Thesis for The Degree of Licentiate of Philosophy

Some results in synthetic homotopy theory

David Wärn

Department of Computer Science and Engineering
University of Gothenburg | Chalmers University of Technology
Gothenburg, Sweden, 2025

Some results in synthetic homotopy theory

David Wärn

ISSN 1652-876X

Department of Computer Science and Engineering
Division of Computing science
University of Gothenburg | Chalmers University of Technology
SE-412 96 Göteborg,
Sweden
Phone: +46(0)31 772 1000

Printed by Chalmers Digitaltryck,
Gothenburg, Sweden 2025.

Some results in synthetic homotopy theory

David Wärn

Department of Computer Science and Engineering
University of Gothenburg | Chalmers University of Technology

Abstract

Synthetic homotopy theory is a relatively new approach to homotopy theory based on a variant of Martin-Löf type theory called homotopy type theory. A central theme of synthetic homotopy theory is the study of identity types, or path spaces, which describe the ways in which two elements of the same type can be equal. This thesis consists of two works in the field of synthetic homotopy theory which shed some light on the nature of identity types.

The first and most significant consists of a precise and useful description of identity types of pushouts by what we call *the zigzag construction*. This can equivalently be seen as a description of the free ∞-groupoid on a graph of spaces. The zigzag construction notably has no exact classical counterpart and should be of interest also outside of type theory. In type theory, it provides a solution to a long-standing open problem in synthetic homotopy theory, namely the problem of showing that the suspension of a 0-type is 1-truncated.

The second consists of an elementary treatment of stabilisation in spaces. This is the phenomenon that pointed types admit unique deloopings once they are sufficiently connected and truncated, and that pointed maps admit unique deloopings under similar conditions. This allows us to construct types with specified identity types. In particular we present a new description of Eilenberg–MacLane spaces.

Keywords

Type theory, homotopy theory, homotopy type theory, synthetic homotopy theory

List of Publications

This thesis is based on the following publications:

[**Paper I**] **D. Wärn**, *Path spaces of pushouts*
To be submitted.

[**Paper II**] **D. Wärn**, *Eilenberg–MacLane spaces and stabilisation in homotopy type theory*
Journal of Homotopy and Related Structures (September 2023), Volume 18, pages 357–368.

Acknowledgment

I would like to thank my colleagues at the Logic & Types unit, past and present, for years' worth of helpful discussions. In particular, the first paper in this thesis owes a great deal to discussions with my co-supervisor Christian Sattler, and the second paper owes a great deal to my supervisor Thierry Coquand.

Contents

Part I

Summary

Chapter 1

Introduction

Around 2006, a surprising link was established between two seemingly unrelated subjects: homotopy theory and type theory. Type theory as we know it today was introduced by Swedish logician and philosopher Per Martin-Löf in order to provide a foundation for constructive mathematics, a foundation which simultaneously functions as a programming language suitable for use in computer proof assistants. Homotopy theory on the other hand is a rapidly evolving branch of mathematics with roots in topology, based on the idea of replacing the notion of sets with that of *spaces*, objects also known as ∞-*groupoids* or *anima*. The connection between these two fields was anticipated by work of Hoffman and Streicher [9] and Awodey and Warren [2], and made precise in foundational work of Voevodsky [26, 12], resulting in a flavour of type theory known as homotopy type theory (HoTT). HoTT attracted significant attention in connection with a *Special Year* held at the Institute for Advanced Study in 2012–2013. The IAS Special year was a collaborative endeavour resulting in a book explaining the fundamental ideas of HoTT, usually referred to as *the HoTT book* [25].

