A diagram editor to mechanise categorical proofs

Ambroise Lafont

CoqPL, 20 January 2024

Naming convention

Yet Another Diagram Editor

I will refer to my editor as **YADE**, or Coreact-**YADE**.

ANR Project[1] (2023 - 2027): Coq-based Rewriting: Towards Executable Applied Category Theory

[1] https://coreact.wiki/

How to test the editor

A web app that runs locally in your browser

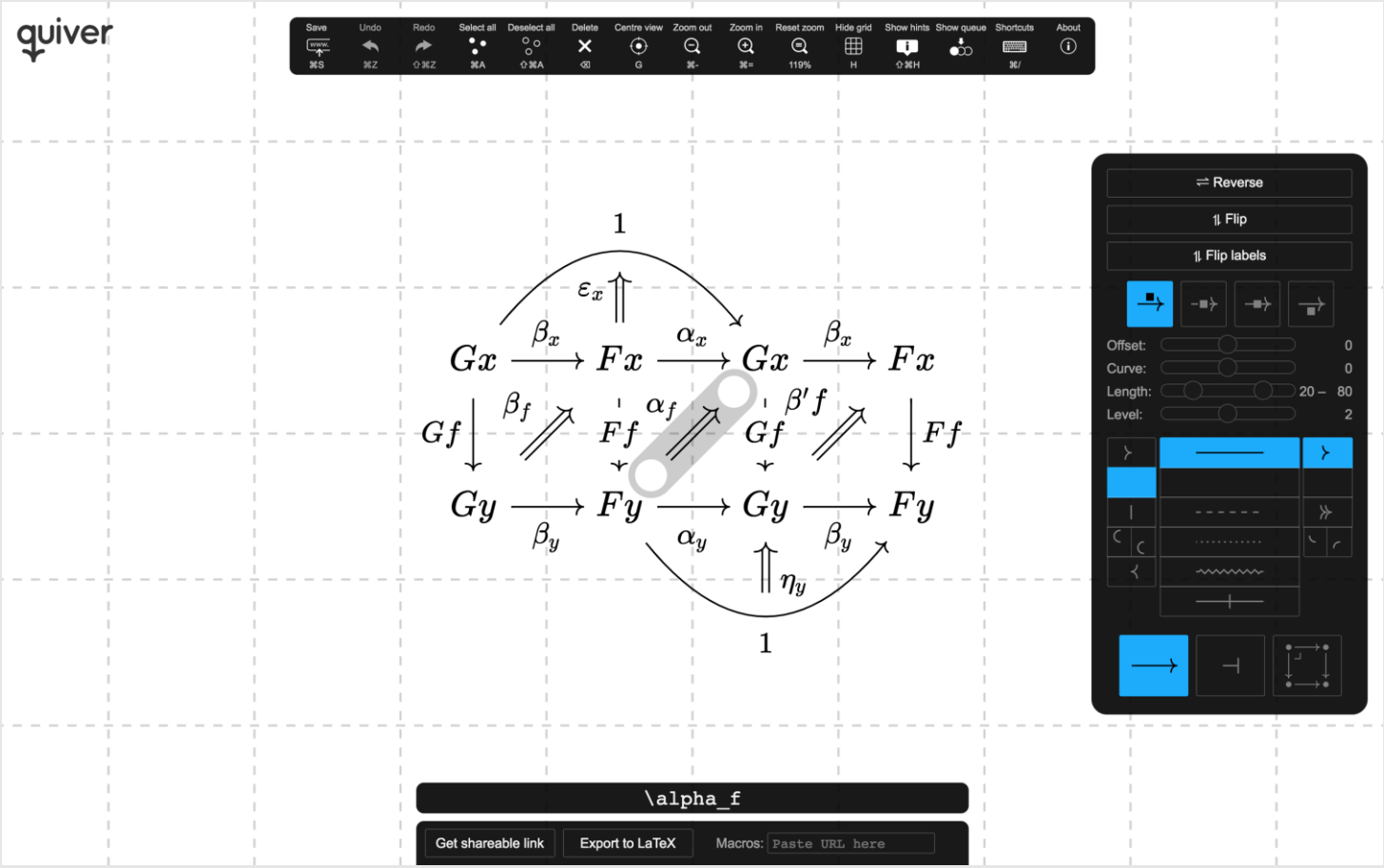
https://amblafont.github.io/graph-editor/index.html

A standalone desktop program

- Embeds the web app using electron
- Additional features (mechanisation)

