A diagram editor to mechanise categorical proofs

Ambroise Lafont

JFLA, 30 January 2024

Packaging

A standalone desktop program

A web app that runs locally in your browser (without mechanisation features)

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

Naming convention

Yet Another Diagram Editor

(not to be confused with *Yet Another Dynamical Engine!*)

