

Opetopic types and higher algebra in Homotopy Type Theory

Internship Report

Harington Elies

Under the supervision of Eric Finster

August 19, 2022

Contents

Introduction	4
Notations and conventions	4
1 Context	4
2 Opetopes and opetopic types	6
2.1 What are opetopes ?	6
2.2 Opetopic types and frames	8
2.3 The monad structure on Src	12
3 Higher categories	14
3.1 Definitions	15
3.2 First consequences of the definition	16
3.3 Truncatedness	18
3.4 Equivalence with categories	19
Conclusion	20
References	21

I warmly thank Eric Finster for introducing me to the wonderful subject of opetopes, his enthusiasm working with me and our many discussions together.

Fiche de synthèse

Le contexte général

Depuis l'avènement des fondations univalentes au début des années 2010, la définition d' ∞ -catégorie en théorie homotopique des types reste un problème ouvert. Différentes approches ont été proposées, certaines synthétiques (changeant la théorie elle-même pour que tout type représente une forme d' ∞ -catégorie), d'autres analytiques (cherchant à définir directement les ∞ -catégories dans la théorie). Mon stage s'inscrit dans cette dernière approche (bien qu'étendant en partie la théorie), et plus particulièrement dans la continuité directe de l'approche opératoire initiée par Allou, Finster et Sozeau in [AFS21]. Cette approche a déjà permis de définir les ∞ -groupoïdes et de prouver que ces derniers sont équivalents aux types.

Le problème étudié

Le but de mon stage était d'implémenter en Cubical Agda (avec règles de réécriture) des bases de la théorie des ∞ -catégories à partir des définitions de [AFS21]. Un des objectifs était la définition de slice- ∞ -catégories, une notion fondamentale à la base de nombreuses définitions plus complexes, mais qui s'est finalement révélée irréalisable dans l'état actuel des recherches. Je me suis donc porté sur une autre question : celle de la caractérisation des catégories (au sens usuel du terme) comme un cas particulier des ∞ -catégories.

La contribution proposée

Dans ce rapport, je présente une preuve de l'équivalence entre les catégories (au sens classique du terme), et les ∞ -catégories vérifiant une certaine propriété de troncation que je définis. Lors de mon stage, j'ai implémenté ces définitions et cette preuve dans l'assistant de preuve Cubical Agda.

Ce résultat est analogue à la construction du **nerf** dans la théorie des ensembles simpliciaux, tout comme le résultat de [AFS21] est analogue à celui de "l'hypothèse d'homotopie". Ces deux résultats mis ensemble corroborent donc le bien-fondé de la définition d' ∞ -catégorie donnée dans [AFS21].

Les arguments en faveur de sa validité

Le résultat fondamental de mon stage est lui-même, comme expliqué plus haut, un argument en faveur de la validité de cette définition d' ∞ -catégorie. La preuve de ce résultat est en soi parfaitement valide, puisqu'implémentée dans un assistant de preuve formelle.

Le bilan et les perspectives

Il serait intéressant de généraliser notre résultat au cas des bicatégories, *i.e.* de caractériser les bicatégories comme un cas particulier d' ∞ -catégories. Par ailleurs, l'objectif à long terme est de pouvoir concrètement travailler avec des ∞ -catégories dans des fondations univalentes. Sous cette perspective, les prochaines questions naturelles sont la définition de slice-(ou plus généralement, comma-) ∞ -catégories et d' ∞ -catégories de foncteurs, dans le but de prouver le lemme de Yoneda. Plus tard, une définition de localisation d' ∞ -catégories pourrait permettre d'utiliser en partie la théorie des catégories de modèles pour définir des ∞ -catégories, et ainsi d'utiliser une machinerie actuellement très puissante en théorie des ensembles.

Introduction

Ever since the development of Univalent Foundations [Uni13] of Mathematics in the early 2010's, the definition of higher coherent structures in Homotopy Type Theory has been a major open problem [Buc18]. Synthetic approaches have been proposed, such as *simplicial type theory* [RS17], and on the analytic side, it was known that the problem could be reduced to finding a definition of *semi-simplicial types* inside HoTT [Buc18]. More recent developments used an opetopic approach rather than a semi-simplicial one. Indeed, the authors of [AFS21] gave a definition of ∞ -category and ∞ -groupoid internal to Homotopy Type Theory using a notion of *opetopic type*, and proved that every type carries a unique structure of ∞ -groupoid, and that every ∞ -groupoid arises in this way.

The initial goal of my internship was to implement parts of the proofs of [AFS21] in the proof assistant Cubical Agda, and possibly to find ways to implement some standard higher-categorical constructions in the context of opetopic types. Most of the definitions we planned to tackle turned out to be out of reach for the moment, but I still managed to prove another "sanity-check" result : the fact that $(\infty, 1)$ -categories satisfying a certain truncation condition are equivalent to ordinary categories. This result and its proof are the subject of section 3. Section 1 gives a brief summary of the issues of defining higher structures in HoTT. Section 2 is dedicated to the definitions and ideas required to understand section 3.

Notations and conventions

We use the following notations throughout this report.

- Unit denotes the datatype with one constructor `tt` : Unit.
- \mathcal{U} denotes the universe of types, whose elements are themselves types (everything we state and prove can be done in a fixed universe, so we do not worry about universe-level issues).
- Regarding curryfication : given a map $f : A \rightarrow B \rightarrow C$ and $a : A, b : B$, we write $f(a, b)$ as a shorthand for $f(a)(b)$. Similarly for dependent maps.

Throughout the report, we put links to the agda code in the margins. The code can be found at <https://github.com/ericfinster/opetopic-types>. More precisely, we refer to the commit `d3eefdf8848f0331df1544861b1e76bb3400bb2c` of the repository, and to the subfolder `/Experimental/Local`. For instance, the indication `Category#42` refers to the file `Experimental/Local/Category.agda` on line 42.

1 Context

In set-based mathematics, there are many definitions of ∞ -categories. The most common approach uses simplicial sets. A simplicial set is a presheaf on the category Δ whose objects

are the sets $\{0, \dots, n\}$ for $n \geq 0$ and whose morphisms are the non-decreasing maps. Unfolding this definition, a simplicial set is a family of sets $(X_n)_{n \geq 0}$ together with morphisms between them satisfying certain conditions. The idea is that the elements of X_n should be thought of as *instances of the n -simplex*, and the maps relate how the different instances of simplices embed or collapse onto one another.

When defining ∞ -categories, the instances of the point are thought of as objects, the instances of the segment as morphisms, the instances of the triangle as 2-morphisms, etc. Thus, ∞ -categories are usually defined as simplicial sets satisfying some additional property. However, many different simplicial sets can represent the same ∞ -category. The study of ∞ -categories "up to equivalence" without loss of information is usually done using *model category theory*.

