Formalizing selected topics from synthetic homotopy theory and category theory in Homotopy Type Theory

Final report

Vojtěch Štěpančík

2024-05-19

Abstract

Homotopy type theory [Uni13] is a recently developed logical framework built on Martin-Löf type theory [MS84], a hierarchy of universes, and the univalence axiom, which characterizes identity types of universes. It is a constructive type theory inspired by the homotopical interpretation — to a first approximation, types in the theory are interpreted as homotopy types, type families are interpreted as fibrations, and identity types are interpreted as path spaces.

This project explores formalization of synthetic homotopy theory, specifically descent for coequalizers, the flattening lemma for coequalizers, and the induction principle of path spaces of coequalizers, following Kraus and von Raumer [KR21]. The formalization is carried out using the Agda proof assistant [Agd].

Introduction

The framework of Homotopy Type Theory allows one to do homotopy theory internally to the logic — take types to be the objects of study, and use syntactic methods for reasoning about types to study homotopical properties such as truncatedness, connectedness, characterization of path spaces of various limits and colimits, descent properties, etc. This line of research is dubbed "synthetic homotopy theory".

In Section 1, we study some homotopical properties of homotopy coequalizers, which form a particular class of homotopy colimits. Then in Section 2 we reflect on the process of transfering these theorems between similar concepts, such as pointed coequalizers, and how the study and formalization of $(\infty,1)$-categories might help to make this transfer less manual.

We attempt to emphasize intuition behind constructions from homotopy type theory, rather than introducing and explaining specific notations, but we redirect the interested reader to an explicitly introductory text, such as Rijke's Introduction to Homotopy Type Theory [Rij22], or the HoTT Book [Uni13].

1 Synthetic homotopy theory

When talking about (co)limits in homotopy type theory, we always mean *homotopy* (co)limits. Formally, these correspond to taking the (co)limits involving (co)fibrant replacements in the model. We may think about it as replacing the required equality "on the nose" between two elements by requiring a *path* between the elements. For example, an equalizer of a pair $f, g : A \to B$ consists of the type of tuples (a, p) such that $a \in A$ and p is a path from $f(a)$ to $g(a)$ in B, while their coequalizer is the type containing a copy of B, and for every $a \in A$ a new path from $f(a)$ to $f(b)$.

Similarly when talking about commutativity, we always mean commutativity *up to homotopy*. For example a "commuting square"

$$\begin{array}{ccc} A & \xrightarrow{h} & X \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[k]{} & Y \end{array}$$

stands for an explicit choice of a homotopy $H : k \circ f \sim g \circ h$, which witnesses that for all elements $a : A$, there exists a path $H(a)$ from $k(f(a))$ to $g(h(a))$ in Y. This complicates things in higher dimensions — more complex shapes, such as commuting cubes or prisms, no longer consist only of proofs of commutativity of their faces, but since those faces are data, we need to ensure *coherence* between this data. We don't elaborate further what this means precisely, but intuitively it corresponds to proving that a "filler" of the higher shape exists for a specific choice of faces.

In proper categorical parlance, a homotopy coequalizer is a homotopy colimit of the diagram consisting of two parallel arrows, in other words, it is the homotopy colimiting cocone of shape

$$A \underset{f}{\overset{g}{\rightrightarrows}} B \xrightarrow{e} C.$$

The descent property and flattening have a concise categorical description in terms of pullbacks: consider a diagram

$$\begin{array}{ccccc} A' & \underset{f'}{\overset{g'}{\rightrightarrows}} & B' & \xrightarrow{e'} & C' \\ {\scriptstyle h_A}\downarrow & & \downarrow{\scriptstyle h_B} & & \downarrow{\scriptstyle h_C} \\ A & \underset{f}{\overset{g}{\rightrightarrows}} & B & \xrightarrow[e]{} & C, \end{array}$$

