A Sequent Calculus for Opetopes

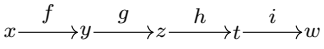
Cédric Ho Thanh IRIF, University of Paris, France
Pierre-Louis Curien IRIF, University of Paris, France
Samuel Mimram LIX, Palaiseau, France

***Abstract*—Opetopes are algebraic descriptions of shapes corresponding to compositions in higher dimensions. As such, they offer an approach to higher-dimensional algebraic structures, and in particular, to the definition of weak ω-categories, which was the original motivation for their introduction by Baez and Dolan. They are classically defined inductively (as free operads in Leinster's approach, or as zoom complexes in the formalism of Kock et al.), using abstract constructions making them difficult to manipulate with a computer. Here, we present a purely syntactic description of opetopes and opetopic sets as a sequent calculus. Our main result is that well-typed opetopes in our sense are in bijection with opetopes as defined in the more traditional approaches. We expect that the resulting structures can serve as natural foundations for mechanized tools based on opetopes.**

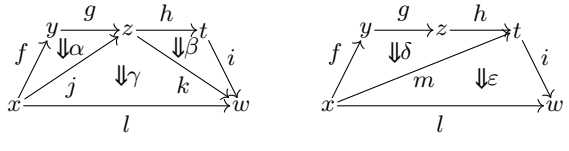
I. Introduction

Opetopes were originally introduced by Baez and Dolan in order to formulate a definition of weak ω-categories [2]. Their name reflects the fact that they encode the possible shapes for higher-dimensional operations: they are *ope*ration poly*topes*. Over the recent years, they have been the subject of many efforts to provide a good definition that would allow exploring their combinatorics [4], [10], [17]. One of the most commonly used nowadays is the formulation based on polynomial functors and the corresponding graphical representation using "zoom complexes" [16].

In order to grasp quickly the nature of opetopes, consider a sequence of four composable arrows



There are various ways in which we can compose them. For instance, we can compose f with g, as well as h with i, and then $g \circ f$ with $i \circ h$. Or we can compose f, g and h together all at once, and then the result with i. These two schemes for composing can respectively be pictured as simple cellular complexes



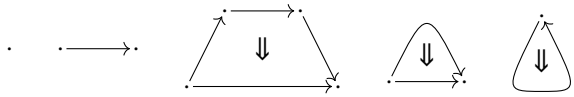
(1)

where x, y, z, t, w are 0-cells, f, g, h, i, j, k, l, m are 1-cells, and $\alpha, \beta, \gamma, \delta, \varepsilon$ are 2-cells.

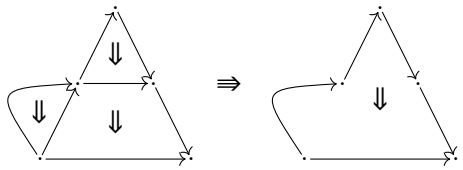
From there, the general idea of getting "higher-dimensional" is that we should take these compositions as "2-operations", which can be composed in various ways. For instance, in the first case, we can compose α with γ, and then β with the result, or all three at once, and so on. The opetopes describe all the ways in which these compositions can be meaningfully specified, in arbitrary dimension.

We can expect (and it is indeed the case) that the combinatorics of these objects is not easy to describe. In particular, a representation which is adapted to computer manipulations and proofs is desirable: we can for instance mention the *Opetopic* proof assistant for higher categories [6], which is based on opetopes. Recently, Mimram and Finster have used type theory as a convenient tool to describe globular weak ω-categories [7]. Our long-term goal is to achieve a similar type-theoretic presentation for opetopic weak ω-categories [2], [11], and we begin here by defining a representation of opetopes and opetopic sets of type-theoretic flavor.

Opetopes. Let us now informally define opetopes by induction on their dimension. There is a unique 0-opetope, drawn as a point. An $(n+1)$-opetope is made out of n-opetopes, and has a source and a target. In small dimensions, opetopes can be described using drawings of the kind above. For instance, the following are, from left to right, the unique 0-opetope, the unique 1-opetope, drawn as an arrow, and three 2-opetopes, respectively,



while the following drawing represents a 3-opetope:



In these drawings, the target of the 1-opetope is a point, the target of the 2-opetopes is the arrow at the bottom, and the target of the 3-opetope is the 2-opetope on the right, while the source of the 1-opetope is again a point, the source of the 2-opetopes is a diagram made of arrows, and the source of the 3-opetope is the diagram made of 2-opetopes on the left of the central arrow.

At this stage, we can already make the following observations:

1) each drawing contains only one top-dimensional cell, whose dimension determines the dimension of the opetope,
2) if an opetope has a non empty source, then its source and target have the same source (which might be empty) and the same target,

978-1-7281-3608-0/19/$31.00

3) if an opetope has an empty source, then its target has the same source and target,
4) an $(n+1)$-cell can have multiple n-cells in its source (including none), but always has exactly one n-cell in its target.

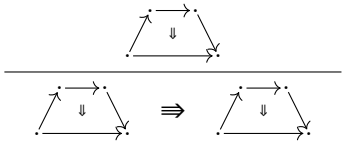
By the above remarks, an n-opetope has a unique top-dimensional cell α, whose target consists of one cell β whose source and target are the same as the ones of the source of α. It follows recursively that the opetope is entirely determined by the source of α, which we call the associated *pasting scheme*, which is of dimension $n-1$. The opetope associated to a pasting scheme of dimension n can be obtained by adding a single cell β of dimension n parallel to the pasting scheme, and an $(n+1)$-cell α from the pasting scheme to β.

One of the goals of the article is to introduce a syntax to faithfully describe opetopes, and to characterize the terms corresponding to opetopes using inference rules.

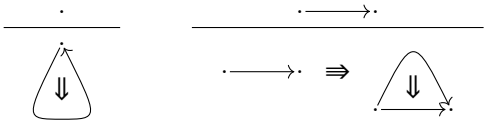
Generating opetopes. The sequent calculus introduced in this article, called OPT, formalizes the observation that pasting schemes are precisely all the shapes one can generate with the following operations.

1) *Introduction of a point.* There is a a unique 0-opetope (the *point*).

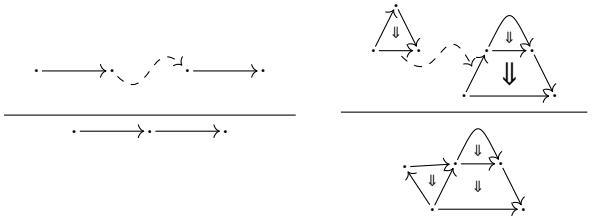
2) *Shift to the next dimension.* Given an n-opetope ω, we can form the $(n+1)$-*extrusion* whose source and target are ω, as illustrated below:



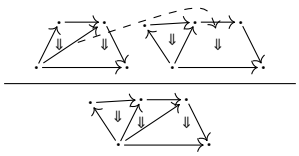
3) *Introduction of degeneracies.* Given an n-opetope ω, we can build an $(n+2)$-opetope with empty source, whose target is the extrusion of ω, as illustrated below for $n=0$ and $n=1$:



4) *Grafting.* Given an $(n+1)$-opetope α and an $(n+1)$-pasting scheme β such that the source of β contains an n-cell of the same shape as the target of α, we can *graft* α to β:



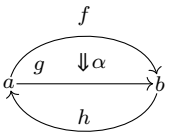
Another fundamental operation we will use is *substitution*, which consists in replacing an opetope in a pasting scheme by another pasting scheme as illustrated below:



Substitution and grafting are (recursively) mutually dependent.

We formally define our syntax and sequent calculus OPT in II and illustrate them through examples in III. We recall the traditional definition of opetopes in IV and show in section V that they precisely correspond to the terms derivable in the system OPT (14).

Opetopic sets. Collections of opetopes which are glued along their faces are called *opetopic sets*. For instance, the following is a depiction of an opetopic set



(2)

consisting of two 0-opetopes, three 1-opetopes and one 2-opetope. Note that it is not itself an opetope, nor even a pasting scheme (i.e., the source of an opetope). Formally, as in other traditional cases (e.g. simplicial or cubical sets), those can be defined as presheaves on the category of opetopes, of which a presentation was given in [12].

Just as for opetopes, it is desirable to be able to manipulate and reason about (finite) opetopic sets in a syntactic fashion. Therefore, on top of the sequent calculus OPT, we give in section VI a syntax and a sequent calculus OPTSET that allows us to derive terms corresponding precisely to finite opetopic sets (19). Finally, in section VII, we give a variant of this system called OPTSET_{M}, where the stratification opetopes vs opetopic sets is removed, and show that it has the same expressive power as OPTSET. We conclude in section VIII.

Related works. We were inspired by the polynomial opetopes (also called "zoom complexes") of Kock et al. [16], which we briefly recall in section IV. Those are themselves equivalent to Leinster's opetopes [17]. It is known that the latter are incompatible with Cheng's opetopes [4], which should be thought of as a symmetric variant. There is a closely related notion of *multitope* [11], [9] which is defined in terms of multicategories (whence the *multi*) instead of operads (*ope*); the two notions can be shown to be equivalent [12]. A syntax for multitopes was proposed in [11], it is closely related to the one given here and allows to faithfully represent opetopes; however, to our knowledge, their description is incomplete in the sense that those terms corresponding actually to opetopes are not characterised by formal rules.

The Opetopic proof assistant [6] for weak higher categories relies on the notion of higher-dimensional tree. In that system, the notion of opetope is built-in, so that we have to trust the implementation. In contrast, the present approach allows us to reason about the construction of opetopes. We moreover believe that the ability to reason by induction on the proof trees, together with the very explicit nature of our syntaxes,

will allow for optimizations in the automated manipulations of opetopes. Another proof assistant for weak higher categories, called CaTT [7], starts from the same idea of generating well-formed pasting schemes through inference rules. However, it is based on globular shapes instead of opetopic ones, making a comparison with the present work difficult: since their introduction, people have unsuccessfully tried to compare the resulting respective categorical formalisms; we hope that their formulation in a common logical language might be of help in this task. We should also mention here the *Globular* proof assistant [3], also based on globular shapes, which is quite popular, notably thanks to its nice graphical interface. In order to witness for the effective character of our sequent calculus, a Python implementation was performed [13] and will be presented in details elsewhere [5].

II. Opt: a sequent calculus for opetopes

Syntax. We first introduce our syntax for describing pasting schemes (which is equivalent to describing opetopes, see 6).