In homotopy type theory on the other hand, the basic objects of the theory (types) are already homotopical in nature, and their equivalence coincides (within the theory) with equality, thanks to the univalence axiom [Uni13]. For instance, in HoTT, for a suitable notion of category (called *univalent categories*), equality coincides with equivalence.

The goal of a definition of ∞ -category in HoTT would be to take advantage of the homotopical nature of the theory to get similar results. However, naively copying the definition of simplicial sets in HoTT wouldn't work : for instance, Whitehead's theorem is true for simplicial sets, but it doesn't hold internally for types in HoTT. We could try to define "simplicial types" as some kind of type-valued presheaf. But the universe of types in HoTT is itself an ∞ -category, so type-valued functors should already contain an infinite hierarchy of coherent data, whose definition was the whole point of trying to define simplicial types in the first place, so this leads to circular reasoning.

At this point there are two ways to approach the issue :

- The synthetic approach, which makes the whole type theory "simplicial" by postulating some primitive simplices (similarly to how cubical type theory postulates an interval type) and builds a *theory of simplicial types* rather than a *definition of simplicial types* **in** type theory [RS17].
- the analytic approach, whose goal is to find a variation on the idea of simplicial type that would actually be definable in type theory, and which would still be sufficient to define ∞ -categories. A possible candidate would be semi-simplicial types, where we keep the face maps but cast aside degeneracies. Then, instead of defining the instances of every possible simplex and then relating them together, one proceeds hierarchically, by first defining the type of points, and then for every pair of points, the type of fillers for the 1-simplex (the directed interval), and then for every triple of points and triple of maps constituting a "frame" for the 2-simplex, a type of fillers for such a frame, etc. see [Buc18]. The point of this approach is that it circumvents the need to specify the "action on maps" in the category Δ as would be done with hypothetical simplicial types, as the relations of inclusion between faces would all hold definitionally. However such a description of semi-simplicial type has never been formalized in HoTT, and it is still

an open problem to know whether this idea is actually implementable without further axioms.

Our approach is also analytic, but instead of using semi-simplicial types, we define *opetopic types*, whose basic shapes are *opetopes* rather than simplices. Our definition of opetopic types is not a definition in plain HoTT though, since it uses additional computation rules (rewrite rules in Agda) to make the definition actually type-check. However, there is no added axiom in the sense that we do not postulate any abstract type or any abstract element in a given type : all our postulates are in the form of rewrite rules, and thus have computational meaning. This is why, although we extend the type theory, we still deem our approach to be analytic and not synthetic : we *define* opetopic types in our extended type theory rather than making a type theory in which "all types are opetopic".

2 Opetopes and opetopic types

Opetopes and opetopic sets were first introduced in [BD97] using the language of operads in order to give a definition of weak n -categories. Alternative definitions of opetopes have since been given, and our notion of opetopic type follows more closely the definition in [Koc+10] in terms of polynomial functors and monads.

2.1 What are opetopes ?

Opetopes are shapes that symbolize many-input single-output pasting diagrams in every dimension. The base case is dimension 0 : there is only one opetope in dimension 0, namely, the point.



An opetope of dimension $n + 1$ is comprised of one $(n + 1)$ -dimensional cell (represented by an $(n + 1)$ -uple arrow) with input a "pasting diagram" of n -dimensional opetopes, and with output a single n -dimensional opetope. What we mean by pasting diagram will be made clearer later, but first we show some examples.

Let's start with dimension 1. The input of a 1-dimensional opetope must be a pasting diagram of 0-opetopes, *i.e.* of points. However, any way of pasting points together always yield a single point, so our 0-dimensional input will always be a single point. There is no choice for the output either, since there is only one 0-dimensional opetope. Hence, there is exactly one 1-dimensional opetope, with a 1-cell joining two points, namely, the arrow.



For readability, we will more often write 1-dimensional arrows with the arrow-head in the middle of the line, but they mean the same thing.



In dimension 2 it gets more interesting : a pasting diagram made out of arrows is a sequence of n arrows for some $n : \mathbb{N}$ (not forgetting the case $n = 0$ where there is zero input arrow and just an input point). The arrows must all "face in the same direction", *i.e.* the input of one arrow cannot be connected to the input of another arrow, and similarly for outputs (the idea is that a pasting diagram in an n -category must have a well-defined composition, and it's not possible to compose opposing-facing cells). Hence a 2-opetope consists of a 2-cell (a double arrow) joining a sequence of some finite number of 1-opetopes (arrows) to a single output arrow. See figure 1 for representations of 2-opetopes.


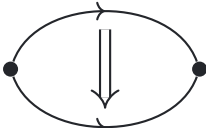
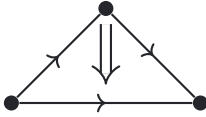
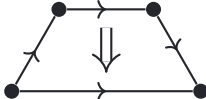
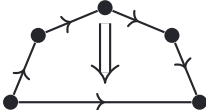

Number of input arrows	0	1	2
Corresponding opetope			
Number of input arrows	3	4	n
Corresponding opetope			

Figure 1: Examples of 2-opetopes. Notice that in the degenerate case $n = 0$, the input and output points of the output arrow are the same.

Remark 2.1. The 2-opetope with no input arrow is called the **2-drop**, and the one with only one input arrow is called the **2-globe**. Their higher analogs will be called the n -drop and the n -globe. Notice how with this definition, the arrow is a 1-globe and the point is a 0-drop.

Now for dimension 3, we need to describe what we mean by "a pasting diagram of 2-opetopes". Just like before, we put 2-opetopes together by pasting outputs onto inputs, so as to keep a global common direction for all 2-cells. Then, given a pasting diagram, the output shape of the corresponding 3-opetope is entirely determined by the "boundary" of the diagram (see figure 2).

The case of dimension 3 sheds some light on the hitherto rather vague notion of "pasting diagram of n -opetopes". Indeed, n -diagrams are built by pasting output faces of n -opetopes onto input faces of other n -opetopes. Thus, a pasting diagram is just a **tree** of n -opetopes, where each n -opetope is a node, and its children are the n -opetopes pasted onto its inputs. More precisely, a given node in a pasting diagram doesn't simply have a list of children, but a *tree* of children, since its children are indexed by its inputs, which themselves form a $(n - 1)$ -diagram, which is also a tree.

