Formalizing Equivalences Without Tears

Tom de Jong ✉ 🏠 iD
School of Computer Science, University of Nottingham, UK

Abstract

This expository note describes two convenient techniques in the context of homotopy type theory for proving – and formalizing – that a given map is an equivalence. The first technique decomposes the map as a series of basic equivalences, while the second refines this approach using the 3-for-2 property of equivalences. The techniques are illustrated by proving a basic result in synthetic homotopy theory.

2012 ACM Subject Classification Theory of computation → Type theory

Keywords and phrases 3-for-2 property, 2-out-of-3 property, definitional equality, equivalence, formalization of mathematics, synthetic homotopy theory, type theory

Digital Object Identifier 10.4230/LIPIcs.TYPES.2024.1

Funding This work was supported by The Royal Society (grant reference URF\R1\191055). The HIM programme was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2047/1 – 390685813.

Acknowledgements I am grateful to Josh Chen, Stefania Damato, Martín Escardó, Nicolai Kraus, Fredrik Nordvall Forsberg, Stiéphen Pradal and Jon Sterling for comments on this note. I also thank the participants, organizers and support staff of the HIM trimester programme *Prospects of Formal Mathematics* for the opportunity to present the ideas in this note. Finally, I thank the anonymous reviewers for their helpful suggestions.

1 Introduction

A very common problem in *homotopy type theory (HoTT)* [8] is to prove that a given map is an equivalence. The purpose of this short note is to describe convenient techniques for doing this, in particular when one is interested in formalizing the argument in a proof assistant. I claim no originality in the results of this note. Indeed, the technique I wish to highlight already informs much of the AGDA-UNIMATH library developed by Rijke and contributors [5], while I picked up the other (decomposition) technique in this note via the Agda development TypeTopology of Escardó and collaborators [1] as well as Escardó's comprehensive introduction to univalent foundations and its formalization in Agda [2]. Rather, I hope that this note will contribute to a greater awareness of these techniques, especially among junior type theorists.

Outline

The note is outlined as follows. Section 2 explains why directly proving that a map is an equivalence is often cumbersome. Section 3 describes an alternative technique by decomposing the given map into a sequence of (smaller) equivalences, while Section 4 further refines this method using the fact that equivalences satisfy the *3-for-2 property*. These techniques are then illustrated in Section 5 in the context of *synthetic homotopy theory* [8, §8]. The example is self-contained and no prior knowledge of this area is required, although someone familiar with classical homotopy theory may find its simplicity appealing and consider it an invitation to learn more (see e.g. [7]).

30th International Conference on Types for Proofs and Programs (TYPES 2024).
Editors: Rasmus Ejlers Møgelberg and Benno van den Berg; Article No. 1; pp. 1:1–1:6
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Terminology

The 3-for-2 property of equivalences states that if any two maps in a commutative triangle are equivalences, then so is the third. This property is perhaps more commonly known as the *2-out-of-3 property*. André Joyal proposed the name 3-for-2 by analogy to how discounts are often advertised; if you prove two maps are equivalences, then the third is for free [4]. Since we are interested in reducing the amount of (formalization) work and wish to get as much as possible for free, the analogy is quite apt and we prefer Joyal's terminology in this note.

Foundations

As mentioned at the very start, this note is concerned with equivalences in homotopy type theory, although the issues and techniques described should carry over to other intensional type theories [3] and classes of maps that satisfy 3-for-2.[1] We mostly adopt the terminology and notation of the HoTT Book [8], e.g. writing $\equiv$ for judgemental (definitional) equality, using $=$ for identity types (sometimes known as propositional equality), and $\sim$ for homotopies (i.e. pointwise identities).

2 A naive approach

Presented with the problem of showing that a map $f : A \to B$ is an equivalence,[2] a direct approach would be to try and construct a map $g : B \to A$ together with identifications $g \circ f \sim \mathrm{id}_A$ and $f \circ g \sim \mathrm{id}_B$. What makes this approach infeasible at times is that the construction of g may be involved, resulting in nontrivial computations (often involving *transport*) when showing that the round trips are homotopic to the identity maps, especially in proof-relevant settings such as HoTT.

In some cases we are lucky and the desired identifications hold definitionally, in which case the proof assistant can simply do the work for us by unfolding definitions. We return to using definitional equalities to our advantage in Section 4.