As explained in the introduction, a typical pasting scheme is pictured on the left of (1). We shall use some of the names of the cells of this picture as variables, and encode the pasting scheme as

$$\gamma(j\leftarrow\alpha, k\leftarrow\beta)$$

Here, $j, k, \alpha, \beta, \gamma$ are now variables, equipped with a dimension (1 for j, k and 2 for α, β, γ), and the notation is meant to be read as "the variable γ in which α (resp. β) has been formally grafted on the input labeled j (resp. k)". Such a term will be given a type

$$i(t\leftarrow h(z\leftarrow g(y\leftarrow f))) \multimap x \multimap \varnothing$$

which expresses the fact that the source is the "composite" $i \circ h \circ g \circ f$, the source of the source is x. Since the pasting scheme was 2-dimensional, there is no further iterated source, and we conclude the sequence by a $\varnothing$ symbol (which can be read as the only (-1)-dimensional pasting scheme).

A type essentially describes a zoom complex in the sense of [16], from which we borrow the symbol $\multimap$. Similarly, the degenerate pasting scheme on the left below will be denoted by the typed term figured on the right:

$$\alpha : \underline{x} \multimap x \multimap \varnothing$$

where the term $\underline{x}$ denotes a degenerate 1-dimensional pasting scheme with x as source.

Variables. We suppose fixed a $\mathbb{N}$-graded set $\mathbb{V}$ of *variables*, the grading being called the *dimension* of the variable. Given $n \in \mathbb{N}$, we write $\mathbb{V}_n$ for the set of variables of dimension n and always suppose that this set is countably infinite. By convention, we set $\mathbb{V}_{-1}$ to be the empty set.

Terms. The terms are likewise graded by a dimension $n \in \{-1\} \sqcup \mathbb{N}$. The sets $\mathbb{T}_n$ of *n-terms*, or syntactic *n-pasting schemes*, are defined by induction as follows:

- $\mathbb{T}_{-1} = \{\varnothing\}$ (i.e. the unique (-1)-term is $\varnothing$),
- for $n \geq 0$, the n-terms are either of the form $\underline{u}$, for $u \in \mathbb{T}_{n-1}$, or are produced by the grammar

$$t \quad ::= \quad x(y_1\leftarrow t_1, \dots, y_k\leftarrow t_k) \tag{3}$$

with $x \in \mathbb{V}_n$, $y_1, \dots, y_k \in \mathbb{V}_{n-1}$, and $t_1, \dots, t_k \in \mathbb{T}_n$.

The terms of the form $\underline{u}$ (resp. the other terms) are called *degenerate* (resp. *non-degenerate*) terms, or *empty* (resp. *non-empty*) syntactic pasting schemes. By convention, the sequence $y_1\leftarrow t_1, \dots, y_k\leftarrow t_k$ above is always considered up to permutation: for σ a bijection of the set $\{1, \dots, k\}$, the terms $x(y_1\leftarrow t_1, \dots, y_k\leftarrow t_k)$ and $x(y_{\sigma(1)}\leftarrow t_{\sigma(1)}, \dots, y_{\sigma(k)}\leftarrow t_{\sigma(k)})$ are considered equal. Note that the above definition entails that $\mathbb{T}_0 = \mathbb{V}_0$ (in particular, there are no degenerate 0-terms).

For instance, if $f, g \in \mathbb{V}_1$, and $a \in \mathbb{V}_0$, then the following is an element of $\mathbb{T}_1$:

$$g(a\leftarrow f())$$

To make notations lighter, we omit the parentheses "()", so that the previous 1-term can be more concisely written as $g(a\leftarrow f)$. A term of the form $g(a_1\leftarrow f_1, \dots, a_k\leftarrow f_k)$ will oftentimes be abbreviated as $g(\overrightarrow{a_i\leftarrow f_i})$, leaving k implicit. Note that an expression of the form $t(x\leftarrow u)$ is not a term when t is not a variable. However, the grafting notation can be conveniently extended to give a meaning to such terms, see 1. Given a term $t \in \mathbb{T}_n$, we write $t^\bullet$ for the set of variables of dimension n occurring in t, sometimes called the *nodes* of t.

Types. A *type* A of dimension n, or *n-type*, is a sequence of terms written as follows:

$$A \quad = \quad s_n \multimap s_{n-1} \multimap \cdots \multimap s_0 \multimap \varnothing$$

with $s_i \in \mathbb{T}_i$ for $0 \leq i \leq n$. A typing of a term $t \in \mathbb{T}_{n+1}$ is a pair of the form $t : A$, for A an n-type. For a typing $t : A$ as above (correctly derived in our system), we call s_{n+1-i} the i-th (iterated) *source* of t. We then write $\mathsf{s}\, t \coloneqq s_n$, $\mathsf{s}\, s_i = s_{i-1}$, and $\mathsf{s}^i\, t \coloneqq s_{n+1-i}$ for the iterated sources. We also set $\mathsf{s}^0\, t \coloneqq t$.

Contexts. As traditionally, a context Γ is a mapping from variables to types, denoted as usual as a list of typings $x : A$ (where we do not distinguish between two contexts differing only by the ordering of the typings they contain). We write $\mathbb{V}_{\Gamma,k}$ for the set of k-variables typed in Γ, and we set $\mathbb{V}_\Gamma \coloneqq \bigsqcup_{k\in\mathbb{N}} \mathbb{V}_{\Gamma,k}$. We write $\mathbb{T}_{\Gamma,k}$ for the set of k-terms whose variables (in any dimension) are in $\mathbb{V}_\Gamma$, and we set $\mathbb{T}_\Gamma \coloneqq \bigsqcup_{k\in\mathbb{N}} \mathbb{T}_{\Gamma,k}$. Our typing system will maintain the following invariant: for a derivable context Γ, if x occurs in the typing of a variable of Γ, then $x \in \mathbb{V}_\Gamma$. Note that in any context Γ, if a variable $x \in \mathbb{V}_{\Gamma,k}$ occurs in the type of $y \in \mathbb{V}_{\Gamma,l}$, then $k < l$, and thus there can never be any cyclic dependency among variables.

Sequents. A *sequent* is an expression of the form

$$E \triangleright \Gamma \vdash t : A$$

where Γ is a context, and the right hand side is a typing. We sometimes write $\vdash_n$ instead of $\vdash$ when we want to highlight the fact that $t \in \mathbb{T}_n$. Additionally to this traditional data, our sequents contain a set E which is a graded equational theory on $\mathbb{V}_\Gamma$, i.e., a set of formal equalities beween variables of Γ of

same dimension. We write $=_E$ for the congruence it induces on terms. If $x =_E y \in t^\bullet$, then by convention $x \in t^\bullet$, so that x and y really are interchangeable.

In the following, sequents are implicitly considered up to α-equivalence: two sequents $(E \triangleright \Gamma \vdash t : A)$ and $(F \triangleright \Delta \vdash u : B)$ are *α-equivalent* when there exists a dimension-preserving bijection σ of $\mathbb{V}$ with

$$(E \triangleright \Gamma \vdash t : A) \quad = \quad \left(F^\sigma \triangleright \Delta^\sigma \vdash u^\sigma : B^\sigma\right),$$

where $(-)^\sigma$ is the obvious substitution according to σ.

Inference rules. The inference rules for our sequent calculus OPT, shown below, should be read along with the illustrations provided in the introduction. The most subtle rule is `graft`, which requires the graft notation and the substitution operation, introduced below, and requires side conditions which ensure that the grafting implements a pushout (see section IV).

- *Introduction of points*: introduces 0-cells, also called points.
$$\frac{x \in \mathbb{V}_0}{\triangleright x : \varnothing \vdash_0 x : \varnothing}\ \texttt{point}$$
- *Shift to the next dimension*: takes a term t and introduces a new cell x having t as source.
$$\frac{E \triangleright \Gamma \vdash_n t : A \qquad x \in \mathbb{V}_{n+1}, x \notin \mathbb{V}_\Gamma}{E \triangleright \Gamma, x : t \multimap A \vdash_{n+1} x : t \multimap A}\ \texttt{shift}$$
- *Introduction of degeneracies*: derives cells whose source is an empty pasting diagram.
$$\frac{E \triangleright \Gamma \vdash_n x : A \qquad x \in \mathbb{V}_n, \delta \in \mathbb{V}_{n+2}, \delta \notin \mathbb{V}_\Gamma}{E \triangleright \Gamma, \delta : \underline{x} \multimap x \multimap A \vdash_{n+2} \delta : \underline{x} \multimap x \multimap A}\ \texttt{degen}$$
- *Grafting*: glues an n-cell x onto an n-term t along a variable $a \in s^\bullet_{n-1}$ with $s_{n-1} \coloneqq \mathsf{s}\, t$.
$$\frac{E \triangleright \Gamma \vdash_n t : s_{n-1} \multimap \dots \qquad F \triangleright \Delta \vdash_n x : A}{G \triangleright \Gamma \cup \Delta \vdash_n t(a \leftarrow x) : s_{n-1}[\mathsf{s}\, x/a] \multimap s_{n-2} \multimap \cdots}\ \texttt{graft}$$
where G is the union of E, F, and potentially a set of additional equalities incurred by the substitution $s_{n-1}[\mathsf{s}\, x/a]$ (see below). When using this rule, we always assume that the following side-conditions are fulfilled:
  - x is a variable (we make this assumption because grafting variables is sufficient to generate all terms, but we could easily generalize to the grafting of an arbitrary term),
  - $t \in \mathbb{T}_n$ is non-degenerate,
  - $a \in (\mathsf{s}\, t)^\bullet$: ensures that a has not been used for grafting beforehand,
  - $\mathsf{s}\, a = \mathsf{s}\mathsf{s}\, x$: ensures that x may indeed be glued onto a by enforcing a certain globularity condition,
  - Γ and Δ are compatible: for all $y \in \mathbb{V}$, if $y \in \mathbb{V}_\Gamma \cap \mathbb{V}_\Delta$, the typing of y in both contexts match modulo the equational theory $E \cup F$; furthermore, the only variables typed in both Γ and Δ are a and the variables occurring in the sources of a:
$$\mathbb{V}_\Gamma \cap \mathbb{V}_\Delta = \{a\} \cup \bigcup_{1 \le i \le n-1} \left(\mathsf{s}^i\, a\right)^\bullet.$$
  The notations $t(a \leftarrow x)$ and $s_1[\mathsf{s}\, x/a]$ are presented below. We sometimes write `graft-`a to make explicit that we grafted onto a.