Hence, opetopes are closely related to the notion of **n -tree**, a datatype that generalizes

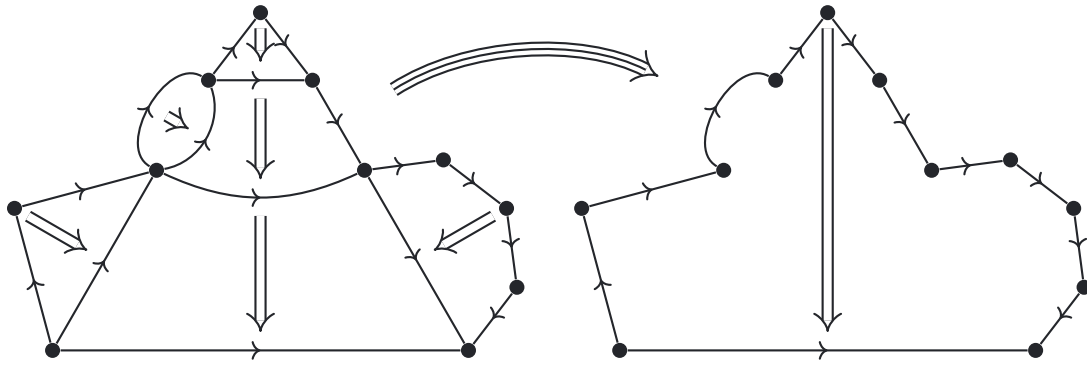


Figure 2: Example of a 3-opetope. Notice how the output 2-opetope on the right is given by the exterior input arrows in the pasting diagram on the left.

trees and lists. The type of n -trees labeled by a type A is defined as an inductive family of types with two constructors :

$$\begin{aligned} \text{leaf} &: \prod_{n:\mathbb{N}} n\text{-tree}(A) \\ \text{node} &: \prod_{n:\mathbb{N}} A \rightarrow n\text{-tree}((n+1)\text{-tree}(A)) \rightarrow (n+1)\text{-tree}(A) \end{aligned}$$

With this definition and for any type A , $0\text{-tree}(A)$ is equivalent to the unit type, $1\text{-tree}(A)$ is equivalent to A , $2\text{-tree}(A)$ is equivalent to the type of lists over A , $3\text{-tree}(A)$ is equivalent to the type of trees over A , etc. We will see that the type of pasting diagrams for opetopes has a very similar definition, with the added subtlety that diagrams will be indexed over the shape of their boundaries, and the reason for that indexing becomes apparent starting in dimension 4.

Indeed, in dimension 4, we need to be careful when pasting 3-opetopes together : for the pasting to make sense, the output 2-opetope of the child 3-opetope must be the same as the input 2-opetope of the parent onto which it is pasted. Hence, when defining opetopes as tree-like structures, we will enforce *through typing* the definitional equality between the boundary of a child diagram and the input of its parent onto which it is pasted.

2.2 Opetopic types and frames

Instead of directly defining opetopes, we first define opetopic types and then recover opetopes as *frames* in a particular opetopic type.

As explained in section 1, an opetopic type X should consist of the data of a type X_o for every opetope o , thought of as "the *instances* (or *labels*) of o in X ", together with a way of relating sub-cells of opetopes (for instance, an instance of a triangle should be related to three instances of arrows : its two input arrows and its one output arrow).

However, when working in HoTT, we don't want these relations between cells to contain homotopical information, hence we want them to hold definitionally. For example, consider an instance of the 2-globe : the fact that the source point of its input is the same as the source point of its output should be a definitional equality, not the data of a propositional one.

The way of accomplishing this is to index instances of n -opetopes by the labels of their boundary. We call the data of a potential labeled boundary in an opetopic type X a **frame** of X (see figure 3). Said differently, an n -frame in X is a labeling for all cells of some opetope in X , except for its highest-dimensional one.

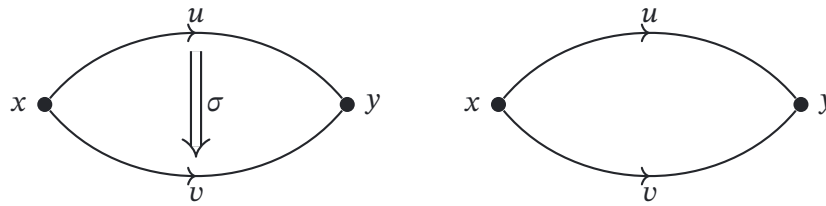


Figure 3: On the left, a labeled 2-globe in some opetopic type X .
On the right, its boundary frame.

We index opetopic types by their dimension : an n -opetopic type will be the data of instances of opetopes up to dimension n . We write $\text{OType}(n)$ for the type of all n -opetopic types. When talking about the frames of some n -opetopic type X , we only refer to the n -dimensional frames, that is, those frames that are missing their top $(n + 1)$ -cell. Given an opetopic type X , we write $\text{Frm}(X)$ for the type of frames of X .

OpetopicType#21

OpetopicType#23

The definition of opetopic types and frames is done by recursion on the dimension. To make the base case as simple as possible, we actually start with $n = -1$, with only one trivial (-1) -opetopic type, and then an opetopic type in dimension $(n + 1)$ is the data of an opetopic type X_n in dimension n together with a type of labels for every frame in X_n . Formally, we give the following definition :

Definition 2.2 (Opetopic types).

$$\begin{aligned} \text{OType}(-1) &\stackrel{\text{def}}{=} \text{Unit} \\ \text{OType}(n + 1) &\stackrel{\text{def}}{=} \sum_{X : \text{OType}(n)} (\text{Frm}(X) \rightarrow \mathcal{U}) \end{aligned}$$

OpetopicType#315

Now we need to formally define frames. There should only be one frame of dimension -1 , whose labels will be the points of the opetopic type. A frame of dimension 0 should consist of an input labeled point and an output labeled point. A frame of dimension 1 should consist of an input sequence of labeled arrows and an output labeled arrow.

More generally, a frame of dimension n should consist of an input diagram of labeled n -opetopes, and an output n -opetope, such that the boundary shape and labels of the input

diagram correspond with the boundary shape and labels of the output opetope. Put differently, an n -frame in $(X_{\leq n-1}, X_n)$ should consist of

- an $(n - 1)$ -frame f ,
- an input diagram s with boundary f (s as in "source"),
- an output label $t : X_n(f)$ (t as in "target").

See figure 4 for a visual representation.

