Opetopic Methods in Homotopy Type Theory

Antoine Allioux

Institut de Recherche en Informatique Fondamentale
Université Paris-Cité, France
antoine.allioux@irif.fr

1 Introduction

In a previous work [3], we extended intensional type theory (ITT) [5] with opetopic types by introducing a universe of polynomial monads. Opetopes are many-to-one geometric shapes that are used in geometric approaches to higher algebra [2] and which can be described by a recursive construction on polynomial monads [4]. Opetopic types are then type-valued coinductive collections of cells whose geometry is governed by opetopes. This definition makes it easy to define weak algebras in a coherent fashion and we applied it to the definition of a range of fully coherent higher algebraic structures such as ∞-groupoids and $(\infty, 1)$-categories. These definitions have proved elusive in homotopy type theory (HoTT) [6] without requiring additional principles such as uniqueness of identity proofs.

In this talk, I take advantage of opetopic types to define the $(\infty, 1)$-category of types internally to type theory.

2 Opetopic Type Theory

Our work takes place in ITT — along with the common constructions that one can find in Agda [1] in which this work has been formalised — extended with a universe of polynomial monads $\mathcal{M} : \mathcal{U}$. This universe allows one to internalize the Baez-Dolan construction [2] on which our definition of opetopic types is based.

Polynomial monads The elements of the universe $\mathcal{M}$ are regarded as codes for our monads. For any monad $M : \mathcal{M}$, the data of its functorial part is given by a set of decoding functions (Figure 1). They can be regarded as a description of a signature of an algebraic theory: the elements of Idx_M, which we refer to as *indices* serve as the sorts of the theory, and for $i : \mathsf{Idx}_M$, the type $\mathsf{Cns}_M(i)$ is the collection of operation symbols whose "output" sort is i. The type $\mathsf{Pos}_M(c)$ then assigns to each operation a collection of "input positions" which are themselves assigned an index via the function Typ_M.

$$\begin{aligned}
&\mathsf{Idx}_M : \mathcal{U} \\
&\mathsf{Cns}_M : \mathsf{Idx}_M \to \mathcal{U} \\
&\mathsf{Pos}_M : \{i : \mathsf{Idx}_M\} \to \mathsf{Cns}_M(i) \to \mathcal{U} \\
&\mathsf{Typ}_M : \{i : \mathsf{Idx}_M\}\ (c : \mathsf{Cns}_M(i)) \\
&\qquad \to \mathsf{Pos}_M(c) \to \mathsf{Idx}_M
\end{aligned}$$

Figure 1: Decoding functions

Next, we equip this data with a unit η_M and a multiplication μ_M satisfying some laws endowing the functor with a structure of monad (Figure 2). μ_M sends a well-typed depth-2 tree of constructors of M to a constructor of M while preserving the positions and their typing. Crucially, this operation is *definitionally* associative and unital with unit η_M.

Baez-Dolan construction Our universe of monads is closed under the Baez-Dolan construction [2]. Given a monad $M : \mathcal{M}$ and a type family $X : \mathsf{Cell}_M$ with $\mathsf{Cell}_M :\equiv \mathsf{Idx}_M \to \mathcal{U}$, we define a new monad M/X. Its type of indices is $\sum_{(i,y):\sum_{i:\mathsf{Idx}_M} X(i)} \sum_{c:\mathsf{Cns}_M(i)} \overrightarrow{X}(c)$ — constructors of M whose inputs and output are decorated with elements in X — where $\overrightarrow{X}(c) :\equiv (p : \mathsf{Pos}_M(c)) \to X\ (\mathsf{Typ}_M(c,p))$ is the type of decorations of inputs of c with well-typed elements of X.

$$
\begin{aligned}
&\eta_M : (i : \mathsf{Idx}_M) \to \mathsf{Cns}_M(i)\\
&\mu_M : \{i : \mathsf{Idx}_M\}\ (c : \mathsf{Cns}_M(i))\\
&\qquad \to ((p : \mathsf{Pos}_M(c)) \to \mathsf{Cns}_M(\mathsf{Typ}_M(c,p)))\\
&\qquad \to \mathsf{Cns}_M(i)
\end{aligned}
$$

Figure 2: Operations of the monad

We name those quadruplets *frames* and denote them $(i,y) \triangleleft (c,x)$. We regard cell families $X : \mathsf{Cell}_{M/X}$ as fillers for those frames relating a configurations of inputs specified by (c,x) with the output (i,y).

