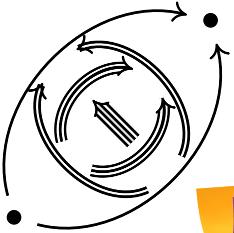
Computational Models of Higher Categories
Lecture 1

Jamie Vicary
University of Cambridge

Midland Graduate School in the Foundations of Computing Science
University of Birmingham
2-6 April 2023

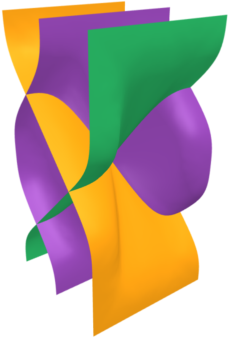
Motivation

Higher category theory describes the composition and equivalence of higher dimensional-processes.



The mathematical theory has a reputation for complexity, “generally regarded as a technical and forbidding subject” (Lurie).

A computational lens helps to bring out the simplicity and accessibility of the subject.



Higher categories are dynamical objects, and best understood by using and manipulating them.

The definition of 1-category

Definition. A *1-category* **C** is given by:

- a collection $\mathrm{Ob}(\mathbf{C})$ of objects
- for any objects A, B, a set of morphisms $\mathbf{C}(A, B)$
- for any object A an identity morphism $\mathrm{id}_A \in \mathbf{C}(A, A)$
- for any objects A, B, C a composition operation $-\circ- : \mathbf{C}(A, B) \times \mathbf{C}(B, C) \to \mathbf{C}(A, C)$
- for any composable morphisms f, g, h an equality $f \circ (g \circ h) = (f \circ g) \circ h$
- for any morphism $f : A \to B$ the equalities $f \circ \mathrm{id}_B = f = \mathrm{id}_A \circ f$

Example. Let $\mathrm{Ob}(\mathbf{C})$ be the empty set.

Overall that corresponds to the following:

- 2 sets (ignoring size issues)
- 2 functions
- 3 equations

Onwards and upwards!

The definition of 2-category

Definition. A *2-category* $\mathbf{C}$ consists of the following data:

- a collection $\mathrm{Ob}(\mathbf{C})$ of *objects*
- for any two objects A, B, a category $\mathbf{C}(A, B)$, with objects called *1-morphisms* drawn as $f : A \to B$, and morphisms called *2-morphisms* drawn as $m : f \Rightarrow g$, or in full form as follows:

$$\begin{tikzcd}
A \arrow[r, bend left=50, "g", ""{name=U, below}] \arrow[r, bend right=50, "f"', ""{name=D}] & B
\arrow[Rightarrow, from=D, to=U, "m"']
\end{tikzcd}$$

- for 2-morphisms $m : f \Rightarrow g$ and $n : g \Rightarrow h$, an operation called *vertical composition* given by their composite as morphisms in $\mathbf{C}(A, B)$, written $m \bullet n$:

$$\begin{tikzcd}
A \arrow[r, bend left=70, "h", ""{name=H, below}] \arrow[r, "g" description, ""{name=G}] \arrow[r, bend right=70, "f"', ""{name=F}] & B
\arrow[Rightarrow, from=G, to=H, "n"']
\arrow[Rightarrow, from=F, to=G, "m"']
\end{tikzcd}$$

The definition of 2-category

- for any triple of objects A, B, C a *horizontal composition* functor:

$$- \circ - : \mathbf{C}(A,B) \times \mathbf{C}(B,C) \to \mathbf{C}(A,C)$$

$$\begin{tikzcd}
A \arrow[r, bend left=50, "g \circ j"{name=U}] \arrow[r, bend right=50, "f \circ h"'{name=D}] & C & = & A \arrow[r, bend left=50, "g"{name=U1}] \arrow[r, bend right=50, "f"'{name=D1}] & B \arrow[r, bend left=50, "j"{name=U2}] \arrow[r, bend right=50, "h"'{name=D2}] & C
\arrow[from=D, to=U, Rightarrow, "m \circ n"']
\arrow[from=D1, to=U1, Rightarrow, "m"']
\arrow[from=D2, to=U2, Rightarrow, "n"']
\end{tikzcd}$$

