A Syntactical Definition of Opetopes

Pierre-Louis Curien\textsuperscript{1}  Cédric Ho Thanh\textsuperscript{2}  Samuel Mimram\textsuperscript{3}

$\pi r^2$ “mise au vert”, April 11, 2018

\textsuperscript{1}IRIF, Paris Diderot University

\textsuperscript{2}IRIF, Paris Diderot University, INSPIRE 2017 Fellow, This project has received funding from the European Union’s Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 665850

\textsuperscript{3}LIX, École Polytechnique
Opetopes
Opetopes are shapes (akin to globules, cubes, simplices, etc.), designed to represent the notion of composition in every dimension. As such, they were introduced to describe laws and coherence in higher categories.
Opetopes are shapes (akin to globules, cubes, simplices, etc.), designed to represent the notion of composition in every dimension. As such, they were introduced to describe laws and coherence in higher categories.

Their distinctive feature is that they are many-to-one, and thus exhibit a tree-like structure.
Definition: base cases

- There is a unique 0-dimensional opetope, which we’ll call the *point* and denote by ♦.
Definition: base cases

- There is a unique 0-dimensional opetope, which we’ll call the *point* and denote by ♦.
- There is a unique 1-opetope, the arrow, denoted by ■.
Definition: base cases

- There is a unique 0-dimensional opetope, which we’ll call the *point* and denote by ♦.
- There is a unique 1-opetope, the arrow, denoted by □. The arrow has a *source* s□ = ♦ and a *target* t□ = ♦.
The opetopes $\diamond$ and $\blacksquare$ can be arranged in a tree (corolla) as follows:
Definition: dimension 2

A 2-opetope is then a coherent tree made with this corolla.
A 2-opetope is then a coherent tree made with this corolla.
A 2-opetope is then a coherent tree made with this corolla.
A 2-opetope is then a coherent tree made with this corolla.
A 2-opetope is then a coherent tree made with this corolla.
A 2-opetope is then a coherent tree made with this corolla.

Since the input edge and output edge of the above corolla have the same decoration (♦), the coherence condition is trivially satisfied.
Intuitively, such trees tell us how to compose the 1-opetope with itself.
Intuitively, such trees tell us how to compose the 1-opetope with itself.
Intuitively, such trees tell us how to compose the 1-opetope with itself.
Intuitively, such trees tell us how to compose the 1-opetope with itself.
In turn, a 2-opetope can be seen as a corolla, whose input vertices correspond to the occurrences of 1-opetopes (or equivalently, to its nodes). The 1-opetopes decorating the input edges are called the sources, while that decorating the root edge is the target.
In turn, a 2-opetope can be seen as a corolla, whose input vertices correspond to the occurrences of 1-opetopes (or equivalently, to its nodes). The 1-opetopes decorating the input edges are called the *sources*, while that decorating the root edge is the *target*.
Definition: dimension 2

In turn, a 2-opetope can be seen as a corolla, whose input vertices correspond to the occurrences of 1-opetopes (or equivalently, to its nodes). The 1-opetopes decorating the input edges are called the sources, while that decorating the root edge is the target.
Definition: dimension 2

In turn, a 2-opetope can be seen as a corolla, whose input vertices correspond to the occurrences of 1-opetopes (or equivalently, to its nodes). The 1-opetopes decorating the input edges are called the sources, while that decorating the root edge is the target.
For $n \geq 2$, an $n$-opetope is a tree made from the corollas associated to $(n - 1)$-opetopes.
Definition: dimension $\geq 2$

For $n \geq 2$, an $n$-opetope is a tree made from the corollas associated to $(n - 1)$-opetopes.

In turn, it can be seen as a corolla whose input vertices correspond to the occurrences of $(n - 1)$-opetopes.
Equivalently, for $n \geq 2$:

**Definition**

An $n$-opetope is a (finite) tree whose

- nodes are decorated by $(n - 1)$-opetopes,
- edges are decorated by $(n - 2)$-opetopes,
- all in a coherent way.
Graphical representations
As we saw before, a seemingly natural way of representing opetopes is by pasting schemes.
However...
Pasting schemes

However...

(this is representing an opetope of dimension only 4, diagram from [CL])
What’s wrong with trees as before?