Constructors of M/X indexed by a frame $(i,y) \triangleleft (c,x)$ are then well-founded trees of frames which multiply to $(i,y) \triangleleft (c,x)$ by using the multiplication μ_M to reduce the constructors and by forgetting the decorations of inner edges. The corolla having $(i,y) \triangleleft (c,x)$ for only node is a unit for the monad M/X. The great power of the opetopic approach is that such trees are very conveniently defined as an inductive type. However, their indexing making use of the monad structure, we require the monad laws to be definitional in order not to run into coherence issues.

Given a constructor $c : \mathsf{Cns}_{M/X}(i)$, its type of positions $\mathsf{Pos}_{M/X}(c)$ is the type of *paths* from the root of c to its different nodes. The typing function then assigns the frame corresponding to a node of c at a specified position.

Finally, the multiplication $\mu_{M/X}$ takes a tree $c : \mathsf{Cns}_{M/X}(i)$ as well as a family of trees $d : \overrightarrow{\mathsf{Cns}_{M/X}}(c)$ indexed by the positions of c and substitutes the trees specified by d for the nodes of c. We also assert that this multiplication is associative and unital by definition.

Opetopic types Given a monad M, a M-opetopic type $X : \mathcal{O}_M$ (Figure 3) is a coinductive collection of opetopic cells. We denote X_n the n-th cell family of an opetopic type X and we denote $X_{>n}$ the opetopic type of families X_m for $m > n$.

$$
\begin{aligned}
&X_0 : \mathsf{Cell}_M\\
&X_{>0} : \mathcal{O}_{M/X_0}
\end{aligned}
$$

Figure 3: Definition of $\mathcal{O}_M$

We define an opetopic type $X : \mathcal{O}_M$ to be *fibrant* if it has the following coinductive property. First, for any decorated constructor $(c,x) : \sum_{c:\mathsf{Cns}_M(i)} \overrightarrow{X_0}(c)$, there is a unique composite $y : X_0(i)$ along with a filler $X_1((i,y) \triangleleft (c,x))$. In other words, the type $\sum_{y:X_0(i)} X_1((i,y) \triangleleft (c,x))$ is contractible. Second, the opetopic type $X_{>0}$ is fibrant.

Applications This suffices to define ∞-groupoids as fibrant Id-opetopic types where Id is the identity monad which has a single trivially indexed unary constructor. Requiring the fibrancy property one dimension higher allows non-invertible 1-cells. We therefore define $(\infty,1)$-categories as Id-opetopic types X such that $X_{>0}$ is fibrant. In [3], we used this system to internalise the result stating that types are equivalent to ∞-groupoids.

In the present talk, I define the universe of types as an $(\infty,1)$-category using a straightforward generalisation of monads to their dependent counterparts.

References

[1] Agda development team. *Agda 2.6.1.1 documentation*. 2020. URL: `https://agda.readthedocs.io/en/v2.6.1.1/`.

[2] John C Baez and James Dolan. "Higher-dimensional algebra III. n-categories and the algebra of opetopes". In: *Advances in Mathematics* 135.2 (1998), pp. 145–206.

[3] Eric Finster, Antoine Allioux, and Matthieu Sozeau. "Types are internal ∞-groupoids". In: *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. 2021, pp. 1–13.

[4] Joachim Kock et al. "Polynomial functors and opetopes". In: *Advances in Mathematics* 224.6 (2010), pp. 2690–2737.

[5] Per Martin-Löf. "An intuitionistic theory of types: Predicative part". In: *Studies in Logic and the Foundations of Mathematics*. Vol. 80. Elsevier, 1975, pp. 73–118.

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