto \mathsf{inr}(y \mapsto f(x,y)))$ with the unique non-trivial map $\bigwedge_{x:X} \bigwedge_{y:Y} K_{n(x,y)} \to_{\mathsf{pt}} K_{\Sigma n}$. What remains to be shown is that we have a pointed equivalence $e : \bigwedge_{x:X} \bigwedge_{y:Y} A(x,y) \simeq \bigwedge_{y:Y} \bigwedge_{x:X} A(x,y)$ such that the following diagram commutes.

$$\begin{array}{ccc}\Pi_{x:X} \Pi_{y:Y} A(x,y) & \xrightarrow{\mathsf{swap}} & \Pi_{y:Y} \Pi_{x:X} A(x,y) \\ \downarrow & & \downarrow \\ \bigwedge_{x:X} \bigwedge_{y:Y} A(x,y) & \xrightarrow[e]{} & \bigwedge_{y:Y} \bigwedge_{x:X} A(x,y)\end{array}$$

Again, this turns out to be rather complicated, and we need another simplifying device.

Definition 32. Let $X : \mathbb{R}P^\infty$ and $A : X \to \mathcal{U}$. We define the unordered join of A, denoted $\bigast_{x:X} A(x)$, by the following pushout.

$$\begin{array}{ccc}X \times \Pi_{x:X} A(x) & \xrightarrow{\mathsf{snd}} & \Pi_{x:X} A(x) \\ {\scriptstyle (x,f) \mapsto (x, f(x))}\downarrow & \ulcorner & \downarrow \\ (x : X) \times A(x) & \longrightarrow & \bigast_{x:X} A(x)\end{array}$$

The following lemma says that the unordered join agrees with the usual definition of joins when X is $\mathbb{2}$. Recall that the usual definition is $A_0 * A_1 \coloneqq A_0 \sqcup^{A_0 \times A_1} A_1$.

Lemma 33. *Given two types A_0 and A_1, we have*

$$\bigast_{x:\mathbb{2}} A_x \simeq A_0 * A_1.$$

Proof. By the 3×3 lemma applied to the following diagram.

$$\begin{array}{ccccc}A_0 & \longleftarrow & \varnothing & \longrightarrow & A_1 \\ \uparrow{\scriptstyle \mathsf{fst}} & & \uparrow & & \uparrow{\scriptstyle \mathsf{snd}} \\ A_0 \times A_1 & \longleftarrow & \varnothing & \longrightarrow & A_0 \times A_1 \\ \downarrow{\scriptstyle \mathsf{id}} & & \downarrow & & \downarrow{\scriptstyle \mathsf{id}} \\ A_0 \times A_1 & \xleftarrow{\mathsf{id}} & A_0 \times A_1 & \xrightarrow{\mathsf{id}} & A_0 \times A_1\end{array}$$

□

The unordered join is relevant for us because $\mathsf{isBiHom}(f)$ is easily seen to be equivalent to

$$(a : \Pi_{x:X} A(x)) \to \underset{x:X}{\ast}(a(x) = \mathrm{pt}) \to (f(a) = \mathrm{pt}).$$

An equivalent way to phrase this (which we will not use) is that $\bigwedge_{x:X} A(x)$ is the cofibre of the projection

$$(a : \Pi_{x:X} A(x)) \times \underset{x:X}{\ast}(a(x) = \mathrm{pt}) \to \Pi_{x:X} A(x)$$

onto the first component; this map might be called an *unordered wedge inclusion*.

Given this characterisation of $\mathsf{isBiHom}(f)$ in terms of the unordered join, the proof of Theorem 20 will eventually reduce to the following lemma.

Lemma 34. *For any $X, Y : \mathbb{R}P^\infty$ and $A : X \times Y \to \mathcal{U}$, we have a function*

$$\underset{x:X}{\ast}\,\underset{y:Y}{\ast} A(x,y) \to \underset{y:Y}{\ast}\,\underset{x:X}{\ast} A(x,y)$$

Interestingly, we do not need to ask for any properties of this function, although it is important that we *construct* a function as opposed to proving its mere existence. Since the proof of Lemma 34 is rather technical, we postpone it for the moment and turn to the main result promised at the beginning of this section.