3 Decomposition into equivalences

Instead of directly arguing that a given map $f : A \to B$ is an equivalence, it is often convenient to instead decompose f as a series of "building block equivalences", general maps that we already know to be equivalences, as depicted below.

$$\begin{tikzcd}
A \arrow[rrrrr, "f"] \arrow[dr, bend right, "\simeq"'] & & & & & B \\
& X_1 \arrow[r, "\simeq"] & X_2 \arrow[r, "\simeq"] & \dots \arrow[r, "\simeq"] & X_n \arrow[ur, bend right, "\simeq"'] &
\end{tikzcd} \tag{1}$$

Some quintessential examples of such building block equivalences are as follows.

[1] However, I don't have a good illustration to hand of applying the techniques to a class of maps other than the equivalences. The class of n-equivalences, i.e. those maps whose n-truncation is an equivalence, comes to mind, but there I have found it easier to work with the fact that these maps can be characterized as those maps for which precomposition into n-types is an equivalence [6, Lem. 2.9] (see also [5, `k-equivalences`]). Moreover, while equivalences are invertible, maps satisfying 3-for-2 need not be of course, which means that it may be more difficult to arrange a diagram like in (2).

[2] The precise definition of an equivalence is somewhat subtle, as discussed at length in [8, §4], but it is not too important for our purposes.

Projection from the total space of a contractible family. Given a type X and a dependent type Y over it, if each $Y(x)$ is contractible (i.e. it is equivalent to the unit type), then the projection map $\mathrm{pr}_1 : \sum(x : X)\, Y(x) \to X$ is an equivalence. In fact, this is an equivalence if and only if each $Y(x)$ is contractible.

Contractibility of singletons. For any type X and $x : X$, the type $\sum(y : X)\, x = y$ is contractible and hence the two projection maps from $\sum(x : X) \sum(y : X)\, x = y$ to X are equivalences.

Associativity of dependent sums. For a type A, and dependent types $B(a)$ and $C(a, b)$ over A and $\sum(a : A)\, B(a)$, respectively, the map

$$\begin{aligned} \sum(a : A) \sum(b : B(a))\, C(a, b) &\to \sum\Big(p : \sum(a : A)\, B(a)\Big)\, C(p) \\ (a, b, c) &\mapsto ((a, b), c) \end{aligned}$$

is an equivalence.

Reindexing dependent sums along an equivalence. Given an equivalence between types $f : A \simeq B$ and a dependent type $Y(b)$ over B, the assignment

$$\begin{aligned} \sum(a : A)\, Y(f(a)) &\to \sum(b : B)\, Y(b) \\ (a, y) &\mapsto (f(a), y) \end{aligned}$$

is an equivalence. And of course there is a similar result for dependent products.

Distributivity of $\prod$ over $\sum$. Given a type A and dependent types B and Y over A and $\sum(a : A)\, B(a)$, respectively, the map

$$\begin{aligned} \prod(a : A) \sum(b : B(a))\, Y(a, b) &\to \sum\Big(f : \prod(a : A)\, B(a)\Big) \prod(a : A)\, Y(a, f(a)) \\ \alpha &\mapsto (\lambda a.\, \mathrm{pr}_1(\alpha(a)), \lambda a.\, \mathrm{pr}_2(\alpha(a))) \end{aligned}$$

is an equivalence with inverse $(f, p) \mapsto \lambda a.\, (f(a), p(a))$.

This, and especially its nondependent version

$$\prod(a : A) \sum(b : B)\, Y(a, b) \to \sum(f : A \to B) \prod(a : A)\, Y(a, f(a)),$$

is traditionally called the "type theoretic axiom of choice", but this is a misnomer as there is no choice involved, see also [8, pp. 32 and 104].

Distributivity of $\sum$ over $+$. Given a type A and dependent types X and Y over it, the map

$$\begin{aligned} \sum(a : A)\, (X(a) + Y(a)) &\to \sum(a : A)\, X(a) + \sum(a : A)\, Y(a) \\ (a, \mathrm{inl}\, x) &\mapsto \mathrm{inl}(a, x) \\ (a, \mathrm{inr}\, y) &\mapsto \mathrm{inr}(a, y) \end{aligned}$$

is an equivalence.

Congruence of type formers. All type formers respect equivalences. For example, if we have $f : A \simeq X$ and $g : B \simeq Y$, then $(A + B) \simeq (X + Y)$ by applying f to the elements on the left and g to the elements on the right.

Composition with a fixed path. Given elements x, y and z of a type X and a path $p_0 : x = y$, the path composition maps

$$\begin{aligned} (z = x) &\to (z = y) \\ p &\mapsto p \cdot p_0 \end{aligned} \qquad \text{and} \qquad \begin{aligned} (x = z) &\to (y = z) \\ p &\mapsto p_0 \cdot p \end{aligned}$$

are equivalences.