![Diagram of a tree with nodes labeled \( \psi_1, \psi_2, \psi_3, \psi_4, \psi_5 \).]
What’s wrong with trees as before?

Each node is decorated by an \((n - 1)\)-opetope \((\psi_1, \ldots)\), so it should come with its own tree, and likewise for the nodes of that tree, etc... It quickly gets out of hand.
This tree approach has been reworked and improved by Batanin, Joyal, Kock, and Mascari [KJBM10].

A *constellation* is a sequence of trees with circles satisfying some conditions.
Syntax
Addresses

We seek to make the notion of occurrences more precise.
Addresses

We seek to make the notion of occurrences more precise.

Observation
If each edge has a name that is unique among its siblings, then each node can be uniquely identified by a path. In turn, the corolla associated to the tree has names for its input edges.
Addresses

We seek to make the notion of occurrences more precise.

Observation
If each edge has a name that is unique among its siblings, then each node can be uniquely identified by a path. In turn, the corolla associated to the tree has names for its input edges.
We seek to make the notion of occurrences more precise.

**Observation**

If each edge has a name that is unique among its siblings, then each node can be uniquely identified by a path. In turn, the corolla associated to the tree has names for its input edges.
In order for all opetopes (of dimension $\geq 2$) to have such addresses, we need to bootstrap the process in dimension 1:
Addresses: base case

In order for all opetopes (of dimension $\geq 2$) to have such addresses, we need to bootstrap the process in dimension 1:

We decide that the name of the input edge is $\ast$ (in blue).
As an example of how the process continues:
As an example of how the process continues:
As an example of how the process continues:
Our take on the representation of opetopes is a syntactic one. Take $\omega$ an $n$-opetope, for $n \geq 2$, as below:
Our take on the representation of opetopes is a syntactic one. Take $\omega$ an $n$-opetope, for $n \geq 2$, as below:

\begin{figure}
\centering
\begin{tikzpicture}
\node (v1) at (0,0) [label=below:$\psi_1$] {$[\epsilon]$};
\node (v2) at (1,-1) [label=left:$\psi_2$] {$[a]$};
\node (v3) at (2,-1) [label=right:$\psi_3$] {$[b]$};
\node (v4) at (2.8,0) [label=right:$\psi_4$] {$[c]$};
\node (v5) at (2.8,1) [label=right:$\psi_5$] {$[aa]$};
\draw (v1) -- (v2);
\draw (v1) -- (v3);
\draw (v2) -- (v5);
\draw (v3) -- (v5);
\draw (v4) -- (v5);
\end{tikzpicture}
\end{figure}

**Observation**

In $\omega$, each node is decorated by a $(n - 1)$-opetope.
Idea

Our take on the representation of opetopes is a syntactic one. Take $\omega$ an $n$-opetope, for $n \geq 2$, as below:

```
ψ₅ → [aa] → ψ₂
   |         |
   |         |
ψ₁ → [ε] → ψ₃ → [b] → ψ₄ → [c]
```

Observation

In $\omega$, each node is decorated by a $(n - 1)$-opetope, and each node has an address.
Thus, the syntactic representation (or coding) $\lowercase{c} \omega \lowercase{w}$ of $\omega$ is

$$
\lowercase{c} = \begin{cases}
[\varepsilon] & \leftarrow \lowercase{c}\psi_1 \lowercase{w} \\
[a] & \leftarrow \lowercase{c}\psi_2 \lowercase{w} \\
[b] & \leftarrow \lowercase{c}\psi_3 \lowercase{w} \\
[c] & \leftarrow \lowercase{c}\psi_4 \lowercase{w} \\
[aa] & \leftarrow \lowercase{c}\psi_5 \lowercase{w}
\end{cases}
$$
The 0 and 1-opetopes are coded by

\[ \begin{align*}
\downarrow \diamond \uparrow &= \diamond, & \downarrow \blacksquare \uparrow &= \blacksquare = \{ * \leftarrow \diamond \}. 
\end{align*} \]
The 0 and 1-opetopes are coded by

\[ \begin{array}{ll}
\downarrow \diamond \downarrow & = \diamond, \\
\downarrow \square \downarrow & = \square \leftarrow \diamond.
\end{array} \]