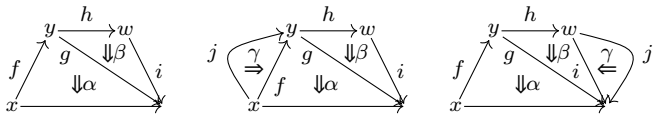
Definition 1 (Graft notation)**.** In the rule `graft`, we consider expressions of the form $t(x \leftarrow u)$, where t, u are non degenerate terms. According to the grammar (3), this is not a well-formed term in general, but it can be reduced to one by repeated use of the following rewrite rule: if $t = y(\overrightarrow{z_i \leftarrow v_i})$, then

$$t(x \leftarrow u) \quad \rightsquigarrow \quad \begin{cases} y(z_1 \leftarrow v_1(x \leftarrow u), z_2 \leftarrow v_2, \dots, z_n \leftarrow v_n) & \text{if } x \in (\mathsf{s}\, v_1)^\bullet \\ y(\overrightarrow{z_i \leftarrow v_i}, x \leftarrow u) & \text{if } x \in (\mathsf{s}\, y)^\bullet. \end{cases} \tag{4}$$

Example 2. Consider the term $t = \alpha(g \leftarrow \beta)$ in a suitable context Γ (see the left figure below). We have

$$t(f \leftarrow \gamma) = \alpha(f \leftarrow \gamma, g \leftarrow \beta) \qquad t(i \leftarrow \gamma) = \alpha(g \leftarrow \beta(i \leftarrow \gamma))$$

which can respectively be pictured as in the middle and right:



Definition 3 (Substitution)**.** The substitution $u[t/a]$, as used in the `graft` rule, is the term obtained by full reduction according to the following rule:

$$x(\overrightarrow{y_i \leftarrow u_i})[t/a] \quad \rightsquigarrow \quad \begin{cases} x(\dots, y_i \leftarrow u_i[t/a], \dots) & \text{if } a \in u_i^\bullet \\ t(\overrightarrow{y_i \leftarrow u_i}) & \text{if } x = a \end{cases} \tag{5}$$

and then, in the case $t = \underline{b}$, by performing the following actions:

- for subterms of the form $x(\dots, a \leftarrow \underline{b}, \dots)$ remove the grafting "$a \leftarrow \underline{b}$" and add $a = b$ to E,
- for subterms of the form $x(a \leftarrow \underline{b}(b \leftarrow u), \dots)$ replace the grafting "$a \leftarrow \underline{b}(b \leftarrow u)$" by "$a \leftarrow u$" and add $a = b$ to E,
- and finally replace any remaining subterms $\underline{b}(b \leftarrow u)$ by u.

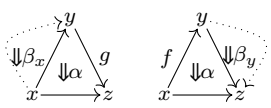
Note that the situation $\underline{b}[t/a]$ never occurs.

Example 4. Consider the term α corresponding to the picture on the right, as well as β_x and β_y:

$$\begin{array}{l} \Gamma \vdash \alpha : g(y \leftarrow f) \multimap x \multimap \varnothing \\ \Gamma \vdash \beta_x : \underline{x} \multimap x \multimap \varnothing \\ \Gamma \vdash \beta_y : \underline{y} \multimap y \multimap \varnothing \end{array}$$

Then the sources $\alpha(f \leftarrow \beta_x)$ and $\alpha(g \leftarrow \beta_y)$ are respectively

$$\begin{array}{l} g(y \leftarrow f)[\underline{x}/f] = g(y \leftarrow \underline{x}) = g \\ g(y \leftarrow f)[\underline{y}/g] = \underline{y}(y \leftarrow f) = f \end{array}$$



and, in the first case, the equation $x = y$ is added to E.

A uniqueness property of typing. Our system is convenient, because it displays a lot of information, but it is quite redundant, as we now show. We associate with a context Γ its "meager" version $\overline{\Gamma}$, obtained by replacing each $x : A$ with $x : \mathsf{s}\, x$ (i.e., by removing all but the top of the tower of

terms in A). In the theorem below, we show that the type of a term t and the context Γ are unique in the sense that they are determined by t and $\overline{\Gamma}$. The key observation is that we can read off the following equalities from the typing system:

$$\mathtt{s}\,\underline{x} = x \qquad \mathtt{s}\,(t(a{\leftarrow}x)) = (\mathtt{s}\,t)[\mathtt{s}\,x/a] \tag{6}$$

Theorem 5. *For $(E \triangleright \Gamma \vdash t : s_{n-1} \multimap s_{n-2} \multimap \cdots \multimap s_0 \multimap \varnothing)$ a derivable sequent, one can reconstruct uniquely Γ and $s_{n-1}, \dots s_0$ from $\overline{\Gamma}$ and t.*

Proof. This is an easy consequence of (6): the only information that is not (recursively) derivable is the source of variables, which is recorded in $\overline{\Gamma}$. □

Syntactic pasting schemes and opetopes. Among the derivable terms, we distinguish those reduced to a variable and call them *syntactic opetopes* (this terminology will be justified in V). Therefore, a context consists of a set of syntactic opetopes together with their (iterated) sources.

Notice that a derivable syntactic n-pasting scheme t induces a derivation of an $(n{+}1)$-dimensional variable α:

$$\frac{E \triangleright \Gamma \vdash_n t : A}{E \triangleright \Gamma, \alpha : t \multimap A \vdash_{n+1} \alpha : t \multimap A}\ \mathtt{shift}$$

This correspondence is in fact bijective, since the term can be recovered as $t = \mathtt{s}\,\alpha$. This allows us to formalize the correspondence between pasting schemes and opetopes as follows in our system:

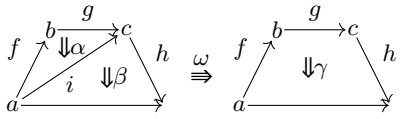
Lemma 6. *The above transformation induces a bijection between syntactic n-pasting schemes and syntactic $(n{+}1)$-opetopes.*

Let us point out that we (seemingly) leave an ambiguity as to whether to consider the variable α above as a syntactic $(n{+}1)$-opetope or as a term reduced to a variable, in which case it in fact stands for a syntactic $(n{+}1)$-pasting scheme corresponding by the bijection in the lemma to an $(n{+}2)$-opetope (an extrusion). It turns out that there is no harm in confusing these two interpretations of α, since we do distinguish between a syntactic pasting scheme and the syntactic opetope of which it is a source: the above bijection is not an identification!

III. Examples

In this section, we provide two examples illustrating our sequent calculus. We omit the contexts for concision since they can easily be constructed by aggregating every variable declarations in the proof tree.

Example 7. The 3-opetope



is derived as follows. First, α can be derived by

$$\dfrac{\dfrac{\dfrac{}{\vdash_0 b : \varnothing}\,\mathtt{point}}{\vdash_1 g : b \multimap \varnothing}\,\mathtt{shift} \qquad \dfrac{\dfrac{}{\vdash_0 a : \varnothing}\,\mathtt{point}}{\vdash_1 f : a \multimap \varnothing}\,\mathtt{shift}}{\dfrac{\vdash_1 g(b{\leftarrow}f) : b[a/b] \multimap \varnothing}{\vdash_2 \alpha : g(b{\leftarrow}f) \multimap a \multimap \varnothing}\,\mathtt{shift}}\,\mathtt{graft}\text{-}b$$

where $b[a/b] = a$, and similarly for β:

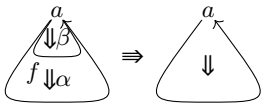
$$\dfrac{\dfrac{\dfrac{}{\vdash_0 c : \varnothing}\,\mathtt{point}}{\vdash_1 h : c \multimap \varnothing}\,\mathtt{shift} \qquad \dfrac{\dfrac{}{\vdash_0 a : \varnothing}\,\mathtt{point}}{\vdash_1 i : a \multimap \varnothing}\,\mathtt{shift}}{\dfrac{\vdash_1 h(c{\leftarrow}i) : c[a/c] \multimap \varnothing}{\vdash_2 \beta : h(c{\leftarrow}i) \multimap a \multimap \varnothing}\,\mathtt{shift}}\,\mathtt{graft}\text{-}c$$

where $c[a/c] = a$. Finally,

$$\dfrac{\dfrac{\vdots}{\vdash_2 \beta : h(c{\leftarrow}i) \multimap a \multimap \varnothing} \qquad \dfrac{\vdots}{\vdash_2 \alpha : g(b{\leftarrow}f) \multimap a \multimap \varnothing}}{\dfrac{\vdash_2 \beta(i{\leftarrow}\alpha) : h(c{\leftarrow}i)[g(b{\leftarrow}f)/i] \multimap a \multimap \varnothing}{\vdash_3 \omega : \beta(i{\leftarrow}\alpha) \multimap h(c{\leftarrow}g(b{\leftarrow}f)) \multimap a \multimap \varnothing}\,\mathtt{shift}}\,\mathtt{graft}\text{-}i$$

where the grafting on i is well defined since $\mathtt{s}\,i = a = \mathtt{ss}\,\alpha$, and $h(c{\leftarrow}i)[g(b{\leftarrow}f)/i] = h(c{\leftarrow}g(b{\leftarrow}f))$.

Example 8 (A degenerate case). The 3-opetope



is derived as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash a : \varnothing}\,\mathtt{point}}{\vdash f : a \multimap \varnothing}\,\mathtt{shift}}{\vdash \alpha : f \multimap a \multimap \varnothing}\,\mathtt{shift} \qquad \dfrac{\dfrac{}{\vdash a : \varnothing}\,\mathtt{point}}{\vdash \beta : \underline{a} \multimap a \multimap \varnothing}\,\mathtt{degen}}{\dfrac{\vdash \alpha(f{\leftarrow}\beta) : \underline{a} \multimap a \multimap \varnothing}{\vdash A : \alpha(f{\leftarrow}\beta) \multimap \underline{a} \multimap a \multimap \varnothing}\,\mathtt{shift}}\,\mathtt{graft}\text{-}f$$

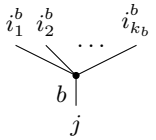
IV. Polynomial functors and opetopes

We will see in the next section that our definition of opetopes coincides with the traditional one based on polynomial functors. We quickly recall here this definition, referring the reader to [14], [15], [16] for a more detailed exposition.