The basic objects of type theory are types, and types in HoTT are to be thought of as spaces. By combining perspectives from type theory, constructive mathematics, and classical homotopy theory, HoTT is a source of new ideas. Of particular relevance for us is the line of research dubbed synthetic homotopy theory. The type-theoretic approach to homotopy theory is called synthetic, as opposed to analytic, because it treats spaces (types) as primitive objects, without ever asking what a space *is*. The answer would in any case be complicated and, arguably, unenlightening. By abstracting away the details of how spaces are represented – technicalities like Quillen model categories and the combinatorics of simplicial objects – the promise of synthetic homotopy theory is to get to the conceptual heart of homotopy theory. At the same time, synthetic homotopy theory offers semantic generality. Just as an argument in group theory can be instantiated in any group, i.e. in any model of the first-order theory of groups, an argument in synthetic homotopy theory can be instantiated in any model of HoTT. A model of HoTT can be thought of as providing an exotic notion of 'space'. A particular class of models of interest

are given by ∞-topoi [19, 24].

Most work in synthetic homotopy theory can however be characterised as reproducing classical results in a synthetic setting. Examples include the proof that the loop space of the circle is $\mathbb{Z}$ [17], the long exact sequence on homotopy groups associated with a fibre sequence [25], the Seifert–Van Kampen theorem [11], the Freudenthal suspension and Blakers–Massey theorems [10], the existence of Eilenberg–MacLane spaces [16], the Mayer–Vietoris sequence [6], the Serre spectral sequence [8], the James construction [4], and the computation of $\pi_4(S^3)$ [18]. These are interesting and non-trivial results, but they have been well-understood since around the middle of the last century. It is natural to ask if synthetic homotopy theory provides value beyond reproducing old results.

There are many possible answers to this question and we do not have space to discuss all the merits of HoTT. One commonly cited point is that HoTT has the benefit of amenability to formalisation. Indeed this is a major selling point of Martin-Löf type theory, variants of which have been implemented in interactive theorem provers including Coq, Agda, and Lean. An interactive theorem prover, or proof assistant, is a piece of software that helps its user write mathematical proofs in a formal language, while checking the correctness of the proof. Since HoTT builds on Martin-Löf type theory, it is also suitable for formalisation in such proof assistants, and indeed all the results in synthetic homotopy theory mentioned above have been formalised. Formalisation of mathematics offers various benefits including ease of collaboration and increased trust in proofs. In principle one could also formalise homotopy theory 'analytically', but in practice such developments encounter technical difficulties and are still far behind their synthetic counterparts.

Still, one would hope for HoTT to lead to proofs of new results of relevance to mainstream homotopy theory. Perhaps the most frequently cited example of this is the synthetic proof of the Blakers–Massey theorem. The original Blakers–Massey theorem is a statement about connectivity in the classical setting of spaces. This was since generalised in different directions: from a connectivity statement to more general statements about 'cellular inequalities' [7], and separately to connectivity statements in the more general setting of ∞-topoi [21]. The synthetic perspective resulted in a new, simpler proof which simultaneously generalised the previous ones, to get a statement about modalities (corresponding to cellular inequalities) in ∞-topoi [1].

The main contribution of this thesis is a vastly more informative refinement of the Blakers–Massey theorem which we call *the zigzag construction*. The setup is the following. Given a span of spaces $B \leftarrow A \rightarrow C$, one can form the (homotopy) pushout D, and then the pullback $B \times_D C$, which comes with a map $A \rightarrow B \times_D C$. We can thus think of A as a (coarse) approximation to $B \times_D C$, and the Blakers–Massey theorem describes the accuracy of this approximation. The zigzag construction on the other hand provides an *exact* description of $B \times_D C$, via a sequence of approximations starting from A. One recovers a strong form of the Blakers–Massey theorem as a corollary, by analysing how the approximations evolve. The zigzag construction is a novel construction of relevance to mainstream homotopy theory, while originating in synthetic homotopy theory and using many ideas from there. Its closest

classical counterpart is the James construction, which is closely related to the zigzag construction but less widely applicable.

The second contribution of this thesis consists of some results on delooping in synthetic homotopy theory. In particular we establish some results in the direction of *stabilisation*, which is the phenomenon that deloopings are unique in certain situations.