- for any object A, a 1-morphism $\mathrm{id}_A : A \to A$ called the *identity 1-morphism*
- natural families of invertible 2-morphisms $\rho_f : f \circ \mathrm{id} \Rightarrow f$ and $\lambda_f : \mathrm{id} \circ f \Rightarrow f$ called the *right and left unitors*
- a natural family of invertible 2-morphisms $\alpha_{f,g,h} : (f \circ g) \circ h \Rightarrow f \circ (g \circ h)$ called the *associators*

The definition of 2-category

- for composable 1-morphisms f, g, the *triangle equation* must be satisfied:

$$\begin{tikzcd}
(f \circ \mathrm{id}) \circ g \arrow[rr, Rightarrow, "\alpha_{f,\mathrm{id},g}"] \arrow[dr, Rightarrow, "\rho_f \circ \mathrm{id}_g"'] & & f \circ (\mathrm{id} \circ g) \arrow[dl, Rightarrow, "\mathrm{id}_f \circ \lambda_g"] \\
& f \circ g &
\end{tikzcd}$$

- for composable 1-morphisms f, g, h, j, the *pentagon equation* must be satisfied:

$$\begin{tikzcd}
& \big(f \circ (g \circ h)\big) \circ j \arrow[rr, Rightarrow, "\alpha_{f,g \circ h,j}"] & & f \circ \big((g \circ h) \circ j\big) \arrow[dr, Rightarrow, "\mathrm{id}_f \circ \alpha_{g,h,j}"] & \\
\big((f \circ g) \circ h\big) \circ j \arrow[ur, Rightarrow, "\alpha_{f,g,h} \circ \mathrm{id}_j"] \arrow[drr, Rightarrow, "\alpha_{f \circ g,h,j}"'] & & & & f \circ \big(g \circ (h \circ j)\big) \\
& & (f \circ g) \circ (h \circ j) \arrow[urr, Rightarrow, "\alpha_{f,g,h \circ j}"'] & &
\end{tikzcd}$$

An important consequence is *coherence* — all well-formed equations commute.

This structure is sometimes called a *bicategory*, or *weak 2-category*.

The 2-category of categories

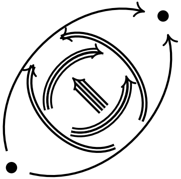
Example. The 2-category **Cat** is defined as follows:

- **objects** are categories
- **1-morphisms** are functors
- **2-morphisms** are natural transformations
- **vertical composition** is componentwise composition of natural transformations, with $(\mu \cdot \nu)_A := \mu_A \circ \nu_A$
- **horizontal composition** is composition of functors

Definitional complexity

Let's think about the complexity of these definitions:

- 1-category: 2 sets, 2 functions, 3 axioms
- 2-category: 3 sets, 6 functions, 6 axioms
- 3-category: 4 sets, 19 functions, 58 axioms
- 4-category: 5 sets, 34 functions, 118 axioms (ish)



As the dimension increases, just *writing down* these definitions becomes difficult.

Furthermore, *using* the definitions becomes almost impossible.

Homotopy theory increases in complexity in each dimension. We need a new approach.

Foundations of higher categories

Alexander Grothendieck was one of the greatest modern mathematicians.

He was obsessed with finding an axiomatic system for ‘well-behaved’ topological spaces, avoiding paradoxes like Banach-Tarski.

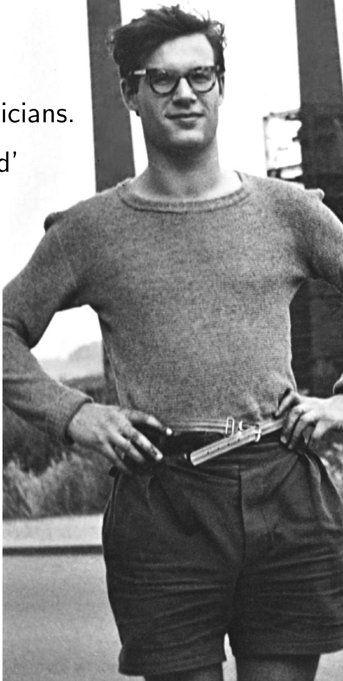
One thing which strikes me ... is the absence of proper foundations for topology itself!