Polynomial functors. A diagram of sets of the form

$$I \xleftarrow{\ s\ } E \xrightarrow{\ p\ } B \xrightarrow{\ t\ } J \tag{7}$$

may be interpreted as the specification of a collection of corollas with inputs *coloured* in I and output *coloured* in J. Namely, we can regard an element $b \in B$ as specifying a corolla with set of inputs (or *arity*) $p^{-1}(b) = \{e^b_1, \dots, e^b_{n_b}\}$. Each input e^b_k is assigned a colour $i^b_k = s(e^b_k)$ in I and the output has colour $j = t(b)$ in J:



By abuse of language, we call such a diagram a *polynomial functor* from I to J (to be precise, a polynomial functor would be a functor $\mathbf{Set}/I \to \mathbf{Set}/J$ induced in a certain way by such a diagram [8], [14]). These can be composed (the composite of P and Q has, as corollas, trees of depth 2, consisting of a corolla of Q at the bottom and corollas of P at the top). The unit for this composition is provided by the polynomial functor made of three identity functions. A morphism of polynomial

functors maps a corolla to a corolla, with a colour-preserving specified bijection between the inputs of the two corollas. We write $\mathbf{Poly}$ for the 2-category with sets as objects and polynomial functors and their morphisms as 1- and 2-cells. A *polynomial monad* is a monad internal to this 2-category.

We also need a more relaxed notion of morphism between polynomial endofunctors $P : I \to I$ and $Q : J \to J$, where now additionally a map from I to J is provided, and the specified bijections are required to be compatible with this map. This gives rise to a category $\mathbf{PolyEnd}$ whose objects are polynomial endofunctors and whose morphisms are such extended morphisms.

Trees. Following [15], we show how to recover rooted trees (with possibly open-ended leaf edges), simply called trees therafter, as special polynomial endofunctors.

A polynomial endofunctor of the form (7) with $I = J$ is a *tree*, with B and I as sets of *nodes* and *edges* respectively, when

- the sets I, B and E are finite,
- the map $t : B \to I$ is injective (it associates to a node its output edge),
- the map $s : E \to I$ is injective and the complement of its image consists of a single element, the *root edge* $\varepsilon^{\mid}$, and
- all nodes have the root node as iterated successor, where the *root node* ε is the node such that $t(\varepsilon) = \varepsilon^{\mid}$, and the *successor* function τ, defined as $\tau = p \circ s^{-1} \circ t$, associates to a node its parent node.

We write $T^{\bullet}$ (resp. $T^{\mid}$) for the set of nodes (resp. of edges) of a tree T and $T^{\uparrow} \subseteq T^{\mid}$ for the set of leaves (a *leaf* being an edge that is not in the image of t). Moreover, we always consider that s is an inclusion and write $\partial^{\bullet}$ (resp. $\partial^{\mid}$) instead of p (resp. t), so that a tree T is of the form

$$T^{\mid} \longleftarrow T^{\mid} \setminus \{\varepsilon^{\mid}\} \xrightarrow{\partial^{\bullet}} T^{\bullet} \xrightarrow{\partial^{\mid}} T^{\mid}$$

By convention (which is in tune with those used in our opetopic syntax), we shall use $x, y, \dots$ as generic names for either edges or nodes, indifferently.

We write $\mathbf{Tree}$ for the full subcategory of $\mathbf{PolyEnd}$ consisting of trees; its morphisms can be shown to be embeddings.

The free polynomial monad. Given a polynomial endofunctor P as in (7), we write $\mathrm{tr}(P)$ for the set of trees built from the corollas in P, which we call P-trees. A P-tree can be described as an isomorphism class of objects in the category $\mathbf{Tree}/P$, whose objects are pairs $(T, \ell : T \to P)$, where T is a tree and ℓ is a morphism in $\mathbf{PolyEnd}$. Unrolling the definition of (extended) morphism, the data of l can be decomposed in three parts:

- $\ell^{\bullet} : T^{\bullet} \to B$ labels the nodes of T with corollas of P,
- $\ell^{\mid} : T^{\mid} \setminus \{\varepsilon^{\mid}\} \to E$ provides for each node x of T a bijection between its set of incoming edges in T and the arity of $\ell^{\bullet}(x)$,
- and finally ℓ labels the root edge $\varepsilon^{\mid}$ of T with a colour $\ell^{c}(\varepsilon^{\mid})$ in I.

Note that these data induce in fact a colour labeling for all edges of T, setting $\ell^{c}(x) = s(\ell^{\mid}(x))$ for a non-root edge x. Finally, we require the following compatibility, for all $x \in T^{\bullet}$:

$$t(\ell^{\bullet}(x)) = \ell^{c}(\partial^{\mid} x). \tag{8}$$

In the sequel, we shall omit the superscripts on ℓ. For a node x of T, we shall also write $\mathsf{s}_x T$ for $\ell(x)$ (leaving the labeling function implicit). Similarly, we shall allow ourselves to abbreviate (T, ℓ) loosely as T, when the context makes it clear that we are speaking of a P-tree. We also use interchangeably the words "labeling", "decoration" and "colouring".

In more concrete terms, a P-tree T can be described recursively as follows:

- a colour i gives rise to a trivial P-tree without any node and with a single edge (being both the root and the unique leaf) decorated with i,
- if $T_1, \dots, T_n$ are P-trees whose root edges are decorated with $i_1, \dots, i_n$ respectively, if b is a corolla, and if $\kappa : \{1, \dots n\} \to p^{-1}(b)$ is a bijection such that

  $$i_j = s(\kappa(j))$$

  for all $j \in 1, \dots n$ (this is what (8) amounts to), then the tree formed by plugging each T_j on the input $\kappa(j)$ of corolla b is a P-tree.

In summary, *P-trees are trees whose nodes are labeled with corollas in B and whose edges are labeled with colours in I in a compatible way*.

We write $\mathrm{tr}^{\uparrow}(P)$ for the set of P-trees equipped with a distinguished leaf. It can be shown that the *free polynomial monad* P^{*} on a polynomial endofunctor P is given by

- the polynomial endofunctor

  $$I \longleftarrow \mathrm{tr}^{\uparrow}(P) \longrightarrow \mathrm{tr}(P) \longrightarrow I \tag{9}$$

  where the function in the middle is the obvious forgetful one, and the functions on the left and on the right return the colour of the distinguished leaf and of the root edge of the tree, respectively,
- equipped with a multiplication and unit [15], [16], where the multiplication is an unbiased version of grafting of trees (defined below) and the unit assigns to each i the tree reduced to a leaf, coloured by i, that we already encountered above.

Grafting. Fix a polynomial endofunctor P as in (7). Given $i \in I$, we write $\mathsf{I}_i \in \mathrm{tr}(P)$ for the unit at i (cf. above), and, given $b \in B$, Y_b for the P-tree consisting of only one corolla whose node is decorated with b:

$$\mathsf{I}_i = \{i\} \longleftarrow \varnothing \longrightarrow \varnothing \longrightarrow \{i\}$$

$$\mathsf{Y}_b = E_b + \{t(b)\} \longleftarrow E_b \longrightarrow \{b\} \longrightarrow E_b + \{t(b)\}$$

with $E_b = p^{-1}(B)$.

Remark 9. Note that with the above choices of sets, all nodes and edges of I_i and Y_b coincide wih their labels, i.e., their labeling functions are identities.

Now suppose given a P-tree S whose root edge is decorated with i and a P-tree T with a distinguished leaf x decorated with i. The *grafting* $T(\!(x{\leftarrow}S)\!)$ of S in T along x is defined as the following pushout (in $\mathbf{PolyEnd}$)

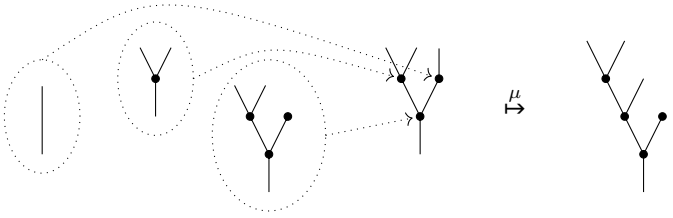
$$\begin{array}{ccc} \mathsf{l}_i & \xrightarrow{x} & T \\ \downarrow & \ulcorner & \downarrow \\ S & \longrightarrow & T(\!(x{\leftarrow}S)\!) \end{array}$$

which informally corresponds to taking the disjoint union of S and T and identifying the root of S with the leaf x of T. More generally, given a tree T with distinct distinguished leaves $x_1, \dots, x_k$ and trees $S_1, \dots, S_k$ whose roots are labeled as the corresponding leaves, one can define their simultaneous grafting $T(\!(x_1{\leftarrow}S_1, \dots, x_k{\leftarrow}S_k)\!)$ as the adequate colimit in $\mathbf{Tree}/P$. Every P-tree can be obtained by iteratively grafting corollas Y_b, starting from an edge l_i.

The $(-)^+$ construction. Given a polynomial endofunctor P as in (7), we write $\mathrm{tr}^\bullet(P)$ for the set of P-trees equipped with a distinguished node, i.e., morphisms $\mathsf{Y}_b \to T$ in $\mathbf{PolyEnd}$, with $b \in B$, and $T \in \mathrm{tr}(P)$. Given a polynomial monad P, the so-called *plus construction* [2], [16] associates to P a new polynomial monad P^+ defined as

$$B \xleftarrow{\mathsf{s}} \mathrm{tr}^\bullet(P) \xrightarrow{\mathsf{p}} \mathrm{tr}(P) \xrightarrow{\mathsf{t}} B$$

where $\mathsf{s}(\mathsf{Y}_b \to T) = b$, $\mathsf{p}(\mathsf{Y}_b \to T) = T$, and $\mathsf{t}(T)$ is given by the unique morphism from the free operad P^* to P. Concretely, $\mathsf{t}(T)$ is computed by viewing T as a program instructing to compose its nodes using the multiplication and the unit of the monad P. The multiplication of the monad P^+ can be described as follows. An element of $P^+ \circ P^+$ is a tree where every node of given arity n is decorated with a tree with n inputs (and matching colours). The multiplication $\mu : P^+ \circ P^+ \to P^+$ returns the tree obtained by substituting each node with its label. For instance,



Opetopes. Consider the identity polynomial endofunctor Z^0, where all sets are singletons:

$$Z^0 = \; 1 \longleftarrow 1 \longrightarrow 1 \xrightarrow{\mathsf{t}} 1$$

It is a polynomial monad, so we may define $Z^{n+1} := (Z^n)^+$, for $n \in \mathbb{N}$. We set, for all $n \geq 0$

$$Z^n = \; \mathbb{O}_n \longleftarrow \mathbb{O}^\bullet_{n+1} \longrightarrow \mathbb{O}_{n+1} \xrightarrow{\mathsf{t}} \mathbb{O}_n$$

with

- $\mathbb{O}_0, \mathbb{O}_1, \mathbb{O}^\bullet_1$ singletons,
- $\mathbb{O}_{n+1} = \mathrm{tr}(Z^{n-1})$, $\mathbb{O}^\bullet_{n+1} = \mathrm{tr}^\bullet(Z^{n-1})$, for $n \geq 1$.