Proof of Theorem 20. Suppose we are given $X, Y : \mathbb{R}P^\infty$, $n : X \times Y \to \mathbb{N}$, fixed throughout the proof; we would like to show that $\smile_X \smile_Y = \smile_Y \smile_X$. Given $f : \Pi_{x:X}\Pi_{y:Y} K_{n(x,y)}$, we claim that we have a map

$$\underset{x:X}{\ast}\,\underset{y:Y}{\ast}(f(x,y) = 0) \to (\smile_{x:X} \smile_{y:Y} f(x,y) = 0). \tag{5}$$

Indeed, we have a map $\ast_{x:X} \ast_{y:Y} (f(x,y) = 0) \to \ast_{x:X} (\smile_{y:Y} f(x,y) = 0)$ by functoriality of the unordered join together with the fact that $\smile_Y$ is a bihom, and a map $(\ast_{x:X} (\smile_{y:Y} f(x,y) = 0)) \to (\smile_{x:X} \smile_{y:Y} f(x,y) = 0)$ since $\smile_X$ is a bihom.

Now let T be the sigma-type consisting of functions $\mu : \Pi_{x:X}\Pi_{y:Y} K_{n(x,y)} \to K_{\Sigma n}$ together with a proof that for all $f : \Pi_{x:X}\Pi_{y:Y} K_{n(x,y)}$, we have $\ast_{x:X} \ast_{y:Y} f(x,y) = \mathrm{pt} \to \mu(f) = \mathrm{pt}$. We can construct two elements of T: on the one hand we have $\smile_X \smile_Y$ together with the argument above that we have a map as in Equation (5). By symmetry and using Lemma 34, we can similarly construct an element of T whose first component is $\smile_Y \smile_X$. Now we claim that these two elements of T are equal; in fact, we claim that there is a unique identification between them. This will finish the proof, since if two pairs are equal then so are their first components.

Since our claim is now a proposition, we may assume that X and Y are both $\mathbb{2}$. In this case, we have that T is the set of pointed maps

$$K_{n(0,0)} \wedge K_{n(0,1)} \wedge K_{n(1,0)} \wedge K_{n(1,1)} \to_{\mathrm{pt}} K_{\Sigma n}.$$

By a version of 'Cavallo's trick' [17, Lemma 15], two pointed maps out of a smash product into a homogeneous type are equal if they are equal when restricted to the product, in this case $\Pi_{i,j\in\{0,1\}} K_{n(i,j)}$. Our two elements of T correspond to the maps

$$(a_{00}, a_{01}, a_{10}, a_{11}) \mapsto (a_{00} \smile a_{01}) \smile (a_{10} \smile a_{11})$$

and

$$(a_{00}, a_{01}, a_{10}, a_{11}) \mapsto (a_{00} \smile a_{10}) \smile (a_{01} \smile a_{11}).$$

Indeed these are equal by commutativity and associativity of the cup product. This concludes the proof. □

A. Proving Lemma 34

No result used in this project has turned out to be quite as problematic as Lemma 34 – the result is highly technical and its computer formalisation was only completed after a year's worth of failed attempts. While we would be very happy to see a more conceptual proof of this statement, we are sceptical to the existence of such a proof. To illustrate why, note that the pushout describing the unordered join $\ast_{x:X} A(x)$ in no way requires X to be a two-element type – the definition makes sense for arbitrary X, although in this case we might write it with an apostrophe $\ast'$ to remind ourselves that it is no longer a good generalisation of the usual join. Thus, we may ask whether it is true, for *any* two types X and Y and family $A : X \times Y \to \mathcal{U}$ whether the map

$$\mathcal{F} : \underset{x:X}{\ast'}\,\underset{y:Y}{\ast'} A(x,y) \to \underset{y:Y}{\ast'}\,\underset{x:X}{\ast'} A(x,y)$$

exists. This seems difficult (if not impossible) to do in general, as we cannot prove in general that the domain and codomain are equivalent. For instance, if we set $X = \mathbb{2}$ and $Y = \mathsf{hProp}$, $A(0,P) :\equiv \neg P$ and $A(1,P) :\equiv P$, then the LHS becomes contractible whereas the RHS is equivalent to the suspension of LEM – a type whose contractibility is independent of HoTT.