1.1 Type theory

In this section we give a brief, informal, and opinionated introduction to Martin-Löf type theory. For a more precise description, we refer to [22].

The starting point for dependent type theory is the idea that whenever we speak of a mathematical object, we should already have in mind what type of object it is. For example 4 is a natural number, and sin is a function from real numbers to real numbers. If A is a type, we write $a : A$ to express that a is an object of type A. *Object* is not the standard terminology; one use various words like *term*, *element*, *point*, or *inhabitant* (of a given type) to evoke various intuitions. Referring to 'elements' of a given type suggests that we think of types as *collections of things*, much like sets in set theory, but there is an important distinction between the two. In set theory, the elementhood relation $x \in y$ is a mathematical statement that can be proved or disproved. In type theory, we speak of a typing *judgement* $a : A$ which belongs to the grammar of type theory.

Many constructions in mathematics are parametrised by some element of a given type. We write $x : A \vdash y : B$ to express that whenever we have some element x of type A, then we also have an object y of type B, whose definition may depend on x. We can thus think of y as some construction parametrised by x. In *dependent* type theory, we also allow the type B to depend on $x : A$, written $x : A \vdash B(x)\ \mathsf{Type}$. In natural language, we say that B is a type family over A. That which comes before the turnstile $\vdash$ is called a *context*, and it can contain an arbitrary finite (possibly empty) sequence of *variables*, like $x : A, y : B(x), z : C(x, y) \vdash D(x, y, z)\ \mathsf{Type}$. Importantly, all the rules of type theory are valid in arbitrary contexts.

Type formers

We assume certain *type formers* that allow to construct new types from old ones. For example, given a type family B over A, we can form the Π-*type* $(x : A) \to B(x)$, traditionally denoted $\Pi_{x:A} B(x)$. Elements of $(x : A) \to B(x)$ are thought of as *dependent functions* from A to B. That is, given an element $f : (x : A) \to B(x)$ and an element $a : A$, we can form an element $f(a) : B(a)$. This is referred to as an *elimination rule* for the Π-type, since it explains how we can use its elements. Conversely, there is also an introduction rule, which explains how to *form* elements of a Π-type: given an element $y : B(x)$ parametrised by an arbitrary $x : A$, or more formally $x : A \vdash y : B(x)$, we have an element $\lambda x.y : (x : A) \to B(x)$. More colloquially, we sometimes

write $x \mapsto y$ in place of $\lambda x.y$. The elimination rule (function application) and the introduction rule (lambda abstraction) are related by a *computation rule*, expressing that $(\lambda x.y)(a)$ coincides with the result of substituting x for a in y.

The computation rule is expressed in terms of *judgmental equality*. Given elements $a, b : A$ of the same type, we write $a \equiv b$ for the judgment expressing that a and b are equal 'on the nose'. Like the typing judgment $a : A$, the judgment $a \equiv b$ belongs to the grammar of type theory; it is wholly distinct from the usual mathematical notion of equality.

However, type theory does also have a way of dealing with mathematical equality, which plays a central role in HoTT. It is expressed using *identity types*. The formation rule for identity types says that given a type A with elements $a, b : A$, we may form a type $a = b$. Importantly, we can only talk about the type $a = b$ when we know that a and b have the same type; it does not make sense to compare elements of different types. An element of $a = b$ is thought of as an *identification* of a and b. The introduction rule for identity types says that every element is equal to itself: for any $a : A$ we have an element $\mathsf{rfl}_a : a = a$. The elimination rule expresses that rfl_a is the *only* element of the identity type. Naively, one might expect this to mean that for any $p : a = b$, we have that $p = \mathsf{rfl}_a$. But there's a problem with this: p has type $a = b$ whereas rfl_a has type $a = a$ so it does not make sense to ask if they can be identified. Instead, the elimination rule says the following. Suppose we have an element $a : A$, and a type family $P(b, p)$ that depends on $b : A$ and $p : a = b$. Assuming we have a term $d : P(a, \mathsf{rfl}_a)$. Then for any $b : A$, $p : a = b$, we have a term $J(P, d, b, p) : P(b, p)$. The computation rule says that $J(P, d, a, \mathsf{rfl}_a)$ can be identified with d. In the context of HoTT, this elimination rule is referred to as (based) *path induction*.