Definition 10. The set of *n-opetopes* is $\mathbb{O}_n$.

In the polynomial setting, an $(n{+}2)$-opetope is thus a tree whose nodes (resp. edges) are decorated with $(n{+}1)$-opetopes (resp. n-opetopes). We write $\blacklozenge$ (resp. $\blacksquare$) for the unique 0-opetope (resp. 1-opetope).

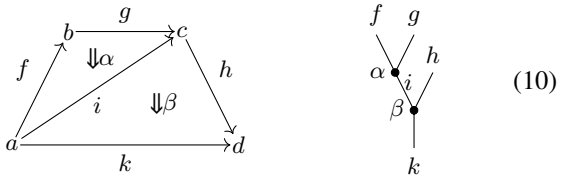
The $(-)^+$ construction induces, for every opetope $\omega \in \mathbb{O}_n$ with $n \geq 1$, a bijection $\wp_\omega : \omega^\uparrow \to (\mathsf{t}\,\omega)^\bullet$, called *readdressing*, between the leaves of ω and the nodes of its target. Also, note that, given an $(n{+}1)$-opetope ω (i.e., a labeled tree) and an edge x of ω which is not the root edge, the label $\ell(x)$ of x is an element of $\mathrm{tr}^\bullet(Z^n)$, which should be seen as the corresponding node of the opetope decorating $\partial^\bullet x$.

Remark 11. In fact, we could have started our construction from

$$Z^{-1} = \; 1 \longleftarrow 0 \longrightarrow 1 \longrightarrow 1$$

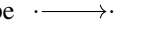
Indeed, we have $Z^0 = (Z^{-1})^*$.

Example 12. The pasting scheme corresponding to the opetope of 7 is recalled on the left and, on the right, we give its representation as a tree



(10)

In this tree,

- the nodes α and β are labeled by the opetope 
- the edges f to k are labeled by the opetope 

Moreover, the pasting scheme of the target of the opetope and the corresponding tree are

(11)

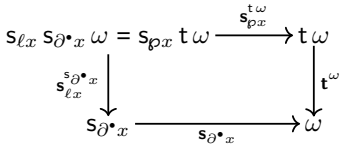
As explained above, the leaves (f, g and h) of the tree (10), are in bijection with the nodes of the tree (11) of the target.

The category of opetopes. Note that for each opetope $\omega \in \mathbb{O}_{n+1}$ and node x of ω, we have $\mathsf{s}_x\,\omega \in \mathbb{O}_n$ and $\mathsf{t}\,\omega \in \mathbb{O}_n$. Working out the relations satisfied by those morphisms, see [12], leads us to define the *category of opetopes* $\mathbb{O}$ as the category whose objects are the opetopes (of any dimension), and whose morphisms are generated by

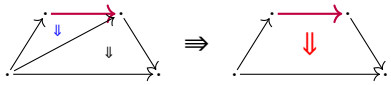
$$\mathbf{s}^\omega_x : \mathsf{s}_x\,\omega \to \omega \qquad\qquad \mathbf{t}^\omega : \mathsf{t}\,\omega \to \omega$$

indexed by $\omega \in \bigsqcup_{n\in\mathbb{N}} \mathbb{O}_n$ and $x \in \omega^\bullet$ (we sometimes omit the superscript ω in the following), and subject to the following "globularity" relations, called here *opetopic identities*:

1) for $\omega \in \mathbb{O}$ and $x \in \omega^\uparrow$,



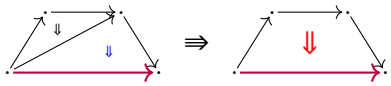
recall that $\wp$ establishes a bijection between the leaves of ω and the nodes of $\mathsf{t}\omega$; this diagram expresses the fact that $\wp$ preserves the decoration :



2) for $\omega \in \mathbb{O}$,

$$\begin{array}{ccc} \mathsf{t}\,\mathsf{t}\omega = \mathsf{t}\,\mathsf{s}_\varepsilon\,\omega & \xrightarrow{\mathsf{t}^{\mathsf{s}_\varepsilon\,\omega}} & \mathsf{s}_\varepsilon\,\omega \\ {\scriptstyle \mathsf{t}^{\mathsf{t}\omega}}\downarrow & & \downarrow{\scriptstyle \mathsf{s}^{\omega}_{\varepsilon}} \\ \mathsf{t}\,\omega & \xrightarrow[\mathsf{t}^{\omega}]{} & \omega \end{array}$$

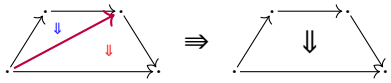
aking to the previous identity, the latter expresses that the decoration of the root edge of ω matches $\mathsf{t}\omega$:



3) for $\omega \in \mathbb{O}$ and $x \in \omega^\bullet$,

$$\begin{array}{ccc} \mathsf{t}\,\mathsf{s}_x\,\omega = \mathsf{s}_{\ell(\partial^{\mathsf{l}}x)}\,\mathsf{s}_{\tau(x)}\,\omega & \xrightarrow{\mathsf{s}^{\mathsf{s}_{\tau(x)}\,\omega}_{\ell(\partial^{\mathsf{l}}x)}} & \mathsf{s}_{\tau(x)} \\ {\scriptstyle \mathsf{t}^{\mathsf{s}_x\,\omega}}\downarrow & & \downarrow{\scriptstyle \mathsf{s}^{\omega}_{\tau(x)}} \\ \mathsf{s}_x\,\omega & \xrightarrow[\mathsf{s}^{\omega}_{x}]{} & \omega \end{array}$$

which means that two nodes (here x and $\tau(x)$) sharing an edge should agree on the decoration of that edge ; this is depicted in the following pasting diagram :



4) for ω degenerate,

$$\begin{array}{ccc} \mathsf{t}\,\mathsf{t}\omega = \mathsf{s}_\varepsilon\,\mathsf{t}\omega & \xrightarrow{\mathsf{s}^{\mathsf{t}\omega}_{\varepsilon}} & \mathsf{t}\omega \\ {\scriptstyle \mathsf{t}^{\mathsf{t}\omega}}\downarrow & & \downarrow{\scriptstyle \mathsf{t}^{\omega}} \\ \mathsf{t}\,\omega & \xrightarrow[\mathsf{t}^{\omega}]{} & \omega \end{array}$$

this last identity expressed the fact that the target of degenerate opetopes are "loops" :

The relations can be illustrated on the opetope ω of 7 (see also 12) as follows:

1) with $x = g \in \omega^{\uparrow}$, we have

$$\mathsf{s}_{\ell g}\,\mathsf{s}_{\partial^\bullet g}\,\omega = \mathsf{s}_{\ell g}\,\mathsf{s}_\alpha\,\omega = \mathsf{s}_{\ell g}\,\alpha = g = \mathsf{s}_{\wp g}\,\gamma = \mathsf{s}_{\wp g}\,\mathsf{t}\,\omega$$

2) we have

$$\mathsf{t}\,\mathsf{t}\omega = \mathsf{t}\,\gamma = k = \mathsf{t}\,\beta = \mathsf{t}\,\mathsf{s}_\varepsilon\,\omega$$

3) with $x = \alpha$, we have

$$\mathsf{t}\,\mathsf{s}_\alpha\,\omega = \mathsf{t}\,\alpha = i = \mathsf{s}_i\,\beta = \mathsf{s}_{\ell(\partial^{\mathsf{l}}\alpha)}\,\mathsf{s}_{\tau(\alpha)}\,\omega$$

4) for the following degenerate opetope ω

$$a \xrightarrow{f} b \quad \overset{\omega}{\Rrightarrow} \quad a \underset{f}{\overset{f}{\Downarrow \alpha}} b$$

we have

$$\mathsf{t}\,\mathsf{t}\omega = \mathsf{t}\,\alpha = f = \mathsf{s}_\varepsilon\,\alpha = \mathsf{s}_\varepsilon\,\mathsf{t}\,\omega\,.$$

V. Equivalence with polynomial opetopes

In this section, all sequents are assumed derivable in Opt. We show that syntactic opetopes, considered up to α-conversion, are in bijective correspondence with polynomial opetopes (10), by constructing an explicit bijection $[\![-]\!]^{\mathbb{O}}_n$ from syntactic n-opetopes up to α-conversion to n-opetopes, with inverse $[\![-]\!]^{\vdash}_n$.

Polynomial coding. We give an inductive construction mapping a syntactic n-pasting scheme, i.e., a sequent typing a term $(E \triangleright \Gamma \vdash_n t : T)$, to an $(n{+}1)$-opetope $[\![t]\!]^{\mathbb{O}}_{n+1}$ (leaving the rest of the sequent implicit), that we call its *polynomial coding*. The construction maintains the following invariant: all the names of nodes and edges of all the involved underlying trees of the opetopes that are recursively constructed are picked from $\mathbb{V}_\Gamma$ (modulo the equivalence relation E) in a dimension-respecting way. The inductive definition is as follows:

$$[\![\varnothing]\!]^{\mathbb{O}}_0 := \blacklozenge \tag{12}$$

$$[\![x]\!]^{\mathbb{O}}_1 := \blacksquare \tag{13}$$

$$[\![\underline{x}]\!]^{\mathbb{O}}_{n+1} := \mathsf{I}_{[\![\mathsf{s}\,x]\!]^{\mathbb{O}}_{n-1}}, \tag{14}$$

$$[\![x(\overrightarrow{y_i \leftarrow u_i})]\!]^{\mathbb{O}}_{n+1} := \mathsf{Y}_{[\![\mathsf{s}\,x]\!]^{\mathbb{O}}_n}(\!(\overrightarrow{y_i \leftarrow [\![u_i]\!]^{\mathbb{O}}_{n+1}})\!). \tag{15}$$

Note that in (15), y_i is indeed a node of $[\![\mathsf{s}\,x]\!]^{\mathbb{O}}_n$ by the invariant, and hence a leaf of $\mathsf{Y}_{[\![\mathsf{s}\,x]\!]^{\mathbb{O}}_n}$ by remark 9.