So, let us try to define $\mathcal{F}$ for $X, Y : \mathbb{R}P^\infty$. The function will be described by the following data:

$$\mathcal{F}_l : \left(\prod_{x:X} \underset{y:Y}{\ast} A(x,y)\right) \to \underset{y:Y}{\ast}\,\underset{x:X}{\ast} A(x,y)$$
$$\mathcal{F}_r : (x : X) \times \underset{y:Y}{\ast} A(x,y) \to \underset{y:Y}{\ast}\,\underset{x:X}{\ast} A(x,y)$$
$$\mathcal{F}_{lr} : (x : X)\left(f : \prod_{x:X} \underset{y:Y}{\ast} A(x,y)\right) \to \mathcal{F}_l(f) = \mathcal{F}_r(x, f(x))$$

The key problem here is defining $\mathcal{F}_l$ – its codomain is a Π-type and thus does not automatically come equipped with an elimination rule. Consequently, we need to understand the type $\prod_{x:X} \ast_{y:Y} A(x,y)$. Luckily, it turns out that we can describe this Π-type using a rather involved construction. To give it a (somewhat) more concise definition, let us define, for any two types B and C, and family $R : B \to C \to \mathcal{U}$, the *relational* pushout, $P(B,C,R)$ to simply be the pushout of the span $B \leftarrow (b : B) \times (c : C) \times R(b,c) \to C$. Any pushout $B \xrightarrow{f} D \xleftarrow{g} C$ can be written as a relational pushout by setting $R(b,c) :\equiv (d : D) \times (f(d) = b) \times (g(d) = c)$ (and vice versa).

Lemma 35. *Let $X : \mathbb{R}P^\infty$, $B, C : X \to \mathcal{U}$, and $R : B(x) \times C(x) \to \mathcal{U}$ (with $x : X$ an implicit argument). Then $\prod_{x:X} P(B(x), C(x), R(x))$ is equivalent to the HIT T with the following constructors.*

$$
\begin{aligned}
bb &: (\textstyle\prod_{x:X} B(x)) \to T \\
cc &: (\textstyle\prod_{x:X} C(x)) \to T \\
bc &: (x : X) \times B(x) \times C(\neg x) \to T \\
rr &: (b : \textstyle\prod_{x:X} B(x))(c : \prod_{x:X} C(x))(r : \prod_{x:X} R(b(x), c(x))) \to \\
&\quad bb(b) = cc(c) \\
br &: (x : X)(b : \textstyle\prod_{k:X} B(k))(c : C(\neg x))(r : R(b(\neg x), c)) \to \\
&\quad bb(a) = bc(b(x), c) \\
cr &: (x : X)(b : B(x))(c : \textstyle\prod_{k:X} C(k))(r : R(b, c(x))) \to \\
&\quad bc(b, c(\neg x)) = cc(c) \\
rr' &: (b : \textstyle\prod_{x:X} B(x))\,(c : \prod_{x:X} C(x)) \to \\
&\quad (r : \textstyle\prod_{x:X} R(b(x), c(x)))\,(x : X) \to \\
&\quad rr(b, c, r) = br(x, \neg x, b, c(\neg x), r(\neg x)) \cdot cr(x, \neg x, b(x), c, r(x))
\end{aligned}
$$

Proof sketch. It is straightforward to define a map $w_X : T \to \Pi_{x:X} P(B(x), C(x), R(x))$ by T-induction. By Lemma 2(a), it is sufficient to show that w_X is an equivalence when $X = 2$. This is somewhat technical but can be done with relative ease. □

If we instantiate the above with $B(x) \coloneqq (y : Y) \times A(x, y)$, $C(x) \coloneqq \prod_{y:Y} A(x, y)$ and $R((y, a), f) \coloneqq (f(y) = a)$, we have $P(B(x), C(x), R(x)) \simeq \ast_{y:Y} A(x, y)$ and thus Lemma 35 tells us that there is an equivalence $w : T \simeq \prod_{x:X} \ast_{y:Y} A(x, y)$. Hence, in order to construct $\mathcal{F}_l$, it is enough to define a map $T \to \ast_{y:Y} \ast_{x:X} A(x, y)$.