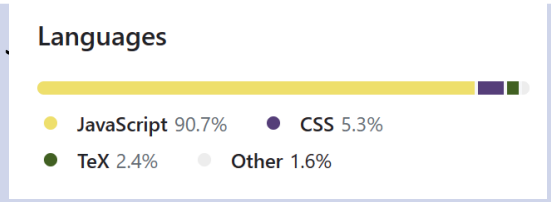
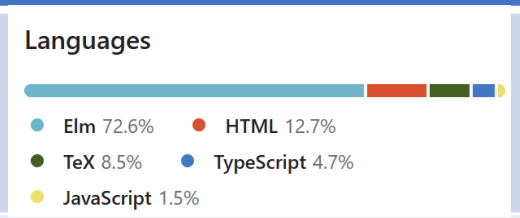
Related software: Quiver

"a modern, graphical editor for commutative and pasting diagrams, capable of rendering high-quality diagrams for screen viewing, and exporting to LaTeX via tikz-cd."



Comparison with quiver

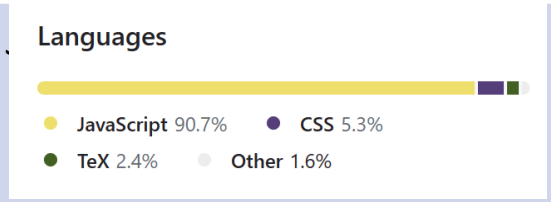
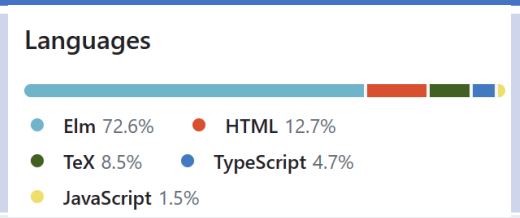
About the same size (around 10k of LoC)

	Quiver	YADE
Programming Languages		
Styling options	+	-
User-friendly	+	-
Editing features	-	+ Tabs, copy & paste, find & replace, expand selection to connected components, …
LaTeX export	yes	yes[1]

[1] Implemented by Tom Hirschowitz

Comparison with quiver

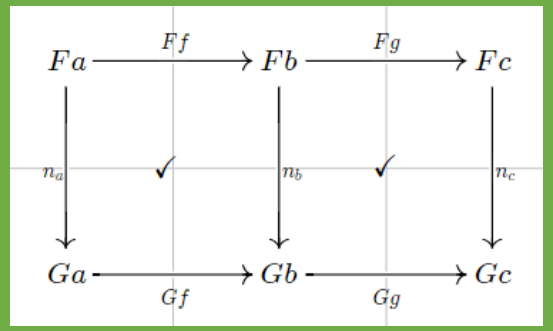
About the same size (around 10k of LoC)

	Quiver	YADE
Programming Languages		
Styling options	+	-
User-friendly	+	-
Editing features	-	+ Tabs, copy & paste, find & replace, expand selection to connected components, …
LaTeX export	yes	yes[1]
Mechanisation features	-	+

[1] Implemented by Tom Hirschowitz

Architecture

Diagram editor
(standalone version)

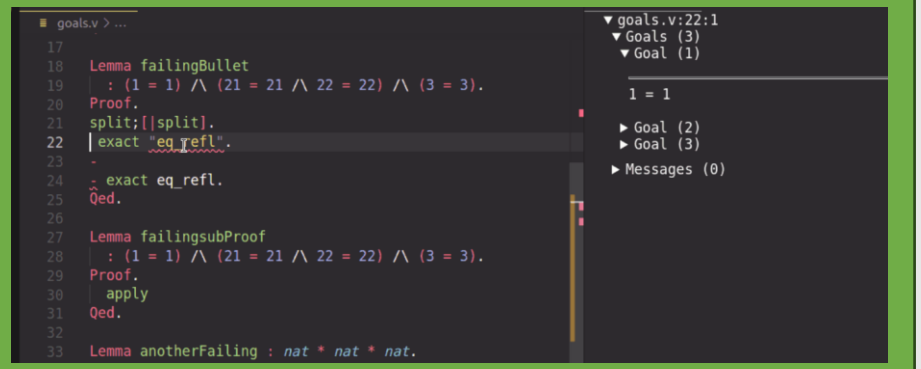


Show diagram under cursor

Generate proofs

Custom vscode extension (building upon coq-lsp)

