Formalising the Symmetry Book

Formalising the Symmetry Book using the UniMath library.

Pallabi Sree Sarker[1]

Supervisor(s): Benedikt Ahrens[1]**, Kobe Wullaert**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Pallabi Sree Sarker
Final project course: CSE3000 Research Project
Thesis committee: Benedikt Ahrens, Kobe Wullaert, Neil Yorke-Smith

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

To address the challenge of the time-consuming nature of proofreading proofs, computer proof assistants—such as the Coq proof assistant—have been developed. The Univalent Mathematics project aims to formalise mathematics using the Coq proof assistant from a univalent perspective, which is based on homotopy type theory.

The Symmetry book is a textbook about symmetries in mathematics written from a univalent viewpoint. This paper focuses on formalising the proofs in chapter 3 of the Symmetry book using the UniMath Coq library. Currently, the book is partly formalised in the Agda UniMath library. In this paper, we aim to formalise the chapter using the UniMath Coq library to verify its correctness. We have successfully formalised all the theorems in sections 3.1 to 3.3, along some other minor proofs.[1]

1 Introduction

Proofs in mathematics can be very lengthy, intricate texts, which are difficult to verify. Humans verifying these proofs is not only time-consuming, but also error-prone. This is why we use computer proof assistants. Computer proof assistants such as the Coq[2] proof assistant use type-checking to verify the correctness of a proof.

The Univalent Mathematics[1] project is a project which aims to formalise mathematics using the Coq proof assistant from a univalent point of view. That is, the foundation of its mathematics is homotopy type theory (HoTT)[2], rather than set theory.

The Symmetry book[3] is a work-in-progress textbook meant for bachelor's student about symmetries in mathematics. It is written from a Univalent point of view.

In this paper we will formalise the proofs in chapter 3 of this book using the UniMath Coq library. Currently, small parts of the Symmetry book are formalised in the Agda UniMath library. Our goal is to formalise more of chapter 3 in the Coq UniMath library so the correctness can be verified and various theorems about the circle can be reused in future formalisations. Specifically, we aim to answer the research question: *Can we formalise the proofs in chapter 3 of the Symmetry book using the UniMath Coq library to verify their correctness?*

This section introduces the basic relevant concepts of homotopy type theory to the reader. Followed by section 2 which outlines the goal of this research paper and our approach. Section 3 provides informal proofs of everything we want to formalise, along with links to the exact lines in the code where the formalisations are. Section 4 provides a description of what AI tools were used during this project and how. Furthermore, the challenges faced during this project are described in section 5. The considerations in regards to a responsible approach to research are discussed in section 6. Lastly, the conclusions and comments on future work are discussed in section 7.

1.1 A note on notation

As this paper formalises the proofs in the Symmetry book, we are using the same notation as the Symmetry book. The most important of which is that the identity type is denoted as $\overset{=}{\to}$ and equivalences as $\overset{\simeq}{\to}$. Furthermore, the constructors base and loop of the circle are denoted as $\bullet$ and $\circlearrowleft$ respectively.

1.2 Homotopy Type Theory

Homotopy type theory forms the basis of Univalent Foundations of Mathematics. The following subsection serves as a very brief introduction to the basic concepts of homotopy type theory that are relevant to this paper. All the information was taken from the Homotopy Type Theory book[2] and the Symmetry book[3], where more details can be found.

Homotopy Type theory is a dependent type theory which regards types as spaces, and elements of those types as points within those spaces. Two elements are considered equal if there exists a path between those elements. This path has a type as well, called the identity type. The identity type has a constructor, namely $\text{refl}_x : x \overset{=}{\to} x$. To illustrate, figure 1 shows a space, or type A, with two elements, a and b. These two elements are equal, as a path p exists between them. Contrarily, figure 2 shows some space B which contains two elements a and b which are *not* equal, as there is no possible path between them.

Inductive types are defined by constructors which generate points of that type. For example, the boolean type is defined by two constructors, namely true and false. Higher inductive types are inductive types (that is, they have constructors which generate points of that type), but they also have constructors which generate paths (equalities) and higher paths (equalities of equalities) in that type.

[1] This abstract was written with help of ChatGPT. The prompt "Write an abstract based on the following introduction: <introduction>" was used. Then, the author edited the generated text to make it more coherent and include more of the necessary information.