He wrote a famous letter in 1983 to the mathematician Daniel Quillen, where he sketched some ideas, but concluded:

One seems caught in an infinite chain of ever messier structures ... one is going to get hopelessly lost, unless one discovers some simple guiding principle.

The next day he solved the problem, and wrote in another letter:

I went on pondering ... motivation does furnish a simple guiding principle in order not to get lost in the messiness of higher structures.



Paths as types

Grothendieck's idea begins with *n*-dimensional disks:

$$\begin{tikzcd} x \bullet \end{tikzcd}$$

$$\begin{tikzcd} x \bullet \arrow[r, bend left, "f"] & \bullet x' \end{tikzcd}$$

$$\begin{tikzcd}[column sep=huge] x \bullet \arrow[r, bend left=50, "f'"] \arrow[r, bend right=50, "f"'] \arrow[r, phantom, "\Uparrow \alpha"] & \bullet x' \end{tikzcd}$$

$$\begin{tikzcd}[column sep=huge] x \bullet \arrow[r, bend left=50, "f'"] \arrow[r, bend right=50, "f"'] \arrow[r, phantom, "\alpha \Uparrow \overset{\Omega}{\Rrightarrow} \Uparrow \alpha'"] & \bullet x' \end{tikzcd}$$

$\ldots$

$D_0 := x : \star$

$D_1 := x : \star,\ x' : \star,$
$f : x \to y$

$D_2 := x : \star,\ x' : \star,$
$f : x \to x',\ f' : x \to x',$
$\alpha : f \to x'$

$D_3 := x : \star,\ x' : \star,$
$f : x \to x',\ f' : x \to x',$
$\alpha : f \to f',\ \alpha' : f \to f',$
$\Omega : \alpha \to \alpha'$

The type $\star$ means *point*. An arrow type $S \to T$ means *path*.

We represent these disks by giving lists of the points and paths in their neighbourhood.

For each disk we give its full *context*: the *types* of all elements involved.

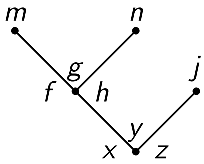
Pasting schemes

Grothendieck suggested "gluing" these disks together, forming geometric *pasting schemes*.

These are beautiful combinatorial objects, and we will look at 4 different representations.

$$\begin{tikzcd}
x \arrow[r, bend left=80, "h"] \arrow[r, "g"] \arrow[r, bend right=80, "f"'] \arrow[r, phantom, shift left=3, "\Uparrow n"] \arrow[r, phantom, shift right=3, "\Uparrow m"] & y \arrow[r, "j"] & z
\end{tikzcd}$$

Disk



Tree

$$x(f(m)g(n)h)y(j)z$$

List

$$x : \star,\ y : \star,\ z : \star,$$
$$f : x \to y,\ g : x \to y,$$
$$h : x \to y,\ j : y \to z,$$
$$m : f \to g,\ n : g \to h$$

Context

The disk perspective gives the fundamental geometrical intuition.

The tree perspective shows every variable at a height given by its dimension.

The list perspective is most economical. It arises from the tree by "tracing round".

The context perspective is most explicit, and also more general.

The *leaf variables* are the topmost elements. Here they are m, n, j.

Words

What are the k-cells in the *free ∞-category* on a pasting scheme? We call these the *words*.

For example, consider $P \;=\; x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w$.