Writing $\text{Src}(X_n, f)$ for the (yet to be defined) type of input diagrams with labels in X_n and boundary f , we thus give the following formal definition. OpetopicType#25

Definition 2.3 (Frames).

$$\begin{aligned} \text{Frm}(tt) &\stackrel{\text{def}}{=} \text{Unit} \\ \text{Frm}(X_{\leq n-1}, X_n) &\stackrel{\text{def}}{=} \sum_{f : \text{Frm}(X_{\leq n-1})} \text{Src}(X_n, f) \times X_n(f) \end{aligned}$$

OpetopicType#320

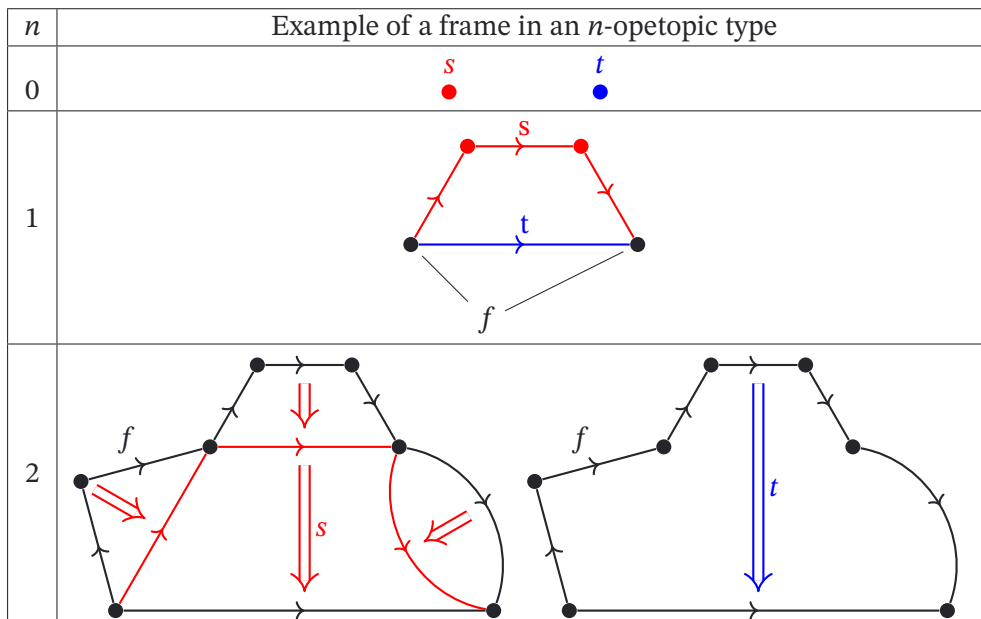


Figure 4: Example of frames (f, s, t) of dimensions 0, 1 and 2. In each case, the boundary frame f is colored in black, the input diagram s in red and the output cell t in blue. Keep in mind that all cells (points, arrows, and double arrows) in this table should come with a respective label, even though they were omitted here for readability.

Here is a first example of an opetopic type.

Definition 2.4 (Terminal opetopic type). We define for every $n \geq -1$ an opetopic type \mathbb{T}_n by induction on n as follows : Terminal#13

$$\begin{aligned}\mathbb{T}_{-1} &\stackrel{\text{def}}{=} \text{tt} \\ \mathbb{T}_{n+1} &\stackrel{\text{def}}{=} (\mathbb{T}_n, \lambda f. \text{Unit})\end{aligned}$$

This opetopic type is called the **unit opetopic type** or **terminal opetopic type** (since it can be characterized as terminal with respect to maps of opetopic types, which we don't define in this report).

Using the unit opetopic type, we directly get a definition of opetopes.

Definition 2.5 (Opetope). An n -dimensional opetope is a frame in \mathbb{T}_{n-1} .

The only remaining thing to define is $\text{Src}(X_n, f)$, that is, the type of pasting diagrams with labels in X_n and boundary frame f . We define them as an inductive family indexed over not only f but also X_n . The constructors resemble those for n -trees as described earlier, indeed a diagram consists either of :

- a leaf (a leaf of a diagram of arrows is a point, a leaf of a diagram of 2-opetopes is an arrow ; more generally, a leaf of a diagram of n -opetopes is a diagram consisting of a single $(n - 1)$ -opetope),
- a node together with its children.

Finally we need to define sources (input diagrams). Suppose we have defined an n -opetopic type $X = (X_0, \dots, X_n)$, and that we want to build a pasting diagram of n -opetopes in X . This diagram should have a base frame $(f, s, t) : \text{Frm}(X_0, \dots, X_{n-1})$ filled with some $\sigma : X_n(f, s, t)$. Then, every arrow in s should come with some child diagram.

Here is how we realize this intuition : instead of starting from (f, s, t) , we start with some diagram s' with boundary f , and **labels not in X_1** , but in

$$\begin{aligned}\text{Branch}(X_n) : \text{Frm}(X_0, \dots, X_{n-2}) &\rightarrow \mathcal{U} \\ f &\mapsto \sum_{s : \text{Src}(X_{n-1}, f)} \sum_{t : X_{n-1}(f)} \text{Src}(X_n, (f, s, t))\end{aligned}$$

An element of $\text{Src}(\text{Branch}(X_n), f)$ is an $(n - 1)$ -diagram labeled with n -diagrams, which is exactly what we want the children of a node to be (see also the similarity with n -trees, which consist of $(n - 1)$ -trees of n -trees).

So, given some $s' : \text{Src}(\text{Branch}(X_n), f)$, we then want to extract the $(n - 1)$ diagram of output faces of each branch, which we call $\text{floor}(s) : \text{Src}(X_{n-1}, f)$. Finally, we can say that a node for X_n consists of some frame $f : \text{Frm}(X_{n-2})$, some $s' : \text{Src}(\text{Branch}(X_n), f)$, some $t : X_{n-1}(f)$ and some $\sigma : X_n(f, \text{floor}(s'), t)$. Then we can say that such a diagram has for output frame $(f, \text{canopy}(s'), t)$ where $\text{canopy}(s')$ is obtained by pasting together the outermost diagrams of every branches. See figure 5 for an illustration.

Our definition however is not complete without addressing the way to compute canopy and floor, and also how to define the boundary frame of a leaf.

To do this we need a fundamental feature of Src : its monad structure.

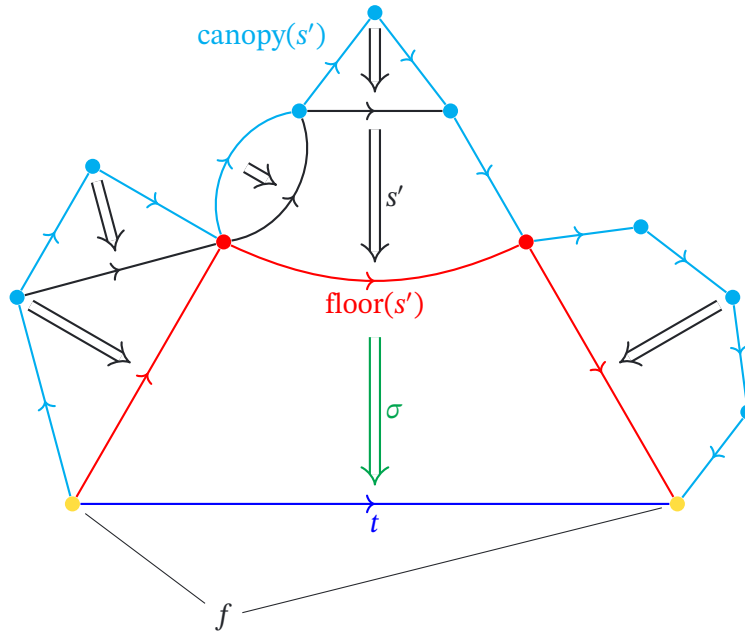


Figure 5: Example of the decomposition of a diagram node. The base frame f consists of the two yellow points. The branches s' are shown in black, their canopy in cyan, and their floor in red. The target filler t is in blue, and the node filler itself, σ , is in green.