[2] https://coq.inria.fr/

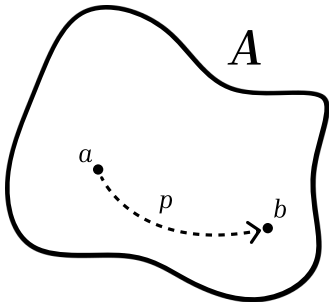

Figure 1: Illustration of some type A. A contains two elements a and b, which are equal as denoted by path p.

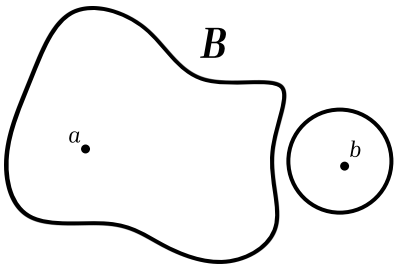

Figure 2: Illustration of some type B. B contains two elements a and b, which are not equal.

Chapter 3 of the Symmetry book concerns a specific higher inductive type, namely the circle S^1. It is defined by the constructors $\bullet : S^1$ and $\circlearrowleft : \bullet \xrightarrow{=} \bullet$. The first constructor generates a point on the circle. The second constructor generates a loop from the point to itself. Figure 3 shows an illustration of this circle with its two constructors. It must be noted that $\circlearrowleft \neq \text{refl}_\bullet$.

Propositional truncation[2, Section 3.7]

A propositional truncation $||A||$ of a type A effectively collapses the type A into a new type which loses all information except for whether A was inhabited–that is, whether A had any elements–or not. $||A||$ thus becomes the proposition 'was A inhabited?'

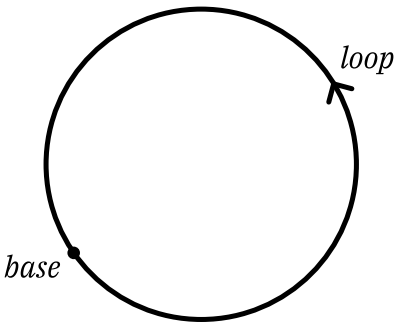

Figure 3: Illustration of the circle S^1. The circle has two constructors: $\bullet$ (base) and $\circlearrowleft$ (loop).

Induction and Recursion Principles of the circle[4, Section 21.1][2, Section 2.3, 6.4]

The induction principle is the principle that allows us to prove something for every element of an inductive type. In the specific case of the induction principle of the circle, in order to prove $\prod_{x:S^1} P(x)$ (that is, for all elements of the circle, $P(x)$), we must provide an element $u : P(\bullet)$ and an element $\ell : \text{tr}_P(\circlearrowleft, u) = u$, where tr is the transport function. The induction principle gives a function

$$f : \prod_{x:S^1} P(x)$$

given the aforementioned elements which can be used to prove something for all elements on the circle. The induction principle furthermore states that we have the identification $\text{apd}_f(\circlearrowleft) \xrightarrow{=} \ell$.

The recursion principle of the circle states that given some type A together with some element $a : A$ and $\ell : a \xrightarrow{=} a$, we have the function $f : S^1 \to A$ which sends $\bullet$ to a. Furthermore we have the identification $\text{ap}_f(\circlearrowleft) \xrightarrow{=} \ell$.

2 Formalization of Proofs in Chapter 3: A Homotopy Type Theory Approach

The following section describes the problem we are trying to solve in this research paper and the approach we took to solve it.[3]

2.1 Problem Description

The Symmetry book is aimed at bachelor's students about different symmetries found in mathematics. It is written from a Univalent point of view. This project aims to formalise some of the proofs in chapter 3 of the Symmetry book using the UniMath Coq library, as it contains proofs that have not yet been formalised in said library.

The research question is: *Can we formalise the proofs in chapter 3 of the Symmetry book using the UniMath Coq library to verify their correctness?*

2.2 Approach

The approach of the project was split up into two phases. The preparatory phase—where I learned about Homotopy Type Theory, the UniMath library and the Coq proof assistant—and the research phase, where I formalised proofs from chapter 3 of the symmetry book using the UniMath Coq library.

[3]This paragraph contains suggestions from Copilot.

Preparatory Phase

During the preparatory phase, I learned about Homotopy Type Theory, the UniMath library, and the Coq proof assistant. I learned about Homotopy Type Theory using part of Egbert Rijke's Introduction to Homotopy Type Theory[4] and materials from the HoTTest summer school[5]. I learned about UniMath and the Coq proof assistant using materials from the UniMath school 2022[6].