The identity type is an example of an (indexed) inductive type, since it is freely generated by some elements (namely rfl_a). Another important example of an inductive type is the type $\mathbb{N}$ of natural numbers. It is freely generated by an element $0 : \mathbb{N}$ together with successors, $x : \mathbb{N} \vdash S(x) : \mathbb{N}$. In HoTT, one also makes use of *higher inductive types*. A higher inductive type is freely generated by some elements and some identifications between elements. For example, given types A, B with a type family $R(a, b)$ parametrised by $a : A$ and $b : B$, we may form the *pushout*, denoted $A \sqcup^R B$. This is freely generated by elements $a : A \vdash \mathsf{inl}(a) : A \sqcup^R B$ and $b : B \vdash \mathsf{inr}(b) : A \sqcup^R B$ together with identifications:

$$a : A, b : B, r : R(a, b) \vdash \mathsf{glue}(\mathsf{r}) : \mathsf{inl}(a) = \mathsf{inr}(b).$$

Other basic type formers include the empty type, the unit type, and Σ-types. The empty type 0 (also written $\emptyset$) has *no* introduction rule – it is meant to have no elements. The corresponding elimination rule is as general as possible: given an element $x : 0$ and any type A we have an element $\mathsf{emptyrec}(x) : A$.

The unit type 1 has a very simple introduction rule, which says that 1 has an element $\star : 1$. The elimination rule expresses that $\star$ is the only element: for any type family P over 1, $p : P(\star)$ and $x : 1$, we have an element $\mathsf{unitrec}(p, x) : P(x)$, with $\mathsf{unitrec}(p, \star) = p$. This can equivalently be expressed by saying that $x = \star$ for any $x : 1$.

There are different ways of presenting Σ-types, also called *dependent pair* types, but the idea is the following. Given an type A and a type family B over A, we can form a type $(a : A) \times B(a)$, traditionally written $\Sigma_{a:A} B(a)$.[1] The introduction rule for Σ-types says that given elements $p : A$ and $q : B(a)$ we have an element $(p, q) : (a : A) \times B(a)$. One further has some rules expressing that all elements of $(a : A) \times B(a)$ are in some sense of this form.

Equivalences, propositions, and sets

An important notion in type theory is that of *invertible maps*, or *equivalences*. Given a function $f : A \to B$ we could say that f is invertible, or an equivalence, when we have a function $g : B \to A$ *inverse* to f in the sense that $g(f(a)) = a$ for all $a : A$ and $f(g(b)) = b$ for all $b : B$. It is often important to *internalise* this notion, to define a *type* $\mathsf{isEquiv}(f)$ whose elements are witnesses that f is invertible. To this end one has to be a bit more careful: one defines $\mathsf{isEquiv}(f)$ to be the Σ-type consisting of $g, h : B \to A$ together with functions $p : (x : A) \to g(f(x)) = x$ and $q : (y : B) \to f(h(y)) = y$. Importantly, all the type formers above can be shown to *respect* equivalences in an appropriate sense. We write $A \simeq B$ for the type $(f : A \to B) \times \mathsf{isEquiv}(f)$.

Following Voevodsky, a type is said to be *contractible* if it is equivalent to the unit type. A type X is said to be a *proposition* if all its identity types $a = b$ with $a\, b : X$ are contractible; this expresses that X has at most one element. A type is said to be a *set* , or 0-truncated, if all its identity types are propositions. In ordinary, non-homotopical mathematics, everything would be a set in this sense. The homotopical nature of type theory is made possible because we don't ask that all types be sets.