Degenerate opetopes are coded as follows:

\[ \phi \bigg\| \{ \{ \downarrow \phi \downarrow \} \bigg\} \]
Preopetopes

\[
\begin{align*}
0\text{-}addr & ::= \ast \\
n\text{-}addr & ::= [\epsilon] \quad n \geq 1 \\
& | [(n - 1)\text{addr} \ldots (n - 1)\text{addr}] \quad n \geq 1 \\
0\text{-}preopt & ::= \blacklozenge \\
n\text{-}preopt & ::= \\
& \begin{cases}
(n - 1)\text{-}addr \leftrightarrow (n - 1)\text{-}preopt \\
(n - 1)\text{-}addr \leftrightarrow (n - 1)\text{-}preopt \\
\{\{n - 2\}\text{-}preopt \quad n \geq 2
\end{cases}
\end{align*}
\]
Example

\[
\begin{align*}
[\epsilon] & \leftarrow \\
[[\ast]] & \leftarrow \\
[[[\ast]]] & \leftarrow
\end{align*}
\]
We thus define a coding function

\[\Psi : n\text{-opetopes} \rightarrow n\text{-preopt}.\]

**Observation**

Clearly, if \(\omega \neq \omega'\), then \(\Psi(\omega) \neq \Psi(\omega')\).
Preopetopes

We thus define a coding function

\[ \prec \rightarrow : n\text{-opetopes} \rightarrow n\text{-preopt.} \]

Observation
Clearly, if \( \omega \neq \omega' \), then \( \prec \omega \neq \prec \omega' \).

- If they have the same dimension, then their tree structure differ and the result is immediate.
Preopetopes

We thus define a coding function

\[ \pre : n\text{-opetopes} \rightarrow n\text{-preopt} \]

**Observation**

Clearly, if \( \omega \neq \omega' \), then \( \pre(\omega) \neq \pre(\omega') \).

- If they have the same dimension, then their tree structure differ and the result is immediate.
- The “depth” of \( \pre(\omega) \) is the dimension of \( \omega \), so if \( \omega \) and \( \omega' \) don’t have the same dimension, then \( \pre(\omega) \) and \( \pre(\omega') \) don’t have the same depth, hence are different.
Question

Is $\text{−}\text{−}$ surjective? In other words, is all preopetope the coding of an actual opetope?
Question
Is $\vdash \vdash$ surjective? In other words, is all preopetope the coding of an actual opetope?

No!

\[ \{ [*] \leftarrow \psi \} \]
Question
Is $\vdash \rightarrow$ surjective? In other words, is all preopetope the coding of an actual opetope?

No!

\[
\begin{cases}
[*] & \vdash \psi
\end{cases}
\]

Goal
Create a deduction system where derivable preopetoposes are exactly the codings of opetoposes.
Inference rules: structure of a sequent

Our system will deal with sequents of the form

\[ \Gamma \vdash p \]

where

- \( p \) is a preopetope;
- \( \Gamma \) is the set of the (addresses of the) leaves of \( p \).
Inference rules: structure of a sequent

Our system will deal with sequents of the form

\[ \Gamma \vdash p \]

where

- \( p \) is a preopetope;
- \( \Gamma \) is the set of the (addresses of the) leaves of \( p \).

\[ \Gamma = \{ [aad], [aae], [ce] \} \]
In order to bootstrap the derivation process, rule **point** creates a 0-opetope ♦ from no premise.

\[
\Gamma \vdash \diamond \quad \text{point}
\]
Inference rules: degen

Given a $n$-preopetope $p$, we can create the degenerate $(n + 2)$-preopetope at $p$, i.e. a tree with no node, and whose unique edge is decorated by $p$.

$$
\frac{\Gamma \vdash p}{\vdash \{\{p\}\} \text{degen}}
$$
Inference rules: degen

Given a $n$-preopetope $p$, we can create the degenerate $(n + 2)$-preopetope at $p$, i.e. a tree with no node, and whose unique edge is decorated by $p$.