I also prepared my research by installing the Coq proof assistant and the UniMath library.

Research Phase

During the research phase, I formalised all the theorems in sections 3.1 to 3.3 and lemma 3.1.6 of the Symmetry book. The formalisation of the proofs followed the order as presented in the book. I started with the formalisation of the definition of the circle and then proceeded to formalise sections 3.1 to 3.3. While some of the minor proofs were not formalised due to time constraints, all the theorems in these chapters have been.

Additionally, I formalised construction 2.25.4(2), which, although not found in chapter 3, was necessary to prove a theorem in section 3.3. In total, I formalised four proofs, of which two theorems. Due to time constraints, I was not able to formalise all the proofs in chapter 3 of the Symmetry book.

To assist me in writing the formalisations, I used GitHub Copilot[4] and ChatGPT[5], which helped me debug my proofs and give suggestions (for more details, please refer to section 4).

3 Implementation

The following section describes the formalisation of the proofs. It also goes through the steps of the proof in a more human-readable format, to make them easier to understand for the reader. All proofs contain links to the corresponding lines in the GitHub repository[6].

3.1 Defining the Circle

Defining a circle, which is a higher inductive type, is quite difficult in Coq, as Coq does not have default support for higher inductive types. Because of this, the HoTT Coq[7, Torus.v] library was used as a source of inspiration for defining the Circle, as well as some help from the UniMath Zulip community.

[4]https://github.com/features/preview/copilot-x

[5]https://chat.openai.com

[6]https://github.com/pujiii/rp-code

The above means that we cannot simply define the circle as follows:

```
Inductive S1 : Type :=
| base : S1.
| loop : base = base.
```

Instead, we must use a workaround to be able to define the circle. We must define our loop constructor as an axiom. This is done as follows:

```
Private Inductive S1 : Type :=
| base : S1.

Axiom loop : base = base.
```

Furthermore, we have to define the induction and recursion principles ourselves. The entire definition of the circle can be found in Circle.v as S1.

3.2 Proof formalisations

The following section discusses all the proof formalisations that I have made. The proofs are all formalised in Coq and can be found in Circle.v and every subsection below has a link to the respective line number.

The circle is connected

We want to prove lemma 3.1 as taken from the Symmetry book [3, Lemma 3.1.6]. The formalisation of this proof can be found in Circle.v under the name `circle_connected`.

Lemma 3.1. *The circle is connected.*

$$\prod_{z:S^1} ||\bullet \xrightarrow{=} z||$$

Proof. We want to construct an element of type $||\bullet \xrightarrow{=} z||$ for all $z : S^1$. We use circle induction on z. As a base case we take $\text{refl}_\bullet : ||\bullet \xrightarrow{=} \bullet||$. The loop case immediately follows, as $||\bullet \xrightarrow{=} \bullet||$ is a proposition. □

The evaluation function is an equivalence

We want to prove theorem 3.2 as taken from the Symmetry book [3, Theorem 3.1.2]. The type of our proof is `isweq(ev A)`, where A is some arbitrary type. `isweq` is the type necessary to construct in order to prove a function is an equivalence. The formalisation can be found in Circle.v under the name `evisweq`.

Theorem 3.2. *The evaluation function, of type:*

$$\text{ev}_A : (S^1 \to A) \to \sum_{a:A} (a \xrightarrow{=} a)$$

defined as $\mathsf{ev}_A(g) :\equiv (g(\bullet), \mathsf{ap}_g(\circlearrowleft))$ *is an equivalence.*

Proof. To prove ev_A is an equivalence we use construction A.1. We must first define its inverse ve_A. We define ve_A which takes as argument a tuple (a, b) to return a function which sends $\bullet$ to a and $\circlearrowleft$ to b. We do this using the recursion principle.

Furthermore, we must construct an identification of type $\mathsf{ev}(\mathsf{ve}((a, l)))$. We do this using using the identification $ap_f(\circlearrowleft) \xrightarrow{=} \ell$.

Lastly, we must construct an identification of type $\mathsf{ve}(\mathsf{ev}(f)) \xrightarrow{=} f$. We do this as follows. By function extensionality it suffices to construct an element of type $\prod_{x:S^1} \mathsf{ve}(\mathsf{ev}(f(x))) \xrightarrow{=} f(x)$. We can construct this using circle induction. The base case is trivial. The inductive case requires us to use construction A.2, followed by the identification the recursion principle has given us, namely $ap_f(\circlearrowleft) \xrightarrow{=} \ell$. Lastly, we use the identification $p^{-1} \cdot p \xrightarrow{=} \text{refl}$.