Universes and the identity types of $\mathbb{N}$

The elimination rule for $\mathbb{N}$ expresses that $\mathbb{N}$ is *freely* generated by $0 : \mathbb{N}$ and $S : \mathbb{N} \to \mathbb{N}$. This in particular means that given another type X with elements $0_X : X$ and $S_X : X \to X$, we obtain a function $\mathbb{N} \to X$ out of $\mathbb{N}$. One often needs a similar way to define *type families* over $\mathbb{N}$. To solve this, and similar problems, it is very often helpful to assume that some kind of *universes* are available.

In any situation where we have a type A with a type family B over A, we can think of B as an A-indexed collection of types. In other words, we can think of A as a 'collection of types', the type corresponding to (or 'encoded by') an element $a : A$ being $B(a)$. When we think of a type family as a collection of types, we often use the term *universe* to refer to this situation. Naively, we might hope to have a 'type of all types', i.e. a type U with a type family El over U such that for *any* type A, we have a corresponding element $\mathsf{code}_A : \mathsf{U}$ with A equivalent to $\mathrm{El}(\mathsf{code}_A)$. This is asking for too much and leads to a paradox familiar from set theory. Instead, what one can reasonably ask for is universes

[1] This notation evokes an analogy between types and numbers, where Σ is analogous to summation $\sum_{i=0}^{n} a_i$. We prefer to avoid this notation since the practice of putting complicated expressions in subscripts scales poorly.

'big enough for any given purpose'. For example, given a finite list of type families B_i over A_i, for $i = 0, \ldots, n$, we can ask for a universe containing each of the given collections of types. Often we also want to ask that the resulting universe is closed under some collection of type formers. Given a universe, i.e. a type family El over U, we say that types of the form $\mathsf{El}(A)$ (or types equivalent to types of this form) are U-*small*. Given a universe U, we can thus define families of U-small types over $\mathbb{N}$ by induction, since they are simply maps of types $\mathbb{N} \to \mathsf{U}$.

A major theme of HoTT is that of describing identity types. For example, we have the following description of identity types $m = n$ where $m, n : \mathbb{N}$:

- $0 = 0$ is contractible, i.e. equivalent to the unit type 1.
- $0 = S(n)$ and $S(n) = 0$ are empty, i.e. equivalent to the empty type 0.
- $S(m) = S(n)$ is equivalent to $m = n$.

The proof of the claim above involves first defining a type family $P : \mathbb{N} \to \mathbb{N} \to \mathsf{U}$ recursively using the rules above and then arguing that $P(m, n)$ is equivalent to $m = n$, essentially by showing that P has the universal property of the identity type. This method of proof is called *encode-decode*. There are similarly canonical descriptions of identity types of the empty type, the unit type, Σ-types, and Π-types[2].

Identity types of higher inductive types and descent

The situation becomes more subtle once one starts to look at identity types of *higher* inductive types. For example, we can construct a (homotopical) circle S^1 by gluing two points along two parallel paths. Formally, this is represented by a pushout $1 \sqcup^{2} 1$, where 2 denotes a type with exactly two elements. Let us pick a base point $b : S^1$ (e.g. $\mathsf{inl}(\star)$). One can describe S^1 as being freely generated by the element $b : S^1$ and an identification $\mathsf{loop} : b = b$ (given by $\mathsf{glue}(0) \cdot \mathsf{glue}(1)^{-1}$). One expects that the 'loop space' $b = b$ is equivalent to the type of *integers*. Indeed any integer $n : \mathbb{Z}$ should determine a loop $b = b$ given by composing loop with itself n times.