\[
\frac{\Gamma \vdash p}{\vdash \text{degen} \quad \begin{array}{c} \vdash \{p \end{array}}
\]

\[
\begin{tikzpicture}
  \node (p) at (0,0) {$p$};
  \node (p1) at (0,1) {$\vdash \{p$};
  \node (p2) at (0,2) {\vdash \text{degen}$};
  \node (p3) at (0,3) {\vdash \Gamma \vdash p$};
\end{tikzpicture}
\]
Given a $n$-preopetope $p$, we can create the degenerate $(n + 2)$-preopetope at $p$, i.e. a tree with no node, and whose unique edge is decorated by $p$. 

\[
\frac{\Gamma \vdash p}{\vdash \{ \{ p \} \}} \text{ degen}
\]
Inference rules: shift

Given a \( n \)-preopetope \( p \) = \{
\[
[p_1] \leftarrow q_1 \\
\vdots \\
[p_k] \leftarrow q_k
\]
\}, we can create an \((n + 1)\)-preopetope, the corolla whose input edges correspond to the nodes of \( p \).

\[
\frac{\Gamma \vdash p}{[[p_1]], \ldots, [[p_k]] \vdash \{[\epsilon] \leftarrow p} \quad \text{shift}
\]
Inference rules: shift

Given a \( n \)-preopetope \( p = \{ [p_1] \leftarrow q_1, \ldots, [p_k] \leftarrow q_k \} \), we can create an \( (n + 1) \)-preopetope, the corolla whose input edges correspond to the nodes of \( p \).

\[
\Gamma \vdash p \\
[[p_1]], \ldots, [[p_k]] \vdash \{ [\epsilon] \leftarrow p \}
\]

\[
\begin{array}{c}
q_1 \quad [\epsilon] \\
q_2 \quad [p_2] \\
q_3 \quad [p_3] \\
q_4 \quad [p_4]
\end{array}
\]
Inference rules: shift

Given a $n$-preopetope $p = \left\{ [p_1] \leftarrow q_1, \ldots, [p_k] \leftarrow q_k \right\}$, we can create an $(n + 1)$-preopetope, the corolla whose input edges correspond to the nodes of $p$.

\[
\frac{\Gamma \vdash p}{[[p_1]], \ldots, [[p_k]] \vdash \left\{ [\epsilon] \leftarrow p \right\}} \text{ shift}
\]
We assume the existence of a magical \textit{target} function

\[
t : \{n\text{-preopt.}\} \longrightarrow \{ (n - 1)\text{-preopt.} \}
\]
Inference rules: \textit{graft}

Given an \(n\)-preopetope \(p\) on the left with a leaf at address \([p[q]]\), and an \((n-1)\)-preopetope \(q\), with target \(t\ q = r\), we may graft \(q\) on \(p\).

\[
\Gamma, [p[q]] \vdash \begin{cases}
[p] & \leftarrow [q] & \leftarrow r & \Delta \vdash q
\end{cases}
\]

where \(\Xi = \{[p[q]d] \mid [d] \text{ address in } q\}\).
Inference rules: graft

\[ [p[q]] \]

\[ [p] \]
Inference rules: graft
Theorem (Curien–HT–Mimram)
The set of derivable preopetopes is in one-to-one correspondence with the set of Leinster opetopes [Lei, KJBH10].
Theorem (Curien–HT–Mimram)
The set of derivable preopetopes is in one-to-one correspondence with the set of Leinster opetopes [Lei, KJBM10].

The proof essentially boils down to make the previous drawings rigorous.

We now refer to derivable preopetopes as simply opetopes.
Theorem (Curien–HT–Mimram)

Opetopes form a decidable subset of the set of preopetopes.
Well-formation algorithm

Theorem (Curien–HT–Mimram)
Opetopes form a decidable subset of the set of preopetopes.
The naive algorithm, that seeks to deconstruct the potential proof tree of a given preopetope, runs in exponential time w.r.t. the number of preopetopes involved in each dimension.
Conclusion
To summarize

• Opetopes are tricky, highly inductive objects.
• Following up on major advancements on the matter [Lei, KJBM10], we propose a purely syntactic definition.
• We believe such explicit “writable” expressions will make opetopes easier to use for subsequent work.
Thank you for your attention!
Eugenia Cheng and Aaron Lauda.  

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

Tom Leinster.  
*Higher operads, higher categories.*