□

Set families

We want to prove lemma 3.3 as taken from the Symmetry book [3, Construction 2.25.4(2)]. We provide the formalisation of the proof of this lemma in Circle.v under the name `set_families`.

Lemma 3.3. *We have the following equivalence.*

$$(A \to Set) \xrightarrow{\simeq} \sum_{B:\mathcal{U}} \sum_{f:B\to A} \prod_{a:A} isSet(f^{-1}(a))$$

Proof. We use the converse of construction A.1, which requires us to provide the following: function $h : \sum_{B:\mathcal{U}} \sum_{f:B\to A} \prod_{a:A} \text{isSet}(f^{-1}(a)) \to (A \to \text{Set})$, a function $g : (A \to \text{Set}) \to \sum_{B:\mathcal{U}} \sum_{f:B\to A} \prod_{a:A} \text{isSet}(f^{-1}(a))$, an identification $g(h(a)) \xrightarrow{=} a$ and an identification $h(g(a)) \xrightarrow{=} a$.

We start by providing a function $h : \sum_{B:\mathcal{U}} \sum_{f:B\to A} \prod_{a:A} \text{isSet}(f^{-1}(a)) \to (A \to \text{Set})$. We construct this as follows. From our input, we take $\prod_{a:A} \text{isSet}(f^{-1}(a)) \to (A \to \text{Set})$ by taking the second projection of the second projection. We use this for the definition of our output function, as we can create a set from an element of isSet.

Next, we must provide a function $g : (A \to \text{Set}) \to \sum_{B:\mathcal{U}} \sum_{f:B\to A} \prod_{a:A} \text{isSet}(f^{-1}(a))$. We call our input $\psi : (A \to \text{Set})$. We define our type $B :\equiv \sum_{a:A} \psi(a)$. We define our function $f :\equiv \text{pr}_1$. Lastly, we must provide a proof that for all a, $\text{isSet}(f^{-1}(a))$. We apply lemma A.3. We know that $\psi(a)$ is a set. Furthermore, we constructed our type B and function f such that $\psi(a) \xrightarrow{\simeq} f^{-1}(a)$. So we conclude that $f^{-1}(a)$ is a set.

Now we must provide an identification $g(h(a)) \xrightarrow{=} a$. Note that a is of type $\sum_{B:\mathcal{U}} \sum_{f:B\to A} \prod_{a:A} \text{isSet}(f^{-1}(a))$. We use lemma A.4. So, to provide this identification, it suffices to give an identification $p : \text{pr}_1(g(h(a))) \xrightarrow{=} \text{pr}_1(a)$ and an identification $\text{tr}_P(p, \text{pr}_2(g(h(a)))) \xrightarrow{=} \text{pr}_2(a)$ where $P(x) :\equiv \lambda x. \sum_{f:x\to A} \prod_{a:A} \text{isSet}(f^{-1}(a))$. The identification $p : \text{pr}_1(g(h(a))) \xrightarrow{=} \text{pr}_1(a)$ follows from lemma A.5.

We can construct identification $\text{tr}_P(p, \text{pr}_2(g(h(a)))) \xrightarrow{=} \text{pr}_2(a)$ as follows. We use lemma A.6. It suffices to give a proof that $\prod_{a:A} \text{isSet}(f^{-1}(a))$ is a predicate, which is immediate as we know isSet (x) is a predicate, and an identification between the first projections of both the left and right-hand side. We can construct this using minor manipulations.

Lastly, we must construct an identification $h(g(a)) \xrightarrow{=} a$. Following univalence and minor manipulations, it suffices to give the equivalence $f^{-1}(a) \xrightarrow{\simeq} f(a)$. We defined both f and B precisely so that this equivalence holds.

□

String of equivalences and a special case of the evaluation function is an equivalence

We want to prove theorem 3.4 as taken from the Symmetry book[3, Theorem 3.3.6]. We provide the formalisation of the proof of this lemma in Circle.v. It is subdivided into part 1 (`ev_set_equiv`) and part 2 (`string_of_equivalences`).