In order to use the encode-decode method sketch above, we need a method for defining type families over S^1. To see what this might look like, first note that given a type family P over S^1, we in particular have a type $P(b)$. The identification $\mathsf{loop} : b = b$ induces a self-equivalence $P(b) \simeq P(b)$.[3] It turns out that the right result to ask for is that *any* type B with a self-equivalence $e : B \simeq B$ determines a type family over S^1, whose fibre over b is B and such that the self-equivalence of the fibre induced by loop corresponds to e. This principle is referred to as *descent* for S^1.

[2]The characterisation of identity types of Π-types says that if $f, g : (a : A) \to B(a)$, then $f = g$ is equivalent to $(a : A) \to f(a) = g(a)$. This is called *function extensionality* and should be treated as a basic assumption of type theory.

[3]This is an instance of a general fact that given a type family P over A and elements $a, b : A$ with an identification $h : a = b$, we obtain an equivalence $P(a) \simeq P(b)$. This general fact is proved directly by path induction.

Let us now consider what happens if we try to prove descent using universes. To construct a type family over S^1, we would define a function $P : S^1 \to \mathsf{U}$ to some universe U. This we can do using the elimination rule for S^1: to define P, we have to give an element $B : \mathsf{U}$ with a self-identification $B = B$. Being asked to produce an self-identification $B = B$ seems a bit much when all we should need is a self-equivalence $\mathrm{El}(B) \simeq \mathrm{El}(B)$. This is where *univalence*, formulated by Voevodsky, comes in.

We say that a type family El over U is *univalent* if we have the following equivalence for all $A, B : \mathsf{U}$.

$$(A = B) \simeq (\mathrm{El}(A) \simeq \mathrm{El}(B)).$$

Thinking of U as a collection of types, this expresses the idea that U contains 'at most one' copy of each type, which is where the word univalent comes from.

Univalence is precisely the property we need in order to prove descent using elimination into a universe. More precisely, to prove descent for S^1 it suffices to assume that every type is contained in a univalent type family. In general the characterisation of identity types of higher inductive types is intimately tied to descent, and hence indirectly to univalence.

In particular, given descent for S^1 one can show that its loop space is the type $\mathbb{Z}$ of integers. For a proof of this, as well as some more discussion of identity types of pushouts, we refer to [15].

Chapter 2

Summary of included papers

Path spaces of pushouts

The motivation for this work is the problem of describing identity types (or *path spaces*) of pushouts. Explicitly, given types A, B with a type family $R(a, b)$ parametrised by $a : A$ and $b : B$, how to describe identity types like $\mathsf{inl}(x) = \mathsf{inr}(y)$, $\mathsf{inl}(x) = \mathsf{inl}(x')$, $\mathsf{inr}(y) = \mathsf{inr}(y')$ with $x, x' : A$ and $y, y' : B$? The exists a considerable body of earlier work on aspects of this problem in HoTT. For example, the Seifert-van Kampen theorem gives a description of 0-truncated identity types [11], the identity types of S^1 are known exactly, as well as identity types of pushouts along a monomorphism [23]. Kraus and von Raumer gave a universal property for identity types in any pushout [15], but the recursive nature of this description makes it a priori difficult to analyse. An important problem, dating back to the HoTT book, was to show that the identity types of $1 \sqcup^X 1$ are 0-truncated if X is 0-truncated [13]. Some partial results were known [14], but prior to this work the general problem was wide open with experts suspecting that the desired result might not be provable in HoTT.

The contribution of this work is a construction, which we call *the zigzag construction*, which gives a precise and definitive description of identity types of pushouts. Unlike Kraus and von Raumer's description, the zigzag construction is non-recursive (in a technical sense), which makes it amenable to analysis. In particular we use it to prove a generalised Blakers–Massey theorem and to show that the identity types of $1 \sqcup^X 1$ are 0-truncated if X is 0-truncated.

Since we expect the zigzag construction to be of interest also outside type theory, we have written this paper in categorical, or diagrammatic, language, with the hope that it can be understood by type theorists and homotopy theorists alike.