We expect the free ∞-category on P to contain the following words:

- *Objects.* Just four: x, y, z, w.
- *1-cells.* Infinitely many: $f,\; g,\; h,\; f \circ g,\; f \circ (g \circ h),\; \mathrm{id}_x,\; \mathrm{id}_y,\; \mathrm{id}_z,\; f \circ \mathrm{id}_x,\; \ldots$
- *2-cells.* Infinitely many: $\mathrm{id}_{\mathrm{id}_x},\; \mathrm{id}_{f\circ g},\; \alpha_{f,g,h},\; \lambda_f,\; \ldots$
- *3-cells.* Infinitely many: $\ldots$
- $\ldots$

Looking at this suggests *four basic types* of word:

- *Variables.* These are all cells in their own right: x, y, z, w, f, g, h.
- *Composites.* The cell $f \circ g$ is built by "composing" f and g.
- *Equivalences.* These are "laws", like $\mathrm{id}_x,\; \alpha_{f,g,h},\; \lambda_f$.
- *Substituted.* These are "compound" objects, such as $f \circ \mathrm{id}_x,\; \mathrm{id}_{f\circ g},\; f \circ (g \circ h)$.

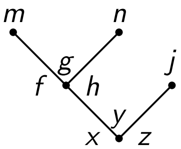
Boundaries of pasting schemes

Given a pasting scheme P, we can build its *source and target boundaries* $\partial^-(P)$, $\partial^+(P)$.

Let's demonstrate this with the previous example:

P

$$\begin{tikzcd}
x \arrow[r, bend left=70, "h"] \arrow[r, "g"] \arrow[r, bend right=70, "f"'] \arrow[r, phantom, shift left=3ex, "\Uparrow n"] \arrow[r, phantom, shift right=3ex, "\Uparrow m"] & y \arrow[r, "j"] & z
\end{tikzcd}$$



$x(f(m)g(n)h)y(j)z$

$x : \star,\ y : \star,\ z : \star,$

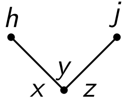
$f : x \to y,\ g : x \to y,$

$h : x \to y,\ j : y \to z,$

$m : f \to g,\ n : g \to h$

$\partial^+(P)$

$$\begin{tikzcd}
x \arrow[r, "h"] & y \arrow[r, "j"] & z
\end{tikzcd}$$



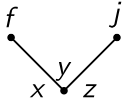
$x(h)y(j)z$

$x : \star,\ y : \star,\ z : \star,$

$h : x \to y,\ j : y \to z,$

$\partial^-(P)$

$$\begin{tikzcd}
x \arrow[r, "f"] & y \arrow[r, "j"] & z
\end{tikzcd}$$



$x(f)y(j)z$

$x : \star,\ y : \star,\ z : \star,$

$f : x \to y,\ j : y \to z,$

The trees handle this nicely: chop off the tree top, keeping the left- or right-most variable.

Composites and coherences

Here is the essence of Grothendieck's big idea:

Composites $\boxed{\text{Given a } \partial^-(P)\text{-word } u \text{ and a } \partial^+(P)\text{-word } v\text{, get a } P\text{-word coh } P : u \Rightarrow v}$

Equivalences $\boxed{\text{Given } P\text{-words } u, v\text{, get a } P\text{-word coh } P : u \Rightarrow v}$

When we invoke these rules, we must also check these side-conditions:

- *Dimension.* The words u, v must have the same dimension.
- *Boundary.* The words u, v must have the same source and target ("globularity").
- *Fullness.* The words u, v must use all the variables of their pasting schemes.

We define $\dim(\text{coh}\, P : u \Rightarrow v) = \dim(u) + 1$, and on variables dim does the obvious thing.

In fact Grothendieck's original scheme didn't use the fullness condition.
He was interested in ∞-*groupoids*, whereas we are focusing here on ∞-categories.

This scheme generates all the cells of traditional globular n-categories ... and more!

Composites and coherences

Let's think about why this works. Here's an example:

$$\begin{tikzcd}
\partial^+(P) & x \arrow[r, "h"] & y \arrow[r, "j"] & z & h \circ (j \circ \mathrm{id}(z)) \\
P & x \arrow[r, bend left=70, "h"] \arrow[r, "g"] \arrow[r, bend right=70, "f"'] \arrow[r, phantom, bend left=35, "\Uparrow n"] \arrow[r, phantom, bend right=35, "\Uparrow m"] & y \arrow[r, "j"] & z & \mathrm{coh}\ P : f \circ j \Rightarrow h \circ (j \circ \mathrm{id}(z)) \\
\partial^-(P) & x \arrow[r, "f"] & y \arrow[r, "j"] & z & f \circ j
\end{tikzcd}$$

Let's check the side-conditions: *Dimension* *Boundary* *Fullness*

To compose the elements of P, it's enough to know how to compose the *boundary* of P.