where the diagrams

$$\begin{array}{ccc} A' & \xrightarrow{f'} & B' \\ {\scriptstyle h_A}\downarrow & \lrcorner & \downarrow{\scriptstyle h_B} \\ A & \xrightarrow[f]{} & B \end{array} \quad , \quad \begin{array}{ccc} A' & \xrightarrow{g'} & B' \\ {\scriptstyle h_A}\downarrow & \lrcorner & \downarrow{\scriptstyle h_B} \\ A & \xrightarrow[g]{} & B \end{array} \quad , \quad \begin{array}{ccc} B' & \xrightarrow{e'} & C' \\ {\scriptstyle h_B}\downarrow & & \downarrow{\scriptstyle h_C} \\ B & \xrightarrow[e]{} & C \end{array} \quad , \quad \begin{array}{ccc} A' & \underset{f'}{\overset{g'}{\rightrightarrows}} & B' \\ & & \\ A & \underset{f}{\overset{g}{\rightrightarrows}} & B \end{array} \quad \begin{array}{c} \xrightarrow{e'} C' \\ \\ \xrightarrow{e'} C \end{array}$$

commute, the two left squares are pullback squares, and the commuting faces compose to a coherent shape. The **descent property** states that if the top half of the diagram is a coequalizer, then the right square is a pullback. Conversely, the **flattening lemma** states that if the right square is a pullback, then the top diagram is a coequalizer.

We break down these statements to a more type-theoretical language, which is easier to prove, and reveals perhaps a more elementary intuition.

First, we use the fact that thanks to the presence of universes in the theory, we may pass from the "displayed" version of fibrations over a type A — that is, a type B equipped with a map $f : B \to A$ — to the "indexed" version of fibrations over a type A — maps of type $P : A \to \mathcal{U}$. To go from the former to the latter, we assign to each element $a : A$ the type of the *homotopy fiber* of f over a, that is the type $\mathrm{fib}_f a := \Sigma(b : B).fb = a$. To go backwards, we collect all the types $P(-)$ into the total space $\Sigma(a : A).Pa$, which is equipped with the first projection onto A. This translation is known as *type duality*, and it manifests the universe $\mathcal{U}$ as the object classifier.

Applying type duality to pullback squares, we get that asking for a square

$$\begin{array}{ccc} A & \xrightarrow{h} & X \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[k]{} & Y \end{array}$$

to be a pullback is equivalent to asking for the square

$$\begin{array}{ccc} \Sigma(b : B).\mathrm{fib}_f b & \xrightarrow{\mathrm{tot}_k h} & \Sigma(y : Y).\mathrm{fib}_g y \\ {\scriptstyle \mathrm{pr}_1}\downarrow & & \downarrow{\scriptstyle \mathrm{pr}_1} \\ B & \xrightarrow[k]{} & Y \end{array}$$

to be a pullback, which turns out to be equivalent to asking for the induced map on fibers $h : (b : B) \to \mathrm{fib}_f b \to \mathrm{fib}_g(kb)$ to be a fiberwise equivalence [Rij19].

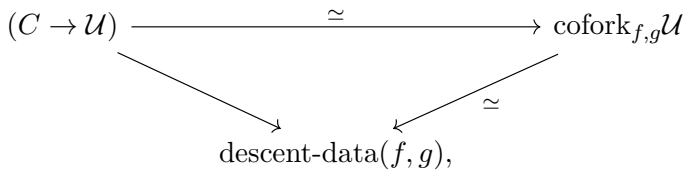
This is where my material contributions to formalization of synthetic homotopy theory start. They are collected in a series of pull request [Ště24a; Ště24b] to the agda-unimath library.

Packaging the two pullback squares from the input data gives rise to the definition of *descent data for coequalizers*. A descent datum for a parallel pair $f, g : A \to B$ consists of

- a type family $P_B : B \to \mathcal{U}$, and
- a family of equivalences $(a : A) \to P_B(fa) \simeq P_B(ga)$.

Given a cofork $e : B \to C, H : e \circ f \sim e \circ g$ under f, g, a family $P : C \to \mathcal{U}$ induces the descent data $(P \circ e, \lambda a \to \mathrm{tr}_P(Ha))$. The descent property states that if e is a coequalizer of f, g, then this assignment is an equivalence, i.e. we have $(C \to \mathcal{U}) \simeq \text{descent-data}(f, g)$.