Eilenberg–MacLane spaces and stabilisation

A central concept in algebra is the notion of a *group*, which can be understood as an algebraic representation of the symmetries of some object. In a type-theoretic setting, when we talk of an object x, we should have in mind that x has some type, call it X. In this case the symmetries of x should (according to the *structure identity principle*) correspond to identifications $x = x$. We say that $x = x$ is the loop space of X and that X is a delooping of $x = x$. It is natural to turn this into a definition: an (∞-)group is something which has the structure of a loop space $x = x$. This leads to a synthetic perspective on group theory which is explored in an upcoming book project [3].

An important phenomenon in the theory of ∞-groups is that of *stabilisation*. This is the phenomenon that pointed types which are sufficiently connected and truncated admit unique ∞-group structures (deloopings). Algebraically, stabilisation can be understood as coming from the fact that the data of a group structure on a pointed type (A, e) involves an H-space structure, i.e. a map $\mu : A \times A \to A$ with compatible identifications $(a : A) \to \mu(a, e) = a$ and $(a : A) \to \mu(e, a) = a$, subject to further coherences. By wedge connectivity the type of H-space structures on (A, a) is contractible if A is sufficiently connected and truncated.

Stabilisation results have been proven in HoTT using the Freudenthal suspension theorem [5]. The purpose of this paper is to revisit these results using more direct methods, avoiding the use of higher inductive types. We present in particular descriptions of deloopings of types and maps. This gives an elementary type-theoretic account of how, in favourable situations, a delooping of (A, a) can be understood as the type of A-torsors. This can be compared with ∞-categorical accounts [19, Theorem 7.2.2.26][20].

Bibliography

[1] Mathieu Anel, Georg Biedermann, Eric Finster and André Joyal. "A generalized Blakers–Massey theorem". In: *Journal of Topology* 13.4 (2020), pp. 1521–1553 (cit. on p. 4).

[2] Steve Awodey and Michael A. Warren. "Homotopy theoretic models of identity types". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 146 (2009), pp. 45–55 (cit. on p. 3).

[3] Marc Bezem, Ulrik Buchholtz, Pierre Cagne, Bjørn Ian Dundas and Daniel R. Grayson. *Symmetry*. `https://github.com/UniMath/SymmetryBook` (cit. on p. 12).

[4] Guillaume Brunerie. "The James Construction and $\pi_4(S^3)$ in Homotopy Type Theory". In: *Journal of Automated Reasoning* 63 (2 2019), pp. 255–284 (cit. on p. 4).

[5] Ulrik Buchholtz, Floris van Doorn and Egbert Rijke. "Higher groups in homotopy type theory". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. 2018, pp. 205–214 (cit. on p. 12).

[6] Evan Cavallo. "Synthetic Cohomology in Homotopy Type Theory". MA thesis. Carnegie Mellon University, 2015 (cit. on p. 4).

[7] Wojciech Chachólski, Jérôme Scherer and Kay Werndli. "Homotopy excision and cellularity". In: *Annales de l'Institut Fourier*. Vol. 66. 6. 2016, pp. 2641–2665 (cit. on p. 4).

[8] Floris van Doorn. "On the Formalization of Higher Inductive Types and Synthetic Homotopy Theory". PhD thesis. Carnegie Mellon University, 2018. URL: `https://arxiv.org/abs/1808.10690` (cit. on p. 4).

[9] Martin Hofmann and Thomas Streicher. "The groupoid interpretation of type theory". In: *Twenty-five years of constructive type theory (Venice, 1995)*. Ed. by Giovanni Sambin and Jan M. Smith. Vol. 36. Oxford Logic Guides. New York: Oxford University Press, 1998, pp. 83–111 (cit. on p. 3).

[10] 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. New York, NY, USA: Association for Computing Machinery, 2016, 565–574. DOI: 10.1145/2933575.2934545 (cit. on p. 4).