It might not be clear because of the coloring limitations, but the two red points are also part of the canopy, and the two yellow points are both part of the canopy and the floor. And of course, the floor and the canopy are also part of s' .

2.3 The monad structure on Src

Let us fix an opetopic type X , and let $P : \text{Frm}(X) \rightarrow \mathcal{U}$. Then we have

$$\text{Src}(P, -) : \text{Frm}(X) \rightarrow \mathcal{U}$$

Now suppose we have $P' : \text{Frm}(X) \rightarrow \mathcal{U}$, and some family of functions

$$\sigma : \prod_{f : \text{Frm}(X)} P(f) \rightarrow P'(f)$$

Given any frame f in X and $s : \text{Src}(P, f)$, we want to be able to define another source $s' : \text{Src}(P', f)$ with the same diagram as s , but where the top-dimension labels have been renamed according to σ . Moreover, renaming s with respect to the identity should just yield s , and a composite of renamings should amount to renaming by a composite.

What we just stated is that Src should be an **endofunctor** of $(\text{Frm}(X) \rightarrow \mathcal{U})$, or, in more categorical notation, $\mathcal{U}/\text{Frm}(X)$. But that's not all.

OpetopicType#72

Let f be a frame in X . For every filler $t : P(f)$, there should be a particular source diagram s which just contains a copy of t . This way, (f, s, t) would be a **globe** diagram in (X, P) , as informally defined in remark 2.1. So we should have a family of maps

OpetopicType#78

$$\eta : \prod_{f : \text{Frm}(X)} P(f) \rightarrow \text{Src}(P, f)$$

Finally suppose we have some $s : \text{Src}(\text{Src}(P), f)$. This means that s is a diagram whose top-dimension fillers are themselves diagrams. Given such a diagram, we should be able to "flatten" the diagram to obtain a source $s' : \text{Src}(P, f)$. For example, in dimension 1, a sequence of sequences of labeled arrows can be "flattened" to a sequence of arrows (this is just ordinary flattening operation on lists). So we get an additional family of maps

OpetopicType#84

$$\mu : \prod_{f : \text{Frm}(X)} \text{Src}(\text{Src}(P), f) \rightarrow \text{Src}(P, f)$$

Those three observations (functoriality, η and μ) should give Src a kind of **monad structure** on the "category" $\mathcal{U} / \text{Frm}(X)$. However, making this precise formally is tricky once again because of coherence issues : the diagrams witnessing the coherence laws of the monad might contain non-trivial homotopical information, which we want to avoid.

This is where Agda's rewrite rules intervene. By postulating rewrite rules about Src , its functoriality, η and μ even before they are formally defined, we can then define them in a way that computes well.

OpetopicType#139-309

Remark 2.6. Finding the right approach to defining η and μ in such a way that Agda's rewrite rules behaved nicely was actually what took most of the time of my internship. Although it is a very interesting subject in itself, I preferred to focus on a concrete, definite result in this report ; namely the equivalence between categories as defined in HoTT and a special class of ∞ -categories defined using opetopic types (see section 3).

Using these monad laws, we can now formally define floor, canopy, and the boundary frame of a leaf.

For instance, given an opetopic type $((X_{\leq n}, P), Q)$, a frame $f : \text{Frm}(X)$ and a label $x : P(f)$, the corresponding source for Q has boundary $(f, \eta_f(x), x)$. In other words, the constructor leaf has type

OpetopicType#371

$$\text{leaf} : \prod_{f : \text{Frm } X} \prod_{x : P(f)} \text{Src}(Q, (f, \eta_f(x), x))$$

For floor and canopy, remember the definition of $\text{Branch}(Q)$: given a frame $f : \text{Frm}(X)$, and element of $\text{Branch}(Q, f)$ is a triple (s, t, σ) where $s : \text{Src}(P, f)$, $t : P(f)$ and $\sigma : \text{Src}(Q, (f, s, t))$. So first and second projection give maps $\pi_1 : \text{Branch}(Q, f) \rightarrow \text{Src}(P, f)$ and $\pi_2 : \text{Branch}(Q, f) \rightarrow P(f)$ for every frame $f : \text{Frm}(X)$. Now given some $s : \text{Src}(\text{Branch}(Q, f))$, functoriality of Src applied to π_2 gives floor(s) : $\text{Src}(P, f)$, and functoriality applied to π_1 gives an element of $\text{Src}(\text{Src}(P), f)$, which by applying μ_f then becomes canopy(s) : $\text{Src}(P, f)$. In the end, we can say the node constructor has type

OpetopicType#374

$$\prod_{f : \text{Frm}(X)} \prod_{t : P(f)} \prod_{s : \text{Src}(\text{Branch}(Q), f)} Q(f, \text{floor}(s), t) \rightarrow \text{Src}(Q, (f, \text{canopy}(s), t))$$

This concludes our formal definition of opetopic types. We will not give further details on the definitions of η and μ , nor on the rewrite rules necessary to make every definition type-check. For more informations on the monad structure of opetopes and opetopic types, see [Koc+10]. For more details on the implementation in type theory and the kind of rewrite rules at play, see [AFS21] or the agda code.

Now we have almost everything we need to tackle definitions of higher structures in type theory using opetopic types. We just need a notion of opetopic type of infinite dimension. The definition is by coinduction, over a base opetopic type.

Definition 2.7. Let X be an opetopic type. An ∞ -opetopic type over X is the coinductive data of :

- a family of fillers $\text{Fill}(X_\infty) : \text{Frm}(X) \rightarrow \mathcal{U}$
- an ∞ -opetopic type $\text{Hom}(X_\infty)$ over $(X, \text{Fill}(X_\infty))$.

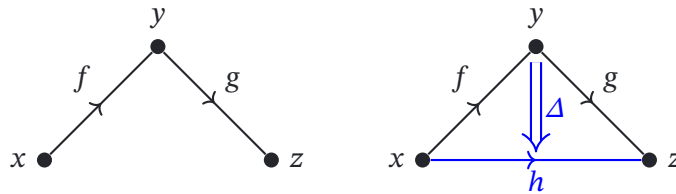
When X is not explicitly given, we will mean an ∞ -opetopic type over the trivial opetopic type $\text{tt} : \text{OType}(-1)$.

Notation. Given an opetopic type X of dimension n and an ∞ -opetopic type X_∞ over X , we might write $X_\infty = (X_{n+1}, X_{n+2}, \dots)$, where $X_{n+1} \stackrel{\text{def}}{=} \text{Fill}(X_\infty)$, $X_{n+2} \stackrel{\text{def}}{=} \text{Fill}(\text{Hom}(X_\infty))$, etc.