It is clear that the coding function is well defined in 14. The following proposition ensures that this is also the case for (15):

Proposition 13. *With the notations of 15, for every index i, we have* $\mathsf{t}\,\mathsf{s}_\varepsilon\,[\![u_i]\!]^{\mathbb{O}}_{n+1} = \mathsf{s}_{y_i}\,\mathsf{s}_\varepsilon\,[\![x]\!]^{\mathbb{O}}_{n+1}$.

Polynomial decoding. Conversely, to every n-opetope ω, we associate a syntactic $(n{-}1)$-pasting scheme $[\![\omega]\!]^{\vdash}$ defined by induction, using `point`, `shift`, `degen`, and `graft`, respectively:

$$\frac{}{[\![\blacklozenge]\!]^{\vdash}} \qquad \frac{[\![\omega]\!]^{\vdash}}{[\![\mathsf{Y}_\omega]\!]^{\vdash}} \qquad \frac{[\![\omega]\!]^{\vdash}}{[\![\mathsf{I}_\omega]\!]^{\vdash}} \qquad \frac{[\![\omega]\!]^{\vdash} \qquad [\![\tau]\!]^{\vdash}}{[\![\omega(\!(a \leftarrow \mathsf{Y}_\tau)\!)]\!]^{\vdash}} \tag{16}$$

All along this induction, names are introduced (rules `point` and `shift`), and α-conversions have to be performed to fulfill the side-conditions of `graft`.

For ω non-degenerate, it can be shown that this assignment of a pasting scheme is independent (up to α-conversion) of the choice of decomposition of ω as a sequence of graftings, and that this thus defines an interpretation $[\![\omega]\!]^{\vdash}$ for every opetope ω which is unique (up to α-conversion), so that we can finally conclude:

Theorem 14. *The functions $[\![-]\!]^{\mathbb{O}}$ and $[\![-]\!]^{\vdash}$ induce a bijection between syntactic $(n-1)$-pasting schemes (or, equivalently, syntactic n-opetopes) modulo α-conversion and n-opetopes.*

VI. A SEQUENT CALCULUS FOR OPETOPIC SETS

We now present OPTSET, a sequent calculus for opetopic sets in the sense that derivable terms (or, more precisely, contexts) correspond to finite opetopic sets. This system is based on OPT in the sense that it starts from sequents in this last system. This is however not essential and a more homogeneous variant is studied in VII. We recall that opetopic sets are the object of the category $\hat{\mathbb{O}}$ of presheaves over the category $\mathbb{O}$.

Syntax. One of the main distinguishing features of the system OPT is that only source faces (as opposed to target ones) of whichever cell is currently being derived are specified. Nonetheless, all the information about targets remains. To adapt our previous calculus to opetopic sets, all faces, including targets, need to be explicitly specified.

Sequents. An opetopic set will be specified by a context Γ together with an equational theory E. We thus consider sequents of the form

$$E \triangleright \Gamma$$

which we call an *opetopic context modulo theory*, or *OCMT*.

Inference rules. It is well known that the Yoneda embedding $O : \mathbb{O} \to \hat{\mathbb{O}}$ exhibits $\hat{\mathbb{O}}$ as a free completion of $\mathbb{O}$ under colimits [18, section I.5], and the restriction to finite presheaves corresponds to completing under finite colimits, as every slice of $\mathbb{O}$ is finite. It is also well known that those colimits can be obtained from the initial object, binary sums, and coequalizers, so our strategy is to add rules syntactically ensuring their existence. We show in 19 that terms in the resulting system actually correspond to finite opetopic sets.

The rules of the system OPTSET are as follows.

- *Introduction of all targets*: takes an opetope $(E \triangleright \Gamma \vdash_n x : X)$ derivable in OPT, and completes it by adding all the "missing target cells" making it into an OCMT (more abstractly, this corresponds to computing the representable opetopic set associated to an opetope).

$$\frac{E \triangleright \Gamma \vdash_n x : X \qquad x \in \mathbb{V}_n}{E' \triangleright \Gamma'}\,\texttt{repr}$$

  where Γ' is obtained from Γ by adding a new variable

$$x^{\mathtt{t}^k\, a} : \mathtt{s}^{k+1}\, a \multimap \mathtt{s}^{k+2}\, a \multimap \cdots$$

  for every $a \in \mathbb{V}_{\Gamma,m}$ and $1 \le k \le m \le n$ (in the following, we write $\mathtt{t}^k\, a$ instead of $x^{\mathtt{t}^k\, a}$ and sometimes $\approx_E$ instead of E' and $\hat{x}$ instead of Γ'), and E' is obtained from E by adding
  - $\mathtt{t}\, a = b$ for all $b \leftarrow a(\cdots)$ occuring in a type in Γ,
  - $\mathtt{t}\,\mathtt{t}\, a = \mathtt{t}\mathtt{s}_\varepsilon\, \mathtt{s}\, a$ for all $a \in \mathbb{V}_{\Gamma',k}$ with non-degenerated source with $2 \le k \le n$,
  - $\mathtt{t}^{k+2}\, a = b$ if $\mathtt{t}^k\, a : \underline{b} \multimap b \multimap \cdots$ with $0 \le k \le n-2$.
- *Zero*: introduces the empty OCMT.

$$\frac{}{\triangleright}\,\texttt{zero}$$

- *Binary sums*: takes two disjoint opetopic sets (i.e., whose cells have different names), and produces their sum.

$$\frac{E \triangleright \Gamma \qquad F \triangleright \Delta}{E \sqcup F \triangleright \Gamma \sqcup \Delta}\,\texttt{sum} \tag{17}$$

  where we suppose $\Gamma \cap \Delta = \varnothing$.
- *Quotients*: identifies two parallel cells in an opetopic set by extending the underlying equational theory:

$$\frac{E \triangleright \Gamma}{E \sqcup \{a = b\} \triangleright \Gamma}\,\texttt{glue} \tag{18}$$

  where we suppose $a, b \in \mathbb{V}_\Gamma$, $\mathtt{s}\, a =_E \mathtt{s}\, b$ and $\mathtt{t}\, a =_E \mathtt{t}\, b$. We sometimes write $\texttt{glue-}(a{=}b)$ to make explicit that we added $\{a = b\}$ to the theory.

Equivalence with finite opetopic sets. We now show that the OCMTs are in bijection with finite opetopic sets. As in V, this is done by providing explicit inverse bijections.

From OCMT to opetopic sets. Given an OCMT $E \triangleright \Gamma$, we write Γ/E for the set $\mathbb{V}_\Gamma$ quotiented by the equivalence relation generated by E. We will see that this set can be equipped with the structure of an opetopic set. In order to do so, our first task is to show that this is the case for representable OCMTs, i.e., whose derivation ends with

$$\frac{E \triangleright \Gamma \vdash_n x : X}{\approx_x \triangleright \hat{x}}\,\texttt{repr}$$

which is achieved in 16 below, by induction. We first need a technical lemma. Given a sequent $(E \triangleright \Gamma \vdash_n x : X)$ and $(a : A) \in \hat{x}$, the *a-restriction* of the sequent is the sequent $E|_a \triangleright \Gamma|_a \vdash_k a : A$ where $E|_a$ (resp. $\Gamma|_a$) is the restriction of E (resp. Γ) to variables and terms whose variables belong to the variables of A.

Lemma 15. *Given a derivable sequent in* OPT *and a variable a of its context, its a-restriction is also derivable.*

In a situation as in the previous lemma, we have, by induction, a well-defined opetope. This induces a map $[\![-]\!]^{\mathbb{O}} : \mathbb{V}_{\hat{x}} \to \mathbb{O}$, which can be shown to factor through $\hat{x}/{\approx_x}$. We now construct source and target maps between the fibers of this map. Suppose that a is of dimension k.

- *Sources*: given $b \in ([\![a]\!]^{\mathbb{O}}_k)^\bullet$ then, by construction, see (15), there is a unique variable in $\mathbb{V}_{\hat{x}|_a, k-1}$ up to the ambient equational theory, which corresponds (under the bijection) to b, which we still write b, and we define $\mathtt{s}_b\, a = b$.
- *Target*: for $a \in \mathbb{V}_{\hat{x},k}$, we define $\mathbf{t}\, a = \mathtt{t}\, a$, the latter being the variable introduced by the `repr` rule.

Theorem 16. *With the structure maps introduced above, $\hat{x}/{\approx_x}$ is an opetopic set, which is isomorphic to the representable $O([\![x]\!]^{\mathbb{O}})$.*

Proof. The fact that $\hat{x}/{\approx_x}$ is an opetopic set can be checked by directly verifying that the relations for source and target, induced by those of the definition of $\mathbb{O}$, are satisfied. The

opetopic set $\hat{x}/\approx_x$ clearly has one maximal element and is thus, by the Yoneda lemma, a quotient of the associated representable $O([\![x]\!]^{\mathbb{O}})$. Finally, the two can be shown to be isomorphic by proving by induction that they have the same number of cells in every dimension. □

By induction, we can now associate with every derivable OCMT $E \triangleright \Gamma$ in OPTSET an opetopic set noted

$$\Gamma/E \tag{19}$$

(equipped with suitable source and target maps) together with, for every variable $a \in \mathbb{V}_\Gamma$, a map

$$\tilde{a} : O([\![a]\!]^{\mathbb{O}}) \to \Gamma/E$$

defined as follows, depending on the last rule used in the derivation of the OCMT:

- `repr`: as described above,
- `zero`: then Γ/E is the initial opetopic set,
- `sum`: with the notations of (17), $(\Gamma \sqcup \Delta)/(E \sqcup F)$ is the coproduct $\Gamma/E \sqcup \Delta/F$,
- `glue`: with the notations of (18), $\Gamma/(E \sqcup \{a = b\})$ is the opetopic set defined as the coequlizer

$$O([\![a]\!]^{\mathbb{O}}) = O([\![b]\!]^{\mathbb{O}}) \underset{\tilde{b}}{\overset{\tilde{a}}{\rightrightarrows}} \Gamma/E \twoheadrightarrow \Gamma/(E \sqcup \{a = b\})$$

It is easy to see that for every proof tree of $E \triangleright \Gamma$, the opetopic set Γ/E is

$$\Gamma/E \cong \frac{\bigsqcup_{a \in \mathbb{V}_\Gamma} O([\![a]\!]^{\mathbb{O}}_k)}{E}$$

(where, as explained before, on the right, we leave implicit the bijection between the variables of Γ and the cells of the associated opetopic set). Thus,

Lemma 17. *For $(E \triangleright \Gamma)$ a derivable OCMT in* OPTSET*, the structure of opetopic set on Γ/E does not depend on the proof tree of $E \triangleright \Gamma$.*

Equivalence. The *syntactic category* of OPTSET is the category $\mathbf{Ctx}$ whose objects are the sequents and a morphism

$$f : E \triangleright \Gamma \to F \triangleright \Delta$$

is a *substitution*, i.e., , a function $f : \mathbb{V}_\Delta \to \mathbb{V}_\Gamma$, which

- respects the dimension of variables,
- respects equality: for $x, y \in \mathbb{V}_\Delta$, $x =_F y$ implies $f(x) =_E f(y)$,
- respects typing: for a typing $(x : A)$, $f(x)$ has type $A' =_F f(A)$ in Δ, where f is extended as a morphism on terms and types in the expected way.