Theorem 3.4. *The evaluation function provides an equivalence*

$$\mathsf{ev}_{Set} : (S^1 \to Set) \to \sum_{X:Set} (X \xrightarrow{=} X)$$

defined by $\mathsf{ev}_{Set}(g) :\equiv (g(\bullet), ap_g(\circlearrowleft))$.

Consequently, we have a string of equivalences

$$\begin{aligned} SetBundle(S^1) &\xrightarrow{\simeq} (S^1 \to Set) && (1)\\ &\xrightarrow{\simeq} \sum_{X:Set} (X \xrightarrow{=} X) \xrightarrow{\simeq} \sum_{X:Set} (X \xrightarrow{\simeq} X) && (2)\\ &\xrightarrow{\simeq} \sum_{X:U} \sum_{f:X\to X} isSet(X) \times isEquiv(f) && (3) \end{aligned}$$

Proof. The first part follows directly from lemma 3.2.

For the second part, we will go step by step. The equivalence $\text{SetBundle}(S^1) \xrightarrow{\simeq} (S^1 \to Set)$ immediately follows from lemma 3.3. The equivalence $(S^1 \to Set)$ $\xrightarrow{\simeq} \sum_{X:\text{Set}} (X \xrightarrow{=} X)$ follows from the first part of our proof.

The equivalence $\sum_{X:\mathrm{Set}}(X \xrightarrow{=} X) \xrightarrow{\simeq} \sum_{X:\mathrm{Set}}(X \xrightarrow{\simeq} X)$ follows from univalence. Lastly, the equivalence $\sum_{X:\mathrm{Set}}(X \xrightarrow{\simeq} X) \xrightarrow{\simeq} \sum_{X:U} \sum_{f:X\to X} isSet(X) \times isEquiv(f)$ can be constructed with minor manipulations. □

4 Use of AI Tools

During this project, I have used several AI tools to aid me in my research. These tools are discussed in this section.

4.1 GitHub Copilot X

During the project, I used GitHub Copilot X[8] for suggestions on steps in my proofs, as well as asking it to explain errors to me. I tried to use it to fix bugs for me, but it was not very good at that. Occasionally I would get an idea from its responses, but its responses were always wrong. The suggestions were often also wrong, but sometimes they were correct. For example, one suggestion was to use `funextfun` in the proof of 3.2. The suggestions that were used in the final proofs are denoted with a comment in the code. Partial sentences of this paper were also written using suggestions from Copilot X. These parts are denoted with a footnote.

4.2 ChatGPT

Before getting access to Copilot X, I used ChatGPT[9] to help me explain code line-by-line, to better understand already written Coq code, it succeeded in this. I also tried to have it explain errors and correct errors for me, but like Copilot, it was not very good at this.

ChatGPT was also used to write some parts of this paper. These parts are denoted with a footnote.[7]

5 Challenges

During this project, I faced some challenges. This section discusses these challenges.

5.1 Learning Homotopy Type Theory

Before this project, I did not know Homotopy Type theory. Therefore, I had to learn it from scratch. As homotopy type theory is quite a niche field, there are not many resources available to learn it. However, the resources that existed[2][5][4] were sufficient.

5.2 Formalisations

Formalising proofs in Coq was a challenge. I had to learn how to use both Coq and the UniMath library. This was quite a difficult task, as formalising proofs is something that requires practice and time. I did not have much time during this project to practice, so it was quite a struggle. However, with help from my supervisor, responsible professor, and the UniMath community on Zulip[8] (in particular the user Niels van der Weide), I learned as I went.

6 Responsible Research

During this project, we take a responsible approach to research. The following section describes the issues and non-issues relating to responsible research in this project.

As data is not used in this project, we do not have to worry about the source and use of data or any other issues related to the use of data.

We do want to ensure that the research is reproducible. To do this we describe all the tools used during the project, as well as steps taken to achieve results.

Lastly, we want the results to be useful for future research. In order to do this, we firstly publish all the source code of this paper on GitHub. Secondly, we provide human-readable explanations of the formalised proofs. This is to ensure anyone can understand the proofs, even if they do not have a background in Homotopy Type Theory or Coq. [9]

7 Conclusions and Future Work

This paper aimed to formalise parts of chapter 3 of the Symmetry book. We formalised the definition of the circle and four proofs about the circle, of which two theorems.