This list is not exhaustive, but should give a good impression of the available building blocks. The interested reader may find many more examples in [1, `UF.EquivalenceExamples`].

The idea is to reduce the task of proving that f is an equivalence to identifying suitable building blocks – the equivalences in (1) – *and proving that the diagram* (1) *commutes.* It is the latter point that may pose similar difficulties to those explained in the previous section: the commutativity proof could involve nontrivial computations. We turn to a refinement in the next section to address this.

We should mention that this technique is still very valuable, especially when we are not interested in having a particular equivalence, or when there is a unique such equivalence, e.g. when proving that a type X is contractible.

4 A refinement using 3-for-2

To address the issue identified above, we will make use of the fact that equivalences satisfy the 3-for-2 property:

▶ **Lemma 1** (3-for-2 for equivalences, [8, Thm. 4.7.1])**.** *In a commutative triangle*

$$\begin{tikzcd}
A \arrow[rr, "h"] \arrow[dr, "f"'] & & C \\
& B \arrow[ur, "g"'] &
\end{tikzcd}$$

if two of the maps are equivalences, then so is the third.

Rather than decomposing f, the idea is to simply involve f into *any* commutative diagram where all (other) maps are equivalences, as depicted below.

$$\begin{tikzcd}
X_1 \arrow[r, "\simeq"] \arrow[rrrrrrrrr, bend right=15, "\simeq"'] & X_2 \arrow[r, "\simeq"] & \ldots \arrow[r, "\simeq"] & X_i \arrow[r, "\simeq"] & A \arrow[r, "f"] & B \arrow[r, "\simeq"] & X_{i+1} \arrow[r, "\simeq"] & \ldots \arrow[r, "\simeq"] & & X_n
\end{tikzcd} \tag{2}$$

By the 3-for-2 property, we can still conclude that f is an equivalence from (2), but verifying the commutativity may now be easier, especially when the type X_n is simpler than B. Indeed, we can often ensure this in practice as illustrated and commented on at the end of the next section.

5 An example in synthetic homotopy theory

Usually [8, §7.5], a type is said to be *n-connected* if its n-truncation is contractible. For us, the following characterization may serve as a definition.

▶ **Proposition 2** ([8, Cor. 7.5.9])**.** *A type A is n-connected if and only if for every n-type*[3] *B, the constants map*

$$\begin{aligned} B &\to (A \to B) \\ b &\mapsto \lambda a.\, b \end{aligned}$$

is an equivalence.

[3] We recall from [8, §7.1] that the n-types are inductively defined for $n \geq -2$: a -2-type is a contractible type and an $(n+1)$-type is a type whose identity types are n-types.

We will illustrate the above techniques by giving a slick proof of a well-known result (see e.g. [8, Thm. 8.2.1]) in homotopy theory: taking the suspension of a type increases its connectedness by one. The reader may wish to compare the proof below to that given in *op. cit*, or inspects its formalization as part of the AGDA-UNIMATH library [5, `Suspensions increase connectedness`].

For completeness, we recall suspensions in homotopy type theory.

▶ **Definition 3** (Suspension Σ). *The* suspension $\Sigma\, A$ *of a type* A *is the pushout of the span* $\mathbf{1} \leftarrow A \rightarrow \mathbf{1}$. *Equivalently, it is the higher inductive type generated by two point constructors* $\mathrm{N}, \mathrm{S} : \Sigma\, A$ *(short for* North *and* South*) and a path constructor* $\mathrm{merid} : A \to \mathrm{N} = \mathrm{S}$ *(short for* meridian*).*

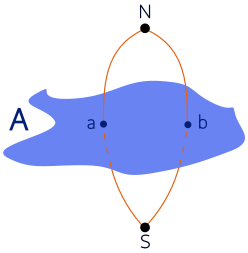

Figure 1 Illustration of the suspension of a type A. The lines from N to S going through the points $a : A$ and $b : A$ represent the paths $\mathrm{merid}(a)$ and $\mathrm{merid}(b)$, respectively.

We shall only need the following basic fact about suspensions which is just the universal property of the suspension as a pushout. (We recall from [8, §2.2] that ap_g denotes the action of g on the identity types.)