We write $\hat{\mathbb{O}}_{\text{fin}}$ for the full subcategory of $\hat{\mathbb{O}}$ whose objects are finite presheaves X, i.e., such that the set $\bigsqcup_{\omega \in \mathbb{O}} X_\omega$ is finite. Our aim is to show that we have an equivalence of categories $\mathbf{Ctx}^{\text{op}} \cong \hat{\mathbb{O}}_{\text{fin}}$. We begin by noticing that morphisms preserve representable opetopes:

Lemma 18. *Let $f : (E \triangleright \Gamma) \to (F \triangleright \Delta)$ be a morphism of OCMT, and $a \in \mathbb{V}_{\Gamma,k}$. Then $[\![a]\!]^{\mathbb{O}}_k = [\![f(a)]\!]^{\mathbb{O}}_k$.*

The *stratification functor* $S : \mathbf{Ctx}^{\text{op}} \longrightarrow \hat{\mathbb{O}}_{\text{fin}}$ is defined on objects by $S(E \triangleright \Gamma) = \Gamma/E$ and on morphisms by $S(f) = f^{\text{op}}$.

Theorem 19. *The stratification functor $S : \mathbf{Ctx}^{\text{op}} \to \hat{\mathbb{O}}_{\text{fin}}$ is an equivalence of categories.*

Proof. An object in the image of S is a finite opetopic set of the form Γ/E for some OCMT $(E \triangleright \Gamma)$. It contains all representables (by 16) as well as the initial object, and it is closed under finite sums and quotients (by definition (19)). It is thus essentially surjective on objects since every finite opetopic set is a finite colimit of representables. By definition, S is faithful. It remains to show that it is full.

Given a morphism of finite opetopic sets $f : S(E \triangleright \Gamma) \to S(F \triangleright \Delta)$, we define a morphism of OCMT $\tilde{f} : F \triangleright \Delta \to E \triangleright \Gamma$ by fixing $\tilde{f}(x)$ to be a chosen element of the equivalence class $f(x)$ for any $x \in \mathbb{V}_\Delta$. We show that this morphism respects the typing by induction on the dimension k of the variable x of type X. The result is immediate for $k = 0$ since in this case $X = \varnothing$ and $f(X) = \varnothing = f(\varnothing)$. Otherwise, the type X of x is $\mathsf{s}\,x \multimap \mathsf{ss}\,x \multimap \ldots \multimap \varnothing$ and the type Y of $\mathsf{t}\,x$ in Γ is $\mathsf{ss}\,x \multimap \ldots \multimap \varnothing$. By induction, the type of $f(\mathsf{t}\,x)$ in Δ is $f(Y)$, and since $f(\mathsf{t}\,x) = \mathsf{t}\,f(x) : \mathsf{ss}\,f(x) \multimap \cdots \multimap \varnothing$, we have

$$(f(\mathsf{ss}\,x) \multimap \cdots \multimap \varnothing) = f(Y) =_F (\mathsf{ss}\,f(x) \multimap \cdots \multimap \varnothing),$$

or in other words, $\mathsf{s}^i f(x) =_F f(\mathsf{s}^i x)$, for $2 \le i \le k$. It remains to show that the latter formula holds in the case $i = 1$ (the case $i = 0$ is tautological). Towards a contradiction, assume $\mathsf{s}\,f(x) \neq_F f(\mathsf{s}\,x)$. Then there exists $y \in ([\![x]\!]^{\mathbb{O}}_k)^\bullet = ([\![f(x)]\!]^{\mathbb{O}}_k)^\bullet$ such that $\mathsf{s}_y f(x) \neq_F f(\mathsf{s}_y x)$, which contradicts the fact that f is a morphism of opetopic sets. Consequently, $\mathsf{s}\,f(x) =_F f(\mathsf{s}\,x)$, and $f(X)$ is the type of $f(x)$ in Δ modulo F. We can conclude that S is full and thus an equivalence of categories. □

Models. As traditionally, a *model* of the type theory OPTSET is a functor $\mathbf{Ctx} \to \mathbf{Set}$ which preserves finite limits, and we write $\mathbf{Mod}(\mathbf{Ctx})$ for the resulting category, with natural transformations as morphisms. Here, models are precisely opetopic sets:

Theorem 20. *We have an equivalence of categories $\hat{\mathbb{O}} \cong \mathbf{Mod}(\mathbf{Ctx})$.*

Proof. We apply Gabriel-Ulmer duality [1]: the category $\hat{\mathbb{O}}$ is a category of presheaves over a locally small category and thus a locally presentable category, and finite opetopic sets are the finitely presentable objects, thus $\mathbf{Mod}(\mathbf{Ctx}) \cong \mathbf{Mod}(\hat{\mathbb{O}}^{\text{op}}_{\text{fin}}) \cong \hat{\mathbb{O}}$. □

VII. THE MIXED SYSTEM FOR OPETOPIC SETS

The OPTSET system, presented in VI, suffers from the following drawback: derivations of opetopic sets using the rule `repr` require, as a precondition, a derivation in the system OPT of II. This makes derivations somewhat unintuitive. Indeed, every finite opetopic set X can be written as

$$X = \frac{\bigsqcup_i O(\omega_i)}{\sim}$$

where $\sim$ represents some quotient. Thus to derive X in OPTSET, the representables $O(\omega_i)$ have to be derived in OPT

first, and only after the `repr` rule has been used on each, can the sums and gluings be performed. In this section, we present system OPTSET$_{\text{M}}$ (the m standing for "mixed") for opetopic sets, which does not depend on OPT, and that allows introducing new cells, making disjoint unions, and gluing cells in any sound order.

Syntax. The syntax of system OPTSET$_{\text{M}}$ relies on sequents from OPT (see II) and OCMTs from OPTSET (see VI). Specifically, we have two types of judgments:

- $E \triangleright \Gamma$, stating that $(E \triangleright \Gamma)$ is a well formed OCMT,
- $E \triangleright \Gamma \vdash t : T$, stating that in OCMT $(E \triangleright \Gamma)$, the term t is well formed, and has type T; we may also write $E \triangleright \Gamma \vdash_n t : T$ if $t \in \mathbb{T}_n$.

Inference rules. The inference rules of the sequent calculus OPTSET$_{\text{M}}$ are as follows.

- *Introduction of points*: introduces 0-cells.
$$\frac{x \in \mathbb{V}_0}{\triangleright x : \varnothing}\ \texttt{point}$$
- *Introduction of degenerate pasting diagrams*: creates a new degenerate pasting diagram.
$$\frac{E \triangleright \Gamma, x : X}{E \triangleright \Gamma, x : X \vdash_k \underline{x} : x \multimap X}\ \texttt{degen}$$
- *Introduction of non-degenerate pasting diagrams*: creates a new non-degenerate pasting diagram consisting of a single cell.
$$\frac{E \triangleright \Gamma, x : X}{E \triangleright \Gamma, x : X \vdash_k x : X}\ \texttt{pd}$$
- *Grafting*: extends a previously derived non-degenerate pasting diagram by grafting a cell.
$$\frac{\begin{array}{c} E \triangleright \Gamma \vdash_n t : s_1 \multimap s_2 \multimap \cdots \\ F \triangleright \Delta \vdash_n x : X \end{array}}{G \triangleright \Gamma \cup \Delta \vdash_n t(a \leftarrow x) : s_1[\mathtt{s}\, x/a] \multimap s_2 \multimap \cdots}\ \texttt{graft}$$
where we suppose that the same side conditions as for rule `graft` of system OPT (see II) are satisfied. Above, G is the union of E, F, and potentially additional equalities incurred by the substitution $s_1[\mathtt{s}\, x/a]$.
- *Shift to the next dimension*: takes a previously derived pasting diagram (degenerate or not), and introduces a new cell having this pasting diagram as source. It also introduces all its targets, and extends the ambient equational theory with the required identities.
$$\frac{E \triangleright \Gamma \vdash_n t : T \qquad x \in \mathbb{V}_{n+1}}{E' \triangleright \Gamma'}\ \texttt{shift}$$
with $x \notin \mathbb{V}_\Gamma$ and where Γ' is obtained from Γ by adding fresh variables
$$x^{\mathtt{t}^k\, x} : s^{k+1} x \multimap s^{k+2} x \multimap \cdots$$
for $0 \le k \le n$ (by convention $x^{\mathtt{t}^0\, x}$ is x and we write $\mathtt{t}^k\, x$ instead of $x^{\mathtt{t}^k\, x}$ in the following) similarly to rule `repr` of VI and E' is obtained from E by adding $\mathtt{t}^{k+2}\, x = \mathtt{t}^k\, x$ for $0 \le k < n-1$ if t is degenerate, say $t = y(\overrightarrow{z_i \leftarrow u_i})$, E' is obtained from E by adding $\mathtt{t}^2\, x = \mathtt{t}\, x$ (if $n > 0$) and $a = b$ for every $b \leftarrow a(\cdots)$ occurring in t.
- *Zero*: introduces the empty OCMT.
$$\frac{}{\triangleright}\ \texttt{zero}$$
- *Binary sums*: takes two disjoint OCMTs, and produces their sum.
$$\frac{E \triangleright \Gamma \qquad F \triangleright \Delta}{E \sqcup F \triangleright \Gamma \sqcup \Delta}\ \texttt{sum}$$
where we suppose $\Gamma \cap \Delta = \varnothing$.
- *Quotients*: identifies two parallel cells in an opetopic set by extending the underlying equational theory.
$$\frac{E \triangleright \Gamma}{E \sqcup \{a = b\} \triangleright \Gamma}\ \texttt{glue}$$
where we suppose $a, b \in \mathbb{V}_\Gamma$, $\mathtt{s}\, a =_E \mathtt{s}\, b$ and $\mathtt{t}\, a =_E \mathtt{t}\, b$.