[11] Kuen-Bang Hou (Favonia) and Michael Shulman. "The Seifert-van Kampen Theorem in Homotopy Type Theory". In: *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*. Ed. by Jean-Marc Talbot and Laurent Regnier. Vol. 62. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016, 22:1–22:16. DOI: 10.4230/LIPIcs.CSL.2016.22 (cit. on pp. 4, 11).

[12] Chris Kapulkin and Peter LeFanu Lumsdaine. *The Simplicial Model of Univalent Foundations (after Voevodsky)*. 2018. arXiv: 1211.2851 [math.LO] (cit. on p. 3).

[13] Nicolai Kraus and Thorsten Altenkirch. "Free Higher Groups in Homotopy Type Theory". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. ACM, July 2018, 599–608. DOI: 10.1145/3209108.3209183. URL: http://dx.doi.org/10.1145/3209108.3209183 (cit. on p. 11).

[14] Nicolai Kraus and Jakob von Raumer. "A rewriting coherence theorem with applications in homotopy type theory". In: *Mathematical Structures in Computer Science* 32.7 (Aug. 2022), 982–1014. ISSN: 1469-8072. DOI: 10.1017/s0960129523000026. URL: http://dx.doi.org/10.1017/S0960129523000026 (cit. on p. 11).

[15] Nicolai Kraus and Jakob von Raumer. "Path Spaces of Higher Inductive Types in Homotopy Type Theory". In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2019, pp. 1–13. DOI: 10.1109/LICS.2019.8785661 (cit. on pp. 9, 11).

[16] Daniel R. Licata and Eric Finster. "Eilenberg-MacLane spaces in homotopy type theory". In: *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. CSL-LICS '14. Vienna, Austria: Association for Computing Machinery, 2014. DOI: 10.1145/2603088.2603153 (cit. on p. 4).

[17] Daniel R. Licata and Michael Shulman. "Calculating the Fundamental Group of the Circle in Homotopy Type Theory". In: *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*. 2013, pp. 223–232. DOI: 10.1109/LICS.2013.28 (cit. on p. 4).

[18] Axel Ljungström and Anders Mörtberg. *Formalising and Computing the Fourth Homotopy Group of the 3-Sphere in Cubical Agda*. 2024. arXiv: 2302.00151 [math.AT]. URL: https://arxiv.org/abs/2302.00151 (cit. on p. 4).

[19] Jacob Lurie. *Higher topos theory*. Princeton University Press, 2009 (cit. on pp. 4, 12).

[20] Thomas Nikolaus, Urs Schreiber and Danny Stevenson. "Principal ∞-bundles: general theory". In: *Journal of Homotopy and Related Structures* 10.4 (June 2014), 749–801. ISSN: 1512-2891. DOI: 10.1007/s40062-014-0083-6 (cit. on p. 12).

[21] Charles Rezk. "Toposes and homotopy toposes". 2005. URL: https://rezk.web.illinois.edu/homotopy-topos-sketch.pdf (cit. on p. 4).

[22] Egbert Rijke. *Introduction to Homotopy Type Theory*. 2022. arXiv: 2212.11082 [math.LO]. URL: https://arxiv.org/abs/2212.11082 (cit. on p. 5).

[23] Christian Sattler and Andrea Vezzosi. "Partial Univalence in n-truncated Type Theory". In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '20. Saarbrücken, Germany: Association for Computing Machinery, 2020, 807–819. ISBN: 9781450371049. DOI: 10.1145/3373718.3394759. URL: https://doi.org/10.1145/3373718.3394759 (cit. on p. 11).

[24] Michael Shulman. *All $(\infty, 1)$-toposes have strict univalent universes*. 2019. arXiv: 1904.07004 [math.AT]. URL: https://arxiv.org/abs/1904.07004 (cit. on p. 4).

[25] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book, 2013 (cit. on pp. 3, 4).

[26] Vladimir Voevodsky. "A very short note on the homotopy λ-calculus". http://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/Hlambda_short_current.pdf. 2006 (cit. on p. 3).