▶ **Proposition 4** (Universal property of the suspension, [8, Exer. 6.11]).
The map

$$\begin{aligned} (\Sigma\, A \to B) \quad &\to \quad \sum (b_N : B) \sum (b_S : B)\, (A \to b_N = b_S) \\ g \quad &\mapsto \quad \big(g(\mathrm{N}), g(\mathrm{S}), \lambda a.\, \mathrm{ap}_g(\mathrm{merid}(a))\big) \end{aligned}$$

is an equivalence for all types A *and* B*.*

▶ **Theorem 5.** *The suspension of an* n*-connected type is* $(n+1)$*-connected.*

Proof. Let A be an n-connected type and B an $(n+1)$-type. By Proposition 2, we need to show that the constants map $\mathrm{consts} : B \to (\Sigma\, A \to B)$ is an equivalence. We first consider the evaluation map

$$\mathrm{eval} : (\Sigma\, A \to B) \to B \quad \text{defined by} \quad \mathrm{eval}(g) :\equiv g(\mathrm{N}),$$

and note that the diagram

$$\begin{tikzcd} B \arrow[r, "\mathrm{consts}"] \arrow[rr, bend right, "\mathrm{id}_B"'] & (\Sigma\, A \to B) \arrow[r, "\mathrm{eval}"] & B \end{tikzcd}$$

commutes definitionally. Hence, by **3-for-2**, it suffices to prove that eval is an equivalence.

We use the **decomposition** technique to do so and consider the following diagram.

$$\begin{tikzcd}
(\Sigma\, A \to B) \arrow[rrr, "\mathrm{eval}"] \arrow[dr, "g \mapsto (g(N), g(S), \dots)", "\simeq"'] & & & B \\
& \sum(b_N : B) \sum(b_S : B)\,(A \to b_N = b_S) \arrow[r, bend right, "\simeq", "(b_N, b_S, \varphi) \mapsto (b_N, b_S, \dots)"'] & \sum(b_N : B) \sum(b_S : B)\,(b_N = b_S) \arrow[ur, "(b_N, b_S, \dots) \mapsto b_N", "\simeq"'] &
\end{tikzcd} \tag{3}$$

Note that because A is n-connected and $b_N = b_S$ is an n-type (since B is an $(n+1)$-type), the constants map $(b_N = b_S) \to (A \to b_N = b_S)$ is an equivalence. The middle equivalence in the diagram (3) is induced by the inverse of this map. It should be stressed that the definitional behaviour of this inverse is completely irrelevant for verifying the commutativity of the diagram (3); we only need to know the middle map's behaviour on the first two components of the Σ-types to see that the diagram commutes definitionally.

Finally, the first map in the decomposition of eval is an equivalence by Proposition 4 and the last map is an equivalence by **contractibility of singletons** (from page 3). ◀

We end this note by pointing out an important **heuristic** that is nicely illustrated by the above proof: *We always try to orient the maps in our diagrams towards the simplest possible type* – which in (3) is clearly B. The reason for this is that checking commutativity should then be the easiest as there is relatively little data in the target type to compare. Similarly, we defined a section of eval, rather than directly defining a map $B \to (\Sigma\, A \to B)$ and proving it's an inverse to eval which would involve the cumbersome task of comparing functions $\Sigma\, A \to B$ for equality.

References

1 Martín H. Escardó and contributors. TypeTopology. `http://www.cs.bham.ac.uk/~mhe/TypeTopology/index.html`. Agda development. URL: `https://github.com/martinescardo/TypeTopology`.

2 Martín Hötzel Escardó. Introduction to univalent foundations of mathematics with Agda, 2019. `arXiv:1911.00580`.

3 Martin Hofmann. *Extensional concepts in intensional type theory*. PhD thesis, University of Edinburgh, 1995. Published in Springer's *Distinguished Dissertations* series in 1997. `doi:10.1007/978-1-4471-0963-1`.

4 André Joyal. Categorical aspects of type theory, 2013. Talk in the *Progress in Higher Categories* session at the *Canadian Mathematical Society (CMS) Summer Meeting* in Halifax. URL: `https://home.sandiego.edu/~shulman/cmshighercategories2013/Joyal.pdf`.

5 Egbert Rijke, Elisabeth Bonnevier, Jonathan Prieto-Cubides, Fredrik Bakke, and others. The agda-unimath library. URL: `https://unimath.github.io/agda-unimath/`.

6 Egbert Rijke, J. Daniel Christensen, Luis Scoccola, and Morgan Opie. Localization in homotopy type theory. *Higher Structures*, 4(1):1–32, 2020. `doi:10.21136/hs.2020.01`.

7 Michael Shulman. Homotopy type theory: The logic of space. In Mathieu Anel and Gabriel Catren, editors, *New Spaces in Mathematics*, pages 322–404. Cambridge University Press, 2021. `doi:10.1017/9781108854429.009`.

8 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.