Mapping out of T is, in general, not much easier than mapping out of $\prod_{x:X} \ast_{y:Y} A(x, y)$: a map out of T must be defined over $\Pi_{x:X} B(x)$ and $\Pi_{x:X} C(x)$ which again forces us to map out of Π-types. The type $\Pi_{x:X} C(x)$ is unproblematic: it is simply $\Pi_{x:X} \Pi_{y:Y} A(x, y)$ and defines an element of $\ast_{y:Y} \ast_{x:X} A(x, y)$ by simply swapping the arguments. However, $\prod_{x:X} B(x) \coloneqq \Pi_{x:X}((y : Y) \times A(x, y))$ is more complicated – where we send an element f of this type depends on the behaviour of $\mathsf{fst} \circ f : X \to Y$. Fortunately, we can understand this type.

Lemma 36. *For any $X, Y : \mathbb{R}P^\infty$, the map $((X \simeq Y) + Y) \to (X \to Y)$ sending equivalences to their underlying functions and $y : Y$ to the constant map $\lambda\, x\,.\, y$ is an equivalence.*

Proof. Since the statement is a proposition, it suffices to show it when $X = Y = 2$. In this case, the statement is simply the trivial observation that any function $2 \to 2$ is either an equivalence or constant. □

Using Lemma 36, we can replace each occurrence of $\Pi_{x:X}((y : Y) \times A(x, y))$ in T with the equivalent type

$$((e : X \simeq Y) \times A(x, e(x))) + ((y : Y) \times A(x, y))$$

which has a more well-behaved elimination principle. After this rewriting, it is possible to define, by pattern matching, a map $\psi_{X,Y} : T \to \ast_{y:Y} \ast_{x:X} A(x, y)$, which allows us to define $\mathcal{F}_l = \psi_{X,Y} \circ \phi$, where ϕ is the appropriate instance of Lemma 35. We then need to define, for each $x : X$, the function $\mathcal{F}_r(x, -)$ and the homotopy $\mathcal{F}_{lr}(x, -)$. In theory, this is somewhat easier: as we have $x : X$ in context, we may apply Lemma 6. In practice, we have to deal with a large number of difficult coherence problems which we completely sweep under the rug here.

V. Joins and E_∞-monoids

Much progress in synthetic homotopy type theory is held back by what is known as the *problem of infinite objects* [18]. This problem appears in many guises, and is traditionally explained in terms of semi-simplicial types. In this work we brush against instances of this problem in many places where we would like to talk about higher commutative, more precisely E_∞-, monoids.

The idea of E_∞-monoids[2] is remarkably natural from a type-theoretic perspective. An E_∞-monoid should consist of a type A and for *any finite type* X a 'multiplication' map

$$\mu_X : A^X \to A.$$

These multiplication maps express that any finite collection of elements of A can be multiplied, and that their order is irrelevant in a homotopy coherent sense. The unary multiplication $\mu_1 : A^1 \to A$ should be the canonical equivalence. These multiplication maps are subject to certain coherences, starting with the following: given a finite type $X : \mathcal{U}$ and a family of finite types $Y : \mathcal{U}$ indexed by X, the two maps $A^{(x:X)\times Y(x)} \to A$ given respectively by $\mu_{(x:X)\times Y(x)}$ and

$$a \mapsto \mu_X(x \mapsto \mu_Y(y \mapsto a(x, y)))$$

are identified. This expresses a kind of generalised associativity.

This is not a complete definition of E_∞-monoids – it is missing an infinite tower of higher coherences – but we can get far with just the data above.

For example, we would like to know that the universe of types $\mathcal{U}$ forms an E_∞-monoid where the multiplication is given by unordered join of types. The unordered binary join would be a special case, and the Fubini map (which really should be an equivalence) of Lemma 34 – our main technical result – would be a consequence of generalised associativity, since $\mu_{X\times Y}$ and $\mu_{Y\times X}$ are related by the path $X \times Y = Y \times X$ obtained from univalence. In this way, the technical burden of this paper would be much lighter if we simply had access to this E_∞-monoid structure!