This type-theoretical reading tells us that descent data characterize type families over coequalizers. The claim is straightforward to prove: we observe that there is a commuting triangle of maps



where the top map is an equivalence by the universal property of coequalizers, and the right map is given by functoriality of Σ and the univalence axiom, which is required to pass from $P(fa) = P(ga)$ to $P(fa) \simeq P(ga)$. It follows that the left map is an equivalence.

As a corollary, we get that every descent datum has a unique "corresponding" type family — the correspondence meaning that the assigned family induces descent data which is equivalent to the original datum. Morphisms and equivalences of descent data are also contained in my formalization. We denote a pair consisting of descent data and its corresponding family as $P \approx (P_B, P_A)$. One of the takeaways of this formalization is that it is convenient to parameterize definitions and theorems over this pair — traditionally, one would only work with a family or only with descent data, and pass to the other ad-hoc via unique existence of the other. When the definitions are parameterized, the user may provide their own convenient equivalence, with perhaps nice computational properties.

Further, I built the infrastructure for sections of descent data, and proved that for a pair $P \approx (P_B, P_A)$, the type of sections $(c : C) \to Pc$ is equivalent to the type of sections of (P_B, P_A).

This infrastructure already allows us to characterize some families over coequalizers. I proved and formalized that when given two families of descent data $P \approx (P_B, P_A)$ and $Q \approx (Q_B, Q_A)$, then the family of function types $\lambda c \to Pc \to Qc$ is equivalent to the descent data whose sections are morphisms of the associated descent data, and the type of families of equivalences $\lambda c \to Pc \simeq Qc$ is equivalent to the type whose sections are equivalences of the associated descent data.

I had already formalized a version of the flattening lemma before taking on this project [Ště23], but that development was less general, as it didn't integrate with the infrastructure of descent data. My new contribution is extending it to the more generic setting, for which I developed new infrastructure around equivalences of coforks under equivalences of pairs of parallel arrows, as mentioned in the progress report. The proof first shows that for $\alpha : P \approx (P_B, P_A)$ over $f, g : A \rightrightarrows B \to C : e$ there is an equivalence of double arrows

$$\begin{array}{ccccc}
\Sigma(b:B).P_B b & \xleftarrow{\mathrm{tot}_f \mathrm{id}} & \Sigma(a:A).P_B(fa) & \xrightarrow{\mathrm{tot}_g P_A} & \Sigma(b:B).P_B b \\
\downarrow{\scriptstyle \mathrm{tot}\alpha} & & \downarrow{\scriptstyle \mathrm{tot}(\alpha \circ f)} & & \downarrow{\scriptstyle \mathrm{tot}\alpha} \\
\Sigma(b:B).P(eb) & \xleftarrow[\mathrm{tot}_f \mathrm{id}]{} & \Sigma(a:A).P(e(fa)) & \xrightarrow[\mathrm{tot}_g\ \mathrm{tr}_P(Ha)]{} & \Sigma(b:B).P(eb),
\end{array}$$

and furthermore there is an equivalence of coforks under it

$$\begin{array}{ccccc}
\Sigma(a:A).P_B(fa) & \rightrightarrows & \Sigma(b:B).P_B b & \longrightarrow & \Sigma(c:C).Pc \\
\downarrow & & \downarrow & & \downarrow{\scriptstyle \mathrm{id}} \\
\Sigma(a:A).P(e(fa)) & \rightrightarrows & \Sigma(b:B).P(eb) & \longrightarrow & \Sigma(c:C).Pc.
\end{array}$$

It follows that the top cofork is a coequalizer if and only if the bottom one is. Since the bottom one is a coequalizer by the less general flattening lemma, we have that given a pair $P \approx (P_B, P_A)$, then the total space $\Sigma(c:C).Pc$ is the coequalizer of the total spaces $\Sigma(a:A).P_B(fa) \rightrightarrows \Sigma(b:B).P_B b$.

The pinnacle of my formalization effort is stating the induction principle of based identity descent data of coequalizers, and showing that the descent data induced by the based identity type satisfies this induction principle. As a corollary, it follows that any descent data satisfying the principle at an element $b_0 : B$ is equivalent to the identity descent data, and in particular there is an equivalence $(b:B) \to (e(b_0) = e(b)) \simeq P_B b$.