3 Higher categories

The simplicial approach to higher categories defines an $(\infty, 1)$ -category as a simplicial set verifying a "horn-filling condition", that is, given a certain kind of partial labeling for a given simplex, there exists some extension of it that labels all faces of the simplex.

For instance, in dimension 2, given a pair of arrows (f, g) as on the left below, there exists an extension (h, Δ) as on the right.



This gives a "definition" of the composite of f and g as h . "Definition" is in quotation marks because we only require existence of such a composite, not unicity. Then we declare arrows to be equivalent if there is a 2-cell witnessing their equivalence. Then the filling conditions in all dimensions imply that this composite is well-defined up to equivalence,

and it is associative up to equivalence, etc. And that's how we get all the higher coherence structure simplicially.

However, our approach takes place in Homotopy Type Theory, where the fundamental objects, types, already have a homotopical interpretation. Because of this, our definition of $(\infty, 1)$ -category won't need to merely state the existence of some fillers : it will state that the data of such fillers is *contractible*.

3.1 Definitions

The *sources* will play the role of horns. We give the following definition :

Definition 3.1. Given an opetopic type $((X, P), U)$ of dimension at least 1, a frame $f : \text{Frm}(X)$ and a source $s : \text{Src}(P, f)$, the type of **horn-fillers** for s is defined to be

Structures#29

$$\sum_{t:P(f)} U(f, s, t)$$

We say that $((X, P), U)$ is **fibrant** if for every frame and source, the type of its horn-fillers is contractible.

Structures#32

We coinductively define an ∞ -opetopic type X_∞ over an opetopic type X to be **ext-fibrant** if

Structures#38

- $((X, \text{Fill}(X_\infty)), \text{Fill}(\text{Hom}(X_\infty)))$ is fibrant,
- $\text{Hom}(X_\infty)$ is ext-fibrant.

Remark 3.2. Being fibrant doesn't imply that $U(f, s, t)$ is contractible for every f, s , and t . It only implies that given $\Delta : U(f, s, t)$ and $\Delta' : U(f, s, t')$, there is path $p : t = t'$ and a path q between Δ and Δ' over p .

Proposition 3.3. *Being fibrant is a proposition, that is : any two witnesses of fibrancy for X are equal. Being ext-fibrant is also a proposition.*

Structures#35,50

Proof. Being contractible is a proposition, and a \prod -type whose codomain is a family of propositions is itself a proposition. \square

We arrive at our definition of $(\infty, 1)$ -category.

Definition 3.4. An ∞ -**groupoid** is an ∞ -opetopic type X_∞ over tt that is ext-fibrant.

An $(\infty, 1)$ -**category** is an ∞ -opetopic type X_∞ over tt such that $\text{Hom}(X_\infty)$ is ext-fibrant.

Category#156

Remark 3.5. This definition was first given in [AFS21], where the authors prove that every ∞ -groupoid is entirely characterized by its base type, and that all types give rise to an associated ∞ -groupoid, hence the main statement of the paper "types are *internal* ∞ -groupoids. Indeed, the whole point of univalent foundations is to be able to interpret types as spaces, or rather

∞ -groupoids, but this was the first result actually implementing this intuition as a statement *inside* of the type theory.

However, the definitions and proof they give are done in a slightly different type theory, which cannot be exactly replicated within Agda. At the time of writing this report there is no known computer-checked proof of the statement "types are internal ∞ -groupoids".

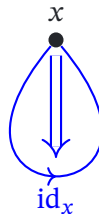
3.2 First consequences of the definition

Let C be an $(\infty, 1)$ -category. We write X_n for its underlying n -opetopic type and $P_n : \text{Frm}(X_n) \rightarrow \mathcal{U}$ for its n -labels. By definition of $(\infty, 1)$ -category, we have for every $n \geq 1$, $((X_n, P_n), P_{n+1})$ is fibrant. We will refer to the elements of $P_0(\text{tt})$ as the **objects** of C , and allow ourselves to write $x : C$ instead of $x : P_0(\text{tt})$. Given two objects x, y of C , we refer to the elements of $P_1(\text{tt}, x, y)$ as the **morphisms from x to y** and write $\text{hom}_C(x, y)$ as a shortcut for $P_1(\text{tt}, x, y)$. Since we already use the letter f to refer to frames, we will rather use the letters a, b, c, \dots to name morphisms.

Let $f : \text{Frm}(X_n)$ and $s : \text{Src}(P_n, f)$. We have that $\sum_{t : P_n(f)} P_{n+1}(f, s, t)$ is contractible. The first component of the center of contraction gives us some $t : P_n(f)$. We call this t the **composite** of s , written $\text{comp}(s)$.

Structures#105

For instance, when $n = 1$, sources are given by a finite sequence of morphisms $a_i : \text{hom}_C(x_i, x_{i+1})$ ($1 \leq i < n$ for some $n : \mathbb{N}$). Thus, fibrancy instantly gives us a notion of composition for arbitrary sequences of morphisms (which we may write $\text{comp}(a_1, \dots, a_n)$ or even $a_1 \cdot \dots \cdot a_n$). In particular, when $n = 0$, we have a composite for the source $s = \text{leaf}(x)$, which we interpret as the *identity morphism* of x and write $\text{id}_x : \text{hom}_C(x, x)$.



Remark 3.6. Notice that in the context of simplicial sets, the identity cells are given by the degeneracies map of the simplices, and the horn-filling conditions for the 2-simplex only allow the definition of binary composition of morphisms. Whereas in the opetopic context, there are no degeneracy maps, but the variety of opetopic shapes permits the construction of identity cells as a result of fibrancy.

Now let's use fibrancy to deduce that composition is associative and unital with respect to the identity morphisms we just defined.

Consider objects x and y and a morphism $a : \text{hom}_C(x, y)$. We want to prove that $\text{comp}(\text{id}_x, a) = a$. Here's the general idea : since the type of horn-fillers of any given source is contractible, it is in particular a proposition. Hence, it suffices to find a source s and two fillers

Category#104

$H : P_2(\dots, s, \text{comp}(\text{id}_x, a))$ and $H' : P_2(\dots, s, a)$ to deduce that (a, s) and $(\text{comp}(\text{id}_x, a), s)$ are equal as horn fillers. This would immediately entail that $a = \text{comp}(\text{id}_x, a)$.

The source we consider is $s = \eta(a)$. To find H , we paste the horn-filler given by fibrancy for the identity of x to the one witnessing the composition of id_x and f , as shown in figure 6. This gives a source $s : \text{Src}(P_2, \eta(a), \text{comp}(\text{id}_x, a))$. Using fibrancy in the next dimension, we get $H \stackrel{\text{def}}{=} \text{comp}(s) : P_2(\dots, \eta(a), \text{comp}(\text{id}_x, a))$. And for H' , we apply the same reasoning to $s' \stackrel{\text{def}}{=} \text{leaf}(a) : \text{Src}(P_2, \eta(a), a)$. Thus composition is left-unital.