The problems associated with this appear at three different levels. At the first level, we cannot even define the whole infinite tower of E_∞-coherences in type theory. This is a well-known instance of the problem of infinite objects, but is not so relevant for us. At the second level, it is not clear how to define the unordered join of a general finite family of types. We have seen how to do it given a finite family of size 2, and it is clear how to do it for 3 and 4, but the complexity grows very quickly, and we run into the problem of infinite objects in trying to define a general pattern. At the third level, consider

[2] Usually referred to by the less descriptive term E_∞-*spaces*.

what happens when we try to reason about unordered joins of just a few spaces – for example as in Lemma 34. At this level, our problems are finitary and in principle surmountable. But they are also remarkably difficult since we lack a systematic way to approach these problems. This story can be compared with that of coherences for the smash product [17].

VI. Steenrod squares and $\pi_4(\mathbb{S}^3)$

The Steenrod squares are not only an esoteric construction: they have several crucial applications in algebraic topology. One typical example is the computation of $\pi_4(\mathbb{S}^3)$, the 4th homotopy group of the 3-sphere, i.e. $\|\mathbb{S}^4 \to_{\text{pt}} \mathbb{S}^3\|_0$. Although the fact that $\pi_4(\mathbb{S}^3) \cong \mathbb{Z}/2\mathbb{Z}$ is well known in HoTT, due to Brunerie [7], it was shown by Ljungström and Mörtberg [19] that this can be shown in a very direct way under the assumption that $\pi_4(\mathbb{S}^3)$ is non-trivial. We can now complete this proof by giving a new (in HoTT) argument for why $\pi_4(\mathbb{S}^3)$ does not vanish. Let $h : \mathbb{S}^3 \to \mathbb{S}^2$ be the *Hopf map*, i.e. the generator of $\pi_3(\mathbb{S}^2)$. If we can show that its suspension $\Sigma h : \mathbb{S}^4 \to \mathbb{S}^3$ is non-trivial, we are done. In HoTT, we define $\mathbb{C}P^2 :\equiv C_h$ to be the cofibre of h. Since suspensions commute with cofibres, we get $C_{\Sigma h} \simeq \Sigma\mathbb{C}P^2$. On the other hand, the cofibre of the constant pointed map $\mathsf{const} : \mathbb{S}^4 \to \mathbb{S}^3$ is equivalent to $\mathbb{S}^5 \vee \mathbb{S}^3$, i.e. the pushout of the span $\mathbb{S}^5 \leftarrow \mathbb{1} \to \mathbb{S}^3$. Thus, we are done if we can show the following.

Theorem 37. $\Sigma\mathbb{C}P^2 \not\simeq \mathbb{S}^5 \vee \mathbb{S}^3$

Proof. Both spaces in questions are suspensions, with $\mathbb{S}^5 \vee \mathbb{S}^3 \simeq \Sigma(\mathbb{S}^4 \vee \mathbb{S}^2)$. We consider the following diagram where $A \in \{\mathbb{C}P^2, \mathbb{S}^4 \vee \mathbb{S}^2\}$ (and where, we remark, all cohomology groups are equivalent to $\mathbb{Z}/2\mathbb{Z}$).

$$\begin{array}{ccc} H^2(A,\mathbb{Z}/2\mathbb{Z}) & \xrightarrow{(-)^2} & H^4(A,\mathbb{Z}/2\mathbb{Z}) \\ \sim\downarrow & & \downarrow \\ H^3(\Sigma A,\mathbb{Z}/2\mathbb{Z}) & \xrightarrow[\mathsf{Sq}^2]{} & H^5(A,\mathbb{Z}/2\mathbb{Z}) \end{array}$$

This diagram is an instance of the suspension property for the Steenrod squares. When $A = \mathbb{C}P^2$, the squaring map is non-trivial (this was proved in HoTT by Brunerie [7] using $\mathbb{Z}$-coefficients but the proof works fine also for $\mathbb{Z}/2\mathbb{Z}$-coefficients). When $A = \mathbb{S}^4 \vee \mathbb{S}^2$, the squaring map is trivial since wedge sums have trivial cup products. So $\Sigma\mathbb{C}P^2$ has non-trivial Sq^2 whereas for $\mathbb{S}^5 \vee \mathbb{S}^3$ it is trivial, and thus we may conclude that these types cannot be equivalent. □

Corollary 38. $\pi_4(\mathbb{S}^3) \neq 0$

VII. Conclusions and Future Work

In this paper, we have proved important properties of the Steenrod squares in homotopy type theory, following a definition of Brunerie. This has necessitated the development of a substantial theory of unordered HITs, and we hope that our paper can serve as an illustration of the current state of synthetic homotopy theory – its power and its limitation.