In homotopy type theory, colimits/objects with a "mapping out" property usually have multiple equivalent statements of universality. We first define an appropriate notion of cocones (e.g. a cofork for coequalizers, a type for initiality, etc.) and dependent/fibered cocones, with natural evaluation maps $\mathrm{ev} : (X \to Y) \to \mathrm{cocone}(Y)$ and $\mathrm{evd} : ((x:X) \to Px) \to \text{dep-cocone}(P)$. The usual statements are

- the universal property: For every other object Y, the evaluation map is an equivalence
- the dependent universal property: For every fibered object $P : X \to \mathcal{U}$, the dependent evaluation map is an equivalence
- the induction principle: For every fibered object $P : X \to \mathcal{U}$, the dependent evaluation map has a section — a map $\mathrm{ind}_P : \text{dep-cocone}(P) \to (x:X) \to Px$ such that $\mathrm{evd} \circ \mathrm{ind}_P \sim \mathrm{id}$
- the recursion principle and uniqueness, also known as initiality: For every object Y, the evaluation map has a *unique* section — a map $\mathrm{rec}_Y : \mathrm{cocone}(Y) \to X \to Y$ such that $\mathrm{ev} \circ \mathrm{rec}_Y \sim id$

Note in particular that the recursion principle additionally needs uniqueness of the section to be equivalent to the properties. This has been established by Sojakova in [Soj15]. None of the statements is universally superior to the others, and logical equivalences among them, while quite mechanical, are not trivial.

In [KR21], Kraus and von Raumer state the induction principle, and then show that it is satisfied by the canonical descent data for identity types by first showing it satisfies initiality. An analogous existing (unfinished) development for pushouts in agda-unimath states the universal property, and shows directly that the canonical descent data satisfies it. I chose to state the induction principle, since the plan is to eventually describe and formalize an analogue of Wärn's zigzag construction [Wär23], which shows the induction principle for an explicitly constructed type. The construction is out of scope for this project. I then show that the canonical descent data satisfies the universal property, and derive that it satisfies the induction principle.

My development avoids repetitive work of defining and building infrastructure for dependent descent data, instead passing by uncurrying from the type of families $(b : B) \to e(b_0) = e(b) \to \mathcal{U}$ to the type of families $(\Sigma(b : B).e(b_0) = e(b)) \to \mathcal{U}$. The flattening lemma is then used to show that the cofork on $\Sigma(b : B).e(b_0) = e(b)$ is a coequalizer, thus satisfying the descent property and allowing us to pass between families and descent data on the cofork of total spaces. The proofs themselves are quite technical.

I then show that every descent data satisfying the induction principle is equivalent to the canonical one, sending refl_{b_0} to the basepoint. This result may be improved in future work — it is analogous to a recursion principle (the type of basepoint preserving equivalences is contractible) without the uniqueness (the type is contractible). However, for foreseeable future needs, this result is sufficient.

2 (∞,1)-category theory

The downside of building the theorems for types is that it scales poorly. Often times we write theorems about types which generalize to pointed types (or to other types with structure), but this generalization requires rewriting all of the existing infrastructure to talk about the corresponding notions for pointed types (e.g. pointed types, pointed maps, pointed homotopies, pointed equivalences). Similarly, Kraus and von Raumer in [KR21] and later Wärn in [Wär24] explicitly use categorical language, but we haven't internalized it in the development, only in the meta-theory to motivate the constructions. One might hope that the various types with structure form a category, and then use categorical methods to write the theorems in full generality. Unfortunately that is not a case.

A category in homotopy type theory is defined as a type O : Type of objects, a family of *sets* of morphisms $x, y : O \vdash A(x, y) :$ Set, and a choice of identity morphisms and a composition operation, subject to further equalities. The important detail is that the morphism families are sets in the sense of homotopy level — for two elements $f, g : A(x, y)$, the type of identifications $f = g$ is a proposition, or in other words two morphisms are identified in at most one way This development is in accord with the classical development of category theory. However consider the type $\mathbb{S}^1$ and its identity map id. There are $\mathbb{Z}$ many ways of identifying id with itself, each z corresponding to winding along the circle z times. This shows that $A(\mathbb{S}^1, \mathbb{S}^1)$ is not a set, hence the collection of types does not form a category.