Visual Studio Code

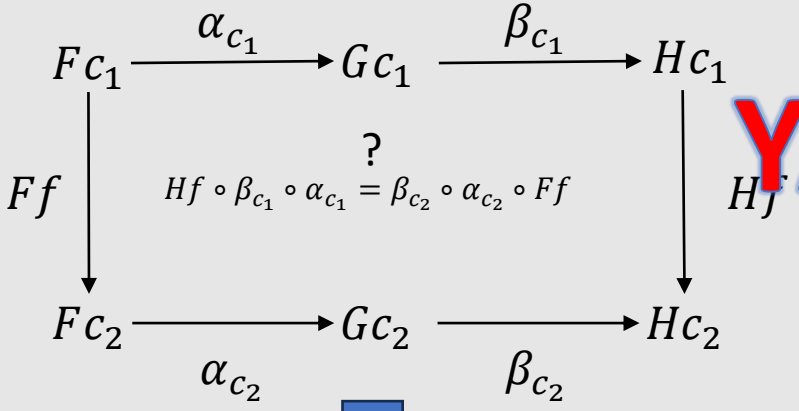


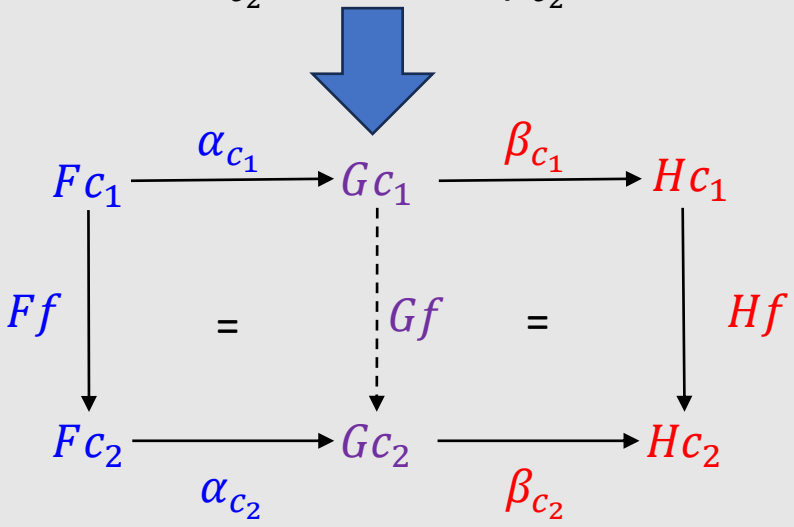
(+ Coq library for custom notations)

Natural transformations compose:

If $\alpha: F \Rightarrow G$ and $\beta: G \Rightarrow H$ are natural, then so is $(\beta_c \circ \alpha_c: Fc \to Hc)_c$

Diagrammatic proof





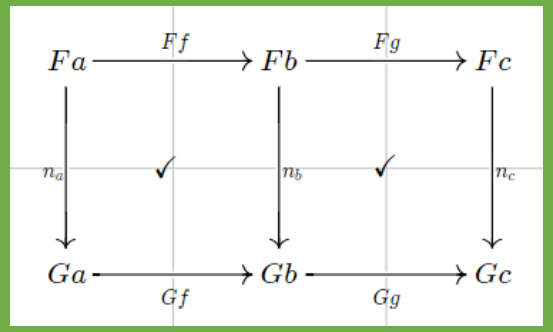
Automatic generation?

Computer-friendly proof

$$Hf \circ \beta_{c_1} \circ \alpha_{c_1}$$
$$= \beta_{c_2} \circ Gf \circ \alpha_{c_1}$$
$$= \beta_{c_2} \circ \alpha_{c_2} \circ Ff$$

Architecture

Diagram editor
(standalone version)



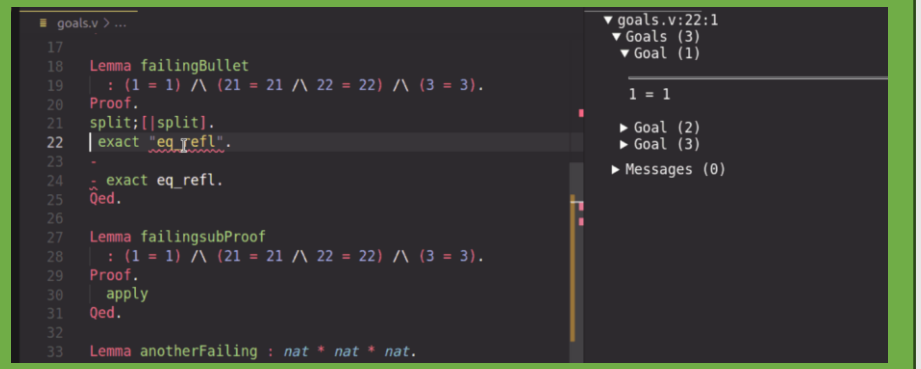
Show diagram under cursor

Generate proofs

Construct diagrams step by step

Custom vscode extension (building upon coq-lsp)