This work suggests a number of further questions. Regarding Steenrod squares, one would like to know that they generate all maps $K(\mathbb{Z}/2\mathbb{Z}, n) \to K(\mathbb{Z}/2\mathbb{Z}, m)$ in an appropriate sense. Can this be proved in homotopy type theory? What about the assertion that the equations listed in Theorem 1 generate all possible relations between the Steenrod squares?

More generally, for any odd prime p there should be an analogue of the Steenrod square, namely the Steenrod reduced pth power $P^n : K(\mathbb{Z}/p\mathbb{Z}, m) \to_{\text{pt}} K(\mathbb{Z}/p\mathbb{Z}, m + 2n(p-1))$. Can this even be studied in homotopy type theory? It is not clear how to even define it without running into the problem of infinite objects.

Let us also highlight the problems discussed in Section V. Is it possible to define the unordered join of a general finite family of types? And is it possible to prove things like Lemma 34 more efficiently and systematically?

References

[1] M. Shulman. (2019, April) All $(\infty, 1)$-toposes have strict univalent universes. Preprint. [Online]. Available: https://arxiv.org/abs/1904.07004

[2] 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

[3] G. Brunerie, A. Ljungström, and A. Mörtberg, "Synthetic Integral Cohomology in Cubical Agda," in *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), F. Manea and A. Simpson, Eds., vol. 216. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, pp. 11:1–11:19. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2022/15731

[4] T. Lamiaux, A. Ljungström, and A. Mörtberg, "Computing cohomology rings in cubical agda," in *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs*, ser. CPP 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 239–252. [Online]. Available: https://doi.org/10.1145/3573105.3575677

[5] A. Ljungström and A. Mörtberg, "Computational synthetic cohomology theory in homotopy type theory," 2024. [Online]. Available: https://arxiv.org/abs/2401.16336

[6] 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

[7] 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

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

[9] G. Brunerie, "The steenrod squares in homotopy type theory," 2016, abstract at *23rd International Conference on Types for Proofs and Programs (TYPES 2017)*. [Online]. Available: https://types2017.elte.hu/proc.pdf#page=45

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

[11] U. Buchholtz, "Unordered pairs in homotopy type theory," 2023, preprint. [Online]. Available: https://ulrikbuchholtz.dk/pairs.pdf

[12] U. Buchholtz and E. Rijke, "The real projective spaces in homotopy type theory," in *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2017, pp. 1–8.

[13] N. Kraus, "Truncation levels in homotopy type theory," July 2015. [Online]. Available: https://eprints.nottingham.ac.uk/28986/
[14] U. Buchholtz, F. van Doorn, and E. Rijke, "Higher Groups 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. 205–214. [Online]. Available: https://doi.org/10.1145/3209108.3209150
[15] D. Wärn, "Eilenberg–maclane spaces and stabilisation in homotopy type theory," *Journal of Homotopy and Related Structures*, vol. 18, no. 2, pp. 357–368, Sep 2023.
[16] J.-C. Hausmann, *Mod two homology and cohomology*. Springer, 2014, vol. 10.
[17] A. Ljungström, "Symmetric monoidal smash products in homotopy type theory," *Mathematical Structures in Computer Science*, p. 1–23, 2024.
[18] U. Buchholtz, *Higher Structures in Homotopy Type Theory*, 2019, pp. 151–172.
[19] A. Ljungström and A. Mörtberg, "Formalising and computing the fourth homotopy group of the 3-sphere in cubical agda," 2024. [Online]. Available: https://arxiv.org/abs/2302.00151