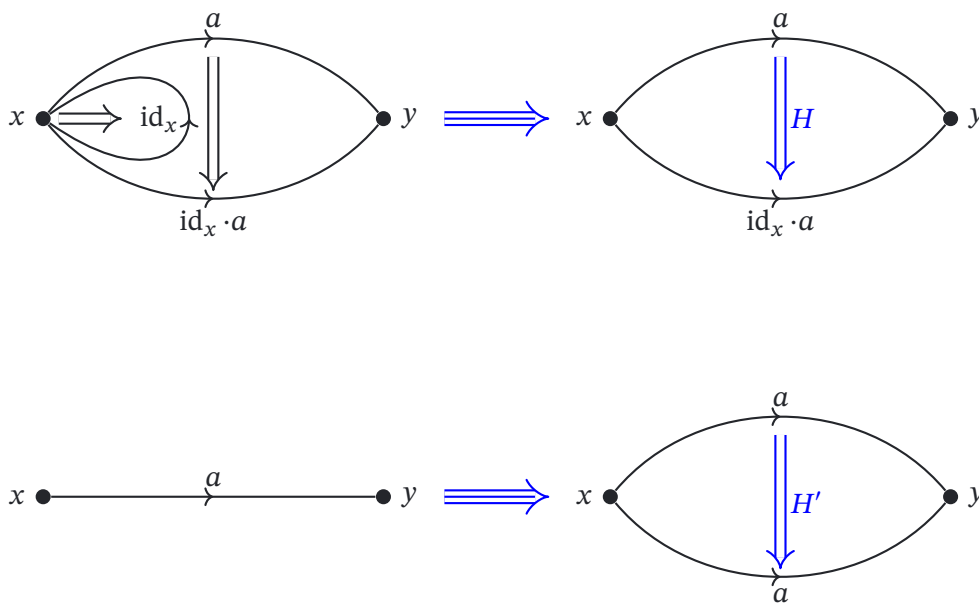


Figure 6: Diagrams showing how fibrancy in dimension 3 gives the 2-dimensional horn-fillers $(\text{id}_x \cdot a, H)$ and (a, H') of $\eta(a)$, whose equality as horn-fillers then implies left-unitality of composition.

Remark 3.7. In some sense, $\text{comp}(s')$ is the identity of a , which we could write id_a even though a is a morphism and not an object. More generally, the ability to "compose" leaves gives and "identity morphism" to every cell in an $(\infty, 1)$ -category.

Right unitality follows in the same way. As for associativity, we do the same reasoning with the diagrams of figure 7. Actually, we can do even better than that : the *three* ways of composing three morphisms coincide.

Category#116

Category#129

$$(a \cdot b) \cdot c = a \cdot b \cdot c = a \cdot (b \cdot c)$$

Or, written with a less ambiguous notation :

$$\text{comp}(\text{comp}(a, b), c) = \text{comp}(a, b, c) = \text{comp}(a, \text{comp}(b, c))$$

The additional equality uses the same reasoning as before applied to the canonical horn-filler for the sequence of morphisms (a, b, c) .

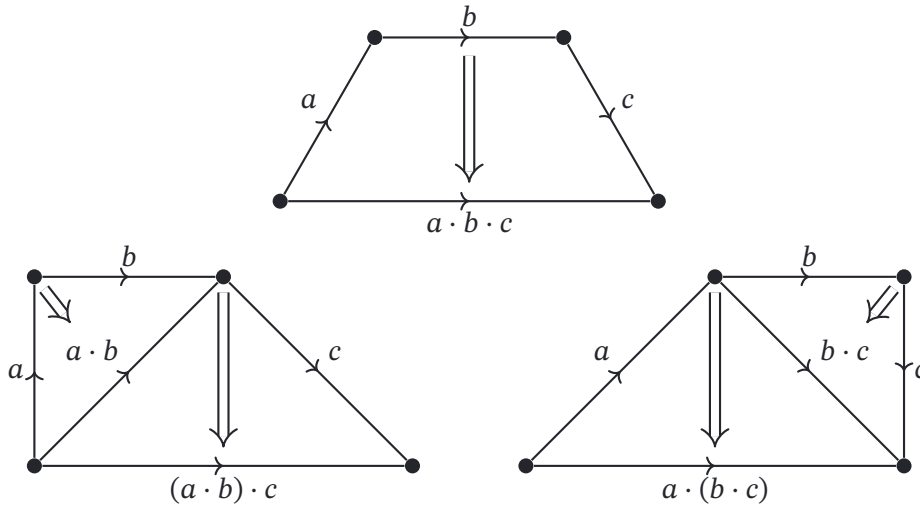


Figure 7: Diagrams showing that the three ways of composing 3 morphisms are ultimately equivalent. Notice that they indeed share the same input source.

3.3 Truncatedness

In Homotopy Type Theory, since types are interpreted as homotopy types, they have a notion of truncatedness (also called h-level, for "homotopy level"). A type is -2 -truncated if it is contractible, it is -1 -truncated if it is a proposition, 0 -truncated if it is a set, 1 -truncated if it is a groupoid, etc.

Since opetopic types represent n -dimensional structures for all n , it is natural to define a notion of truncatedness for them too. We give the following coinductive definition

Definition 3.8. An ∞ -opetopic type X_∞ over some opetopic type X is said to be n -truncated if :

- for every frame $f : \text{Frm}(X)$, the type $\text{Fill}(X_\infty)(f)$ is n -truncated,
- the opetopic type $\text{Hom}(X_\infty)$ is $(n - 1)$ -truncated.

Structures#94

Where by convention for truncatedness levels, $(-2) - 1 = -2$.

Remark 3.9. Since a type being n -truncated is a proposition, it follows that an opetopic type being n -truncated is also a proposition.

Structures#101

Next we are going to prove that this result is well-behaved with respect to fibrancy. More precisely, we have the following result.

Theorem 3.10. Let X_∞ be an opetopic type over some opetopic type X . If X_∞ is ext-fibrant, then X_∞ is n -truncated if and only if for every frame $f : \text{Frm}(X)$, the type $\text{Fill}(X_\infty)(f)$ is n -truncated.

Structures#121

The proof relies on the following theorem from [Uni13] about identity type systems, later called **the fundamental theorem of identity types** in [Rij20].

Theorem 3.11 ([Uni13], Th. 5.8.2). *Let X be a type, and $R : X \rightarrow \mathcal{U}$ a family of types over X . The following are equivalent.*

- The type $\sum_{x:X} R(x)$ is contractible.
- Given any $x_0 : X$ and $r_0 : R(x_0)$, the induced transport map

$$\begin{aligned} (x_0 = y) &\rightarrow R(y) \\ \text{refl} &\mapsto r_0 \end{aligned}$$

is an equivalence for every $y : X$.