Although we were not able to formalise all proofs in chapter 3 during time constraints, all the proofs that were attempted to formalise were formalised in the end. We can thus conclude that we have successfully answered the research question: *Can we formalise the proofs in chapter 3 of the Symmetry book using the UniMath Coq library to verify their correctness?*. [10]

In future research, more proofs from the Symmetry book should be formalised. Furthermore, as the Symmetry book is a work in progress, the collection of formalisations should grow as it grows.

[7] This paragraph contains suggestions from Copilot.

[8] https://unimath.zulipchat.com/

[9] This paragraph contains suggestions from Copilot.

[10] This paragraph contains suggestions from Copilot.

A Theorems, Lemmas, Constructions and Corollaries used

This appendix contains all the lemmas and constructions used with their source. When the source is UniMath, the word UniMath contains a link to the specific line code on the UniMath GitHub repository containing the proof.

Construction A.1. *Let X, Y be types. For each equivalence $f : X \to Y$, we have a function $g : Y \to X$ such that for all $x : X$ we have $g(f(x)) \xrightarrow{=} x$ and for all $y : Y$ we have $f(g(y)) \xrightarrow{=} y$. Conversely, if we have such a function g, then f is an equivalence.*

Construction A.1 is taken from the Symmetry book.[3, Construction 2.9.9]

Construction A.2. *Let X, Y be types, $f, g : X \to Y$ functions, and let $Z(x) :\equiv (f(x) \xrightarrow{=} g(x))$ for every x. Then for all x, x' in X, $e : x \xrightarrow{=} x$, and $i : f(x) \xrightarrow{=} g(x)$ we have:*

$$tr_Z(e, i) \xrightarrow{=} ap_g(e) \cdot i \cdot ap_f(e)^{-1}$$

Construction A.2 is taken from the Symmetry book. [3, Construction 2.14.3]

Lemma A.3. *Let X, Y be types and $f : X \xrightarrow{\simeq} Y$ an equivalence. It holds that if X is a set, then Y is also a set.*

Lemma A.3 is a special case of `isofhlevelweqf` in UniMath[1, isofhlevelweqf].

Lemma A.4. *Let A be a type, $B : A \to \mathcal{U}$ be a dependent type and $s, s' : \sum_{x:A} B(x)$. Given $p : pr_1(s) \xrightarrow{=} pr_1(s')$ and $q : tr_B(p, pr_2(s)) \xrightarrow{=} pr_2(s')$, it holds that $s \xrightarrow{=} s'$.*

Lemma A.4 is taken from UniMath[1, total2_pathsf].

Lemma A.5. *Let A, B be types and $f : A \to B$ a function. It holds that $\sum_{b:B} f^{-1}(b) \xrightarrow{\simeq} A$.*

Lemma A.5 is taken from UniMath[1, sum_of_fibers].

Lemma A.6. *Let A be a type, $B : A \to \mathcal{U}$ a dependent type and $s, s' : \sum_{x:A} B(x)$. Given that B is a predicate and $pr_1(s) \xrightarrow{=} pr_1(s')$, it holds that $s \xrightarrow{=} s'$.*

Lemma A.6 is taken from UniMath[1, subtypePath].

References

[1] UniMath, *Unimath coq library*, Accessed: 2023-05-10. [Online]. Available: https://github.com/UniMath/UniMath.

[2] S. Awodey, T. Coquand, and V. Voevodsky, *Homotopy type theory: Univalent foundations of mathematics*. Univalent Foundations Program, Institute for Advanced Study, 2013. [Online]. Available: https://homotopytypetheory.org/book/.

[3] M. Bezem, U. Buchholtz, P. Cagne, B. I. Dundas, and D. R. Grayson, *Symmetry*, https://github.com/UniMath/SymmetryBook, Commit: `cd4f14a`, May 10, 2023.

[4] E. Rijke, *Introduction to Homotopy Type Theory*. 2022. arXiv: 2212.11082 [math.LO]. [Online]. Available: https://arxiv.org/pdf/2212.11082.pdf.

[5] M. H. Escardó *et al.*, *Hottest summer school*, https://github.com/martinescardo/HoTTEST-Summer-School, 2022.

[6] UniMath, *Unimath school*, https://github.com/UniMath/Schools, 2022.

[7] HoTT, *Coq hott*, https://github.com/HoTT/Coq-HoTT, 2023.

[8] Github, *Copilot x*, https://github.com/features/preview/copilot-x, 2023.

[9] OpenAI, *Chatgpt*, https://chat.openai.com, version May 24, 2023.