Visual Studio Code



(+ Coq library for custom notations)

Building the diagrammatic proof interactively

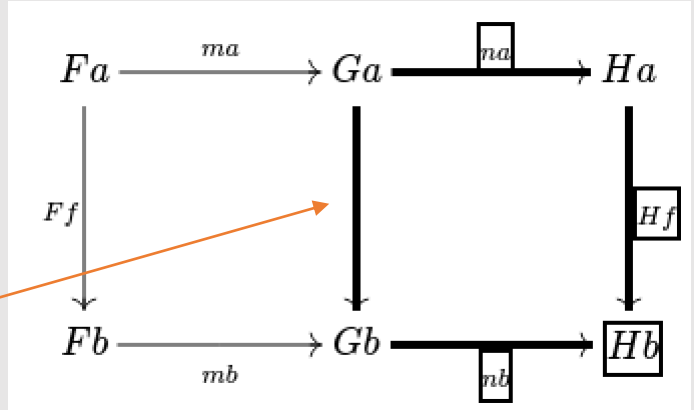
1) Select a subdiagram

2) Create a proof node, labelled with the Coq tactic `naturality.`

⇒ Coq (in vscode) checks that this tactic solves the goal:

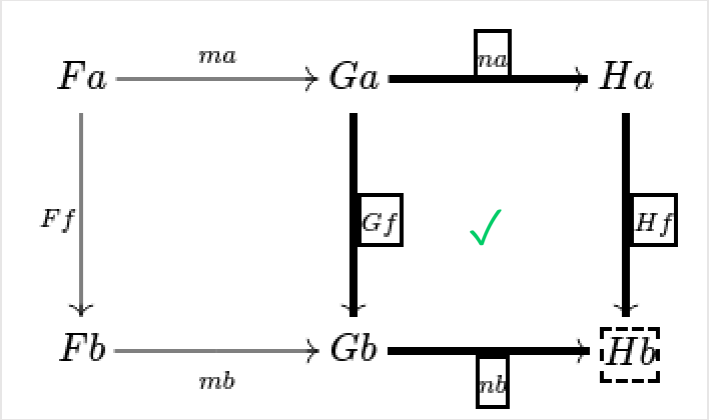
unnamed arrow

$$Hf \circ n_a = nb \circ \_$$

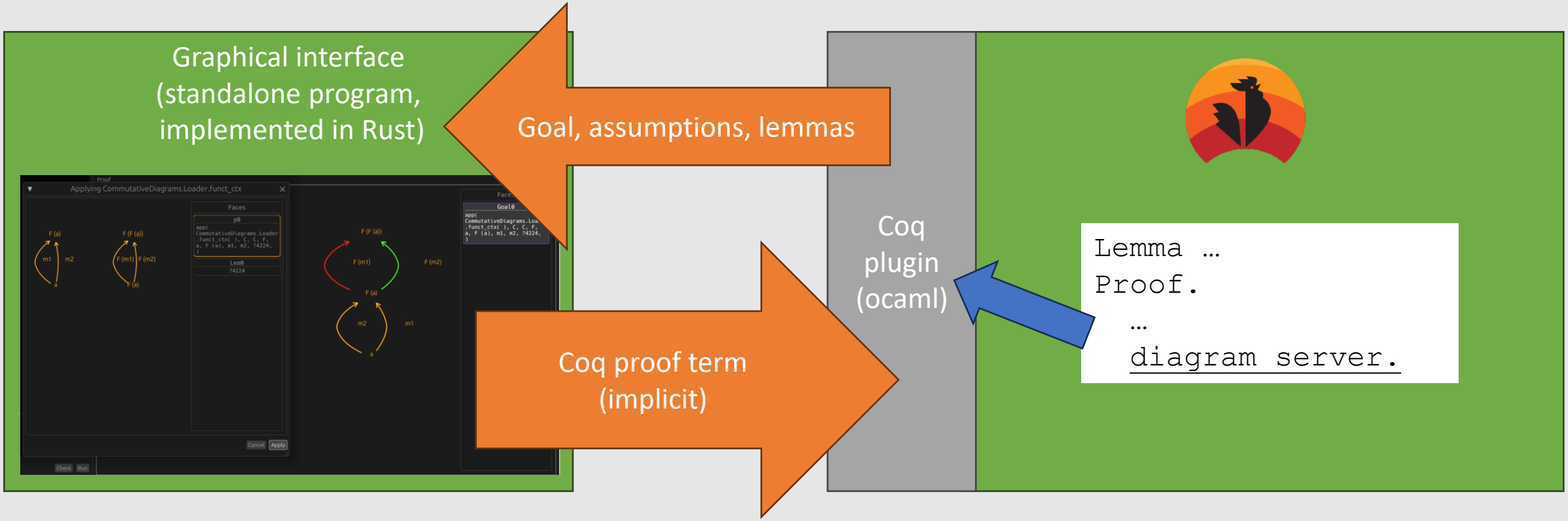


⇒ The diagram gets completed in YADE:

- The unnamed arrow is instantiated by Coq's inferred instantiation
- The proof node is marked as validated (indicated by a green ✓)



Another related software for mechanisation: Luc Chabassier's interface[1] for diagrammatic proofs



- Automatic layout (limited manual editing)
- Proof tactics (including invoking a lemma)

[1]https://github.com/dwarfmaster/commutative-diagrams

Demo[1] of YADE

(Based on the category theory library of Hierarchy Builder + custom tactics & notations)