Proof of theorem 3.10. Let X_∞ be ext-fibrant. If X_∞ is n -truncated, then by definition $\text{Fill}(X_\infty)(f)$ is n -truncated for all $f : \text{Frm}(X)$. Now suppose that $\text{Fill}(X_\infty)(f)$ is n -truncated for all $f : \text{Frm}(X)$. Then by the lemma, fibrancy of $(X, \text{Fill}(X_\infty), \text{Fill}(\text{Hom}(X_\infty)))$ implies that $\text{Fill}(\text{Hom}(X_\infty))(f, s, t)$ (with $(f, s, t) : \text{Frm}(X, \text{Fill}(X_\infty))$) is equivalent to the type $\text{comp}(s) = t$ which $(n-1)$ -truncated. Thus by coinductive hypothesis, $\text{Hom}(X_\infty)$ is $(n-1)$ -truncated. Together with the truncatedness assumption on $\text{Fill}(X_\infty)$, it follows that X_∞ is n -truncated. \square

3.4 Equivalence with categories

We are now ready for the main result of this report. Let us recall the definition of categories from [Uni13].

Definition 3.12. A category is the data of :

- a type Ob of objects,
- a family $\text{hom} : \text{Ob} \rightarrow \text{Ob} \rightarrow \mathcal{U}$ of types of morphisms between objects,
- a composition operation on morphisms,
- a distinguished morphism in every $\text{hom}(x, x)$ called the identity of x ,
- witnesses that composition is associative and unital with respect to the identities,
- a witness that for every $x, y : \text{Ob}$, $\text{hom}(x, y)$ is a set.

We now give our alternative definition of category :

Definition 3.13. A 1-category is an $(\infty, 1)$ -category C such that $\text{Hom}(C)$ is 0-truncated.

Category#159

Theorem 3.14. *The type of categories and the type of 1-categories are equivalent.*

Category#254

Proof. First, consider a 1-category C . By definition, its Hom is 1-truncated, and the previous remarks about unitality and associativity of composition in $(\infty, 1)$ -categories allows us to associate a category to C .

The opposite direction is way harder. Let C be a category. We define an ∞ -opetopic type X over tt , called the **opetopic nerve of X** , as follows :

- $X_0(\text{tt}) \stackrel{\text{def}}{=} \text{Ob}(C)$.
- $X_1(\text{tt}, x, y) \stackrel{\text{def}}{=} \text{hom}(x, y)$.
- To define X_2 , we first define the composite $\text{comp}'(s)$ of a source $s : \text{Src}(X_1, f)$ by induction on s . The composite of a leaf is the identity, whereas the composite of a node is the composite in C of its "head" and the composite of its "tail". We then pose $X_2(f, s, t) \stackrel{\text{def}}{=} (\text{comp}'(s) = t)$.
- For $n \geq 3$, $X_n(f) \stackrel{\text{def}}{=} \text{Unit}$.

Fibrancy of $((\dots, X_1), X_2)$ is a direct consequence of the converse direction of theorem 3.11.

The difficult part of the proof lies in the fibrancy of $((\dots, X_2), X_3)$, which amounts to saying that given a source diagram $s : \text{Src}(X_2, (f, c, t))$ (where every node is the composite of its children), the composite of its canopy c is equal to t . The proof is once again by induction on s , and relies on the following "compatibilities" between composition, η and μ :

- The composite of $\eta(a)$ is equal to a .
- Given $s : \text{Src}(\text{Src}(X_1), f)$, the composite of $\mu(s)$ is equal to the composite of the composites of the labels in s (that is, the composite of the source $s' : \text{Src}(X_1, f)$ obtained by applying the functoriality of Src to the "composite" operation). This is again proven by induction on s .

The complete details can be found in the agda code.

Truncatedness follows from the set-truncatedness of the hom-types in C and from the contractibility of the Unit type. Ext-fibrancy finally follows from truncatedness and fibrancy in dimension 2, using lemma 3.10 (this lemma can actually be avoided here, since in higher dimensions we're only considering the Unit type everywhere).

We then need to prove that the two above constructions are inverse from each other. The direction "category \rightarrow 1-category \rightarrow category" is rather straightforward. For the other direction however we need to verify one additional fact : that the iterated binary composition of morphisms in a 1-category is equal to the direct composition given by fibrancy. This is once again proven by induction and using associativity and unitality. \square

Conclusion

Together with the internal equivalence between types and ∞ -groupoids, this new and, as far as we now, never before formalized result give strong reason to believe that the opetopic definition of ∞ -categories in type theory is "right".

However, there is still a lot of work to be done before it can be used to actually *do* higher category theory. Currently, we have definitions for opetopic maps (and thus, higher functors), cartesian products, dependent opetopic types, Σ -opetopic types, the join of two opetopic types, and some other basic constructions. We have a candidate definition for (∞, n) -categories, which has however not yet been implemented, and for the opetopic universe of types (both due to Finster).

The problem of defining slice- ∞ -categories (and more generally, comma- ∞ -categories) is still open, as well as the problem of defining the *opetopic type* of opetopic maps. For both of these issues, a seemingly necessary first step would be to define *representable* opetopic types (*i.e.* those corresponding to the primitive opetopic shapes) and a good notion of tensor product.

In a more distant future, a definition of *localization* of ∞ -categories may render definable many ∞ -categories usually described using model category theory.

References

- [AFS21] Antoine Allieux, Eric Finster, and Matthieu Sozeau. *Types are Internal ∞ -Groupoids*. 2021. URL: <https://arxiv.org/abs/2105.00024>.
- [BD97] John C. Baez and James Dolan. “Higher-Dimensional Algebra III: n-Categories and the Algebra of Opetopes”. In: (1997). URL: <https://arxiv.org/abs/q-alg/9702014>.
- [Buc18] Ulrik Buchholtz. *Higher Structures in Homotopy Type Theory*. 2018. URL: <https://arxiv.org/abs/1807.02177>.
- [Koc+10] Joachim Kock et al. “Polynomial functors and opetopes”. In: *Advances in Mathematics* 224.6 (Aug. 2010), pp. 2690–2737. URL: <https://doi.org/10.1016/j.aim.2010.02.012>.
- [RS17] Emily Riehl and Michael Shulman. *A type theory for synthetic ∞ -categories*. 2017. URL: <https://arxiv.org/abs/1705.07442>.
- [Rij20] Egbert Rijke. *Introduction to Homotopy Type Theory*. <https://raw.githubusercontent.com/martinescardo/HoTTEST-Summer-School/main/HoTT/hott-intro.pdf>, 2020.
- [Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: <https://homotopytypetheory.org/book>, 2013.