Equivalence with opetopic sets. We prove in 23 that the sets of OCMTs derivable in OPTSET$_{\text{M}}$ and OPTSET are the same. This is done by rewriting proof trees in OPTSET to proof trees in OPTSET$_{\text{M}}$ (21) and conversely (22). In the proofs below, we shall decorate the rules of the systems OPT and OPTSET by a prime (e.g. `shift'`), in order to differentiate them from the rules of system OPTSET$_{\text{M}}$.

Proposition 21. *Every OCMT derivable in system* OPTSET *is also derivable in system* OPTSET$_{\text{M}}$.

Proof. A proof in OPTSET begins with derivations in OPT followed by rules `repr'` followed by a derivation in system OPTSET. Since the rule `glue'` is exactly the rule `glue` and likewise for `sum`, it is enough to show that a rule of OPT followed by `repr'` can be rewritten as `repr'` followed by a rule in OPTSET, which can be done by case analysis, thus "pushing the `repr'` rules towards the leaves". □

Proposition 22. *Every OCMT derivable in system* OPTSET$_{\text{M}}$ *is also derivable in system* OPTSET.

Proof. The idea is to perform the converse of the proof of previous proposition. Starting from a proof in OPTSET$_{\text{M}}$, we can locally permute rules in order to obtain a proof consisting of rules `point` or `degen` or `pd` or `graft` or `shift` only followed by rules `glue` and `sum` only. Then, starting from the leaves, we introduce instances of `repr'`, and by "pushing them down", i.e. by using the inverse rewriting rules of 21, we rewrite the proof tree to one in system OPTSET. □

We can thus conclude:

Theorem 23. *The two systems* OPTSET$_{\text{M}}$ *and* OPTSET *are equivalent.*

Example. We now derive the opetopic set (2) in system OPTSET$_{\text{M}}$. For clarity (or due to space reasons), we sometimes omit the type (or part of the type) of variables. First, the cell f is derived as follows:
$$\dfrac{\dfrac{\dfrac{}{\triangleright a : \varnothing}\ \texttt{point}}{\triangleright a \vdash_0 a}\ \texttt{pd}}{\triangleright a, \mathtt{t}\, f : \varnothing, f : a \multimap \varnothing}\ \texttt{shift}$$
Note how the `shift` rule instance not only introduces f, but also its target $\mathtt{t}\, f$. The introduction of point b and its

identification with $\mathtt{t}\,f$ will happen later in the proof tree. The cell g is derived from there in a similar fashion:

$$\dfrac{\dfrac{\vdots}{\dfrac{\triangleright a, \mathtt{t}\,f, f}{\triangleright a, \mathtt{t}\,f, f \vdash_0 a}\,\mathtt{pd}}}{\triangleright a, \mathtt{t}\,f, \mathtt{t}\,g : \varnothing, f, g : a \multimap \varnothing}\,\mathtt{shift}$$

We could also derive h from here, but decide to do so in a different branch:

$$\dfrac{\dfrac{\dfrac{}{\triangleright b : \varnothing}\,\mathtt{point}}{\triangleright b \vdash_0 b}\,\mathtt{pd}}{\triangleright b, \mathtt{t}\,h : \varnothing, h : b \multimap \varnothing}\,\mathtt{shift}$$

Next, we join those two proof trees by using rule `sum`, and identify b with $\mathtt{t}\,f$ and $\mathtt{t}\,g$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots \quad \vdots}{\dfrac{\triangleright a, \mathtt{t}\,f, \mathtt{t}\,g, f, g \qquad \triangleright b, \mathtt{t}\,h, h}{\triangleright a, b, \mathtt{t}\,f, \mathtt{t}\,g, \mathtt{t}\,h, f, g, h}\,\mathtt{sum}}}{b = \mathtt{t}\,f \triangleright a, b, \mathtt{t}\,f, \mathtt{t}\,g, \mathtt{t}\,h, f, g, h}\,\mathtt{glue\text{-}}(b = \mathtt{t}\,f)}}{b = \mathtt{t}\,f = \mathtt{t}\,g \triangleright a, b, \mathtt{t}\,f, \mathtt{t}\,g, \mathtt{t}\,h, f, g, h}\,\mathtt{glue\text{-}}(b = \mathtt{t}\,g)$$

Next, we derive the last missing cell, α:

$$\dfrac{\dfrac{\vdots}{\dfrac{b = \mathtt{t}\,f = \mathtt{t}\,g \triangleright a, b, \mathtt{t}\,f, \mathtt{t}\,g, \mathtt{t}\,h, f, g, h}{b = \mathtt{t}\,f = \mathtt{t}\,g \triangleright a, b, \mathtt{t}\,f, \mathtt{t}\,g, \mathtt{t}\,h, f, g, h \vdash_1 f}\,\mathtt{pd}}}{E_1 \triangleright a, b, \mathtt{t}\,f, \mathtt{t}\,g, \mathtt{t}\,h, \mathtt{t}^2\,\alpha : \varnothing, f, g, h, \mathtt{t}\,\alpha : a, \alpha : f}\,\mathtt{shift}$$

where E_1 stands for $b = \mathtt{t}\,f = \mathtt{t}\,g, \mathtt{t}^2\,\alpha = \mathtt{t}\,f$. Note that in addition of α, this last step also introduces $\mathtt{t}\,\alpha$ and $\mathtt{t}^2\,\alpha$, as well as the identity $\mathtt{t}^2\,\alpha = \mathtt{t}\,f$, since $\mathtt{s}\,\alpha = f$. The rest of the derivation addresses the remaining gluings:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{E_1 \triangleright a, b, \mathtt{t}\,f, \mathtt{t}\,g, \mathtt{t}\,h, \mathtt{t}^2\,\alpha, f, g, h, \mathtt{t}\,\alpha, \alpha}\,\mathtt{glue\text{-}}(b = \mathtt{t}^2\,\alpha)}{E_2 \triangleright a, b, \mathtt{t}\,f, \mathtt{t}\,g, \mathtt{t}\,h, \mathtt{t}^2\,\alpha, f, g, h, \mathtt{t}\,\alpha, \alpha}\,\mathtt{glue\text{-}}(g = \mathtt{t}\,\alpha)}{E_3 \triangleright a, b, \mathtt{t}\,f, \mathtt{t}\,g, \mathtt{t}\,h, \mathtt{t}^2\,\alpha, f, g, h, \mathtt{t}\,\alpha, \alpha}\,\mathtt{glue\text{-}}(a = \mathtt{t}\,h)}{E_4 \triangleright a, b, \mathtt{t}\,f, \mathtt{t}\,g, \mathtt{t}\,h, \mathtt{t}^2\,\alpha, f, g, h, \mathtt{t}\,\alpha, \alpha}$$

where E_2, E_3, E_4 stand for $E_1, b = \mathtt{t}^2\,\alpha$, $E_2, g = \mathtt{t}\,\alpha$, and $E_3, a = \mathtt{t}\,h$, respectively.

VIII. Conclusion and future works

In this article, we have introduced sequent calculi in order to manipulate opetopes and opetopic sets. We are currently experimenting with an implementation [13], in order to evaluate how natural such systems are to use in practice.

We are also investigating variants of those systems, see [5]. In particular, instead of using named variables modulo α-conversion, we identify them by the address of the corresponding node in trees (the address being here the sequence of edges one should follow in order to reach the node starting from the root). This gives rise to "de Bruijn-like" notations, which are more difficult to read on paper, but can be more efficiently implemented because they avoid the renaming issues.

As explained in the introduction, we believe that this work is a first step toward a type-theoretic definition of opetopic weak ω-categories.

Acknowledgment

The first author has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement number 665850.

References

[1] Jiří Adámek and Jiří Rosický. *Locally presentable and accessible categories*, volume 189 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1994.

[2] John C. Baez and James Dolan. Higher-dimensional algebra. III. n-categories and the algebra of opetopes. *Advances in Mathematics*, 135(2):145–206, 1998.

[3] Krzysztof Bar, Aleks Kissinger, and Jamie Vicary. Globular: an online proof assistant for higher-dimensional rewriting. *arXiv e-prints*, page arXiv:1612.01093, December 2016.

[4] Eugenia Cheng. The category of opetopes and the category of opetopic sets. *Theory and Applications of Categories*, 11:No. 16, 353–374, 2003.

[5] Pierre-Louis Curien, Cédric Ho Thanh, and Samuel Mimram. Syntactic approaches for opetopes. `arXiv:1903.05848 [math.CT]`, March 2019.

[6] Eric Finster. Opetopic.net. `http://opetopic.net`, May 2016.

[7] Eric Finster and Samuel Mimram. A Type-Theoretical Definition of Weak ω-Categories. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, 2017.

[8] Nicola Gambino and Joachim Kock. Polynomial functors and polynomial monads. *Mathematical Proceedings of the Cambridge Philosophical Society*, 154(1):153–192, 2013.

[9] Victor Harnik, Michael Makkai, and Marek Zawadowski. Computads and multitopic sets. `arXiv:0811.3215 [math.CT]`, 2008.

[10] Claudio Hermida, Michael Makkai, and John Power. On weak higher dimensional categories. I. 1. *Journal of Pure and Applied Algebra*, 154(1-3):221–246, 2000. Category theory and its applications (Montreal, QC, 1997).

[11] Claudio Hermida, Michael Makkai, and John Power. On weak higher-dimensional categories. I. 3. *Journal of Pure and Applied Algebra*, 166(1-2):83–104, 2002.

[12] Cédric Ho Thanh. The equivalence between opetopic sets and many-to-one polygraphs. `arXiv:1806.08645 [math.CT]`, 2018.

[13] Cédric Ho Thanh. opetopy. `https://github.com/altaris/opetopy`, April 2018.

[14] Joachim Kock. Notes on polynomial functors. `http://mat.uab.es/~kock/cat/polynomial.pdf`.

[15] Joachim Kock. Polynomial functors and trees. *International Mathematics Research Notices*, 2011(3):609–673, January 2011.

[16] Joachim Kock, André Joyal, Michael Batanin, and Jean-François Mascari. Polynomial functors and opetopes. *Advances in Mathematics*, 224(6):2690–2737, 2010.

[17] Tom Leinster. *Higher Operads, Higher Categories*. Cambridge University Press, 2004.

[18] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in geometry and logic*. Universitext. Springer-Verlag, New York, 1994. A first introduction to topos theory, Corrected reprint of the 1992 edition.