A distributive law $\delta: TS \Rightarrow ST$ between two monads S and T induces a monad structure on ST.

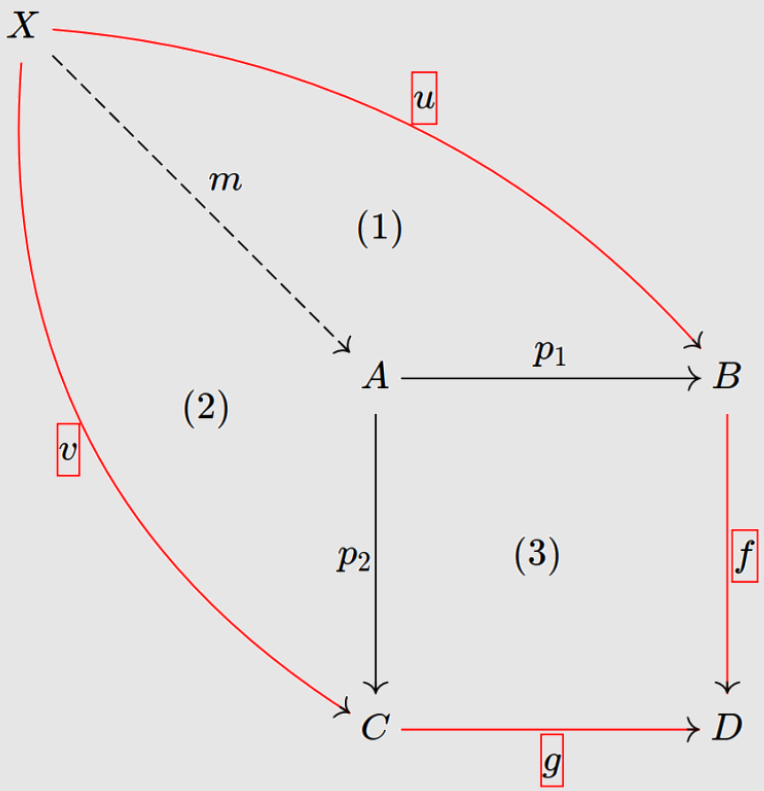
Let us show that the induced multiplication $STST \overset{S\delta T}{\to} SSTT \overset{\mu^S \mu^T}{\to} ST$ is associative.

[1] https://github.com/amblafont/vscode-yade-example

Proof generation: sketch of the algorithm

1) Save all subdiagrams as rewrite rules: "top right branch" → "bottom left branch"

$$(1)\ u \to p_1 \circ m$$
$$(2)\ p_2 \circ m \to v$$
$$(3)\ f \circ p_1 \to g \circ p_2$$



2) Identify the top right branch of the outer diagram.

$$f \circ u$$

3) Use greedily the rewrite rules until reaching the bottom left branch

$$f \circ u \to^{(1)} f \circ p_1 \circ m \to^{(3)} g \circ p_2 \circ m \to^{(2)} g \circ v$$

4) Remember the rewrite steps and generate the coq proof script accordingly.
If a Coq proof is provided inside a subdiagram, use it to justify the rewrite step.