This works because every pasting scheme is *contractible*—homotopy equivalent to a point.

If you stop to think about it, it's a surprising and profound idea.

The proof assistant Catt

The proof assistant Catt verifies formal statements in this theory, with the following syntax.

- **Coherence construction.**

  Syntax. `coh name (pasting) : source => target`

  Example. `coh comp (x(f)y(g)z) : x => z`

- **Coherence application.** Only the leaf arguments are needed.

  Syntax. `...  (name arg1 arg2 ...)  ...`

  Example. `...  (comp p q) ...`

- **Comment.**

  Syntax. `# what a wonderful day`

Equality and truncation

We can use this theory to build the words of a finitely-generated ∞-category.

The only equality relation that we impose is α-*equivalence*, i.e. renaming bound variables.

Here is a simple example:

```
coh comp (x(f)y(g)z) : x => z
coh newcomp (u(p)v(q)w) : u => w
```

The words `comp` and `newcomp` will be considered identical in the theory.

Sometimes we want to work in an *n*-category, rather than an ∞-category.

To achieve this we can use *truncation*: for *n*-cells p, q, we consider $p = q$ just when there exists some *invertible* $(n+1)$-cell $p \to q$.

A cell is *invertible* when it is an equivalence, or a composite of invertible cells.

Examples

Let's look at some examples to get a feel for this new definition.

$x \xrightarrow{f} y \xrightarrow{g} z$ `coh comp (x(f)y(g)z) : x => z` "$f \circ g$"

x `coh id0 (x) : x => x` "$\mathrm{id}(x)$"

$x \xrightarrow{f} y$ `coh lunit (x(f)y) : comp f (id0 y) => f` "λ_f"

$x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w$ `coh comp3 (x(f)y(g)z(h)w) : x => w` "$f \circ g \circ h$" (!)

$x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w$ "$\alpha_{f,g,h}$"

```
coh assoc (x(f)y(g)z(h)w) :
  comp (comp f g) h => comp f (comp g h)
```

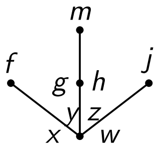
~~$x \xrightarrow{f} y$ `coh finv (x(f)y) : y => x` "f^{-1}"~~

~~$x \xrightarrow{f} y \xrightarrow{g} z$ `coh justf (x(f)y(g)z) : f => f` "f"~~

Class 1 – Activities

Activity 1.1. For each pasting scheme below, write it in ball, tree, list and context form.

(a)
$$\begin{tikzcd} x \arrow[r, bend left, "g"] \arrow[r, bend right, "f"'] \arrow[r, phantom, "\Uparrow m"] & y \arrow[r, bend left, "j"] \arrow[r, bend right, "h"'] \arrow[r, phantom, "\Uparrow n"] & z \end{tikzcd}$$

(b) 

(c) $(x(f(m(p)n)g)y)$

Activity 1.2. Get the proof assistant Catt working on your machine (see course webpage for instructions). Enter the examples on slide 18 to check they are correct.

Activity 1.3. Use the proof assistant Catt to build the following coherence cells.

(a) The triangle coherence 3-cell (see slide 6.)

(b) The pentagon coherence 3-cell (see slide 6.)

(c) The unit coherence 3-cell $\lambda_{\text{id}(x)} \to \rho_{\text{id}(x)}$. (Here x is an object.)

(d) The interchanger coherence 3-cell $(p \bullet q) \circ (r \bullet s) \to (p \circ r) \bullet (q \circ s)$, where $\circ$ is horizontal composition, and $\bullet$ is vertical composition. (Here p, q, r, s are 2-cells.)

(e) (Hard!) The associahedron 4-cell, which expresses coherence of the pentagon.