It appears we need a higher notion of categories, and the immediate generalization is called "wild categories". A wild category is defined similarly to a category, except $A(x, y)$ can be any type. For many practical purposes wild categories suffice, but the structure is inherently incoherent — the unit and associativity laws are identifications of morphisms, and since morphisms don't necessarily form sets anymore, they become choices of laws, i.e. structure, instead of properties. As such, higher diagrams involving the laws need not commute, for example the pentagon identity familiar from monoidal categories does not in general hold in an arbitrary wild category.

Equipping a wild category with all higher coherences amounts to axiomatizing and formalizing (∞,1) categories. Formalizing them in a usable and ergonomic way is still an open problem. The formalization for agda-unimath is an ongoing research project with Fredrik Bakke, and it's based on a notion of coinductively defined hierarchy of globular sets with structure. The project has been deprioritized in favor of other developments, but I contributed the definition of isomorphisms in noncoherent (∞,∞)-precategories.

A noncoherent (∞,∞)-precategory is a reflexive and transitive globular type. The reflexivity condition provides a choice of identity cells, and transitivity provides a composition operation. This is the minimal structure on which we can coinductively define isomorphisms as biinvertible maps, where the witnesses of invertibility are higher cells, themselves required to be isomorphisms. However because of the absence of coherences, we cannot show that isomorphisms are closed under composition, nor that identity arrows are isomorphisms, because unit laws are part of the coherence structure.

The plan is to eventually proceed with using isomorphisms to define noncoherent (∞,n)-precategories, for which all cells above level n are isomorphisms, and k-coherent (∞,n)-precategories, which have coherences expressed up to higher isomorphisms. The goal of this joint work, which is out of scope for the semestral project, is to define in general k-coherent (n,r)-precategories, of which (∞,1)-precategories of various coherence levels are of particular interest, because they generalize the structure of types, maps, homotopies and higher homotopies. My expectation is that general theorems, such as descent and flattening of various colimits, will then be formulated with the lowest required level of coherence necessary.

References

[Agd] Agda Developers. *Agda*. Version 2.6.4. URL: `https://agda.readthedocs.io/`.

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

[MS84] P. Martin-Löf and G. Sambin. *Intuitionistic Type Theory*. Lecture notes. Bibliopolis, 1984. ISBN: 9788870881059.

[Rij19] Egbert Rijke. *Classifying Types*. 2019. arXiv: `1906.09435 [math.LO]`.

[Rij22] Egbert Rijke. *Introduction to Homotopy Type Theory*. 2022. arXiv: 2212.11082 [math.LO].

[Soj15] Kristina Sojakova. "Higher Inductive Types as Homotopy-Initial Algebras". In: *SIGPLAN Not.* 50.1 (Jan. 2015), pp. 31–42. ISSN: 0362-1340. DOI: 10.1145/2775051.2676983. URL: https://doi.org/10.1145/2775051.2676983.

[Ště23] Vojtěch Štěpančík. *Coforks, coequalizers of types, their flattening lemma*. 2023. URL: https://github.com/UniMath/agda-unimath/pull/792.

[Ště24a] Vojtěch Štěpančík. *Descent and induction principle of identity types of coequalizers*. 2024. URL: https://github.com/UniMath/agda-unimath/pull/1140.

[Ště24b] Vojtěch Štěpančík. *Rebase infrastructure for coequalizers to double arrows*. 2024. URL: https://github.com/UniMath/agda-unimath/pull/1098.

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book, 2013.

[Wär23] David Wärn. *Path Spaces of Pushouts*. Apr. 2023. URL: https://dwarn.se/po-paths.pdf.

[Wär24] David Wärn. *Path Spaces of Pushouts*. Feb. 19, 2024. arXiv: 2402.12339 [math]. (Visited on 02/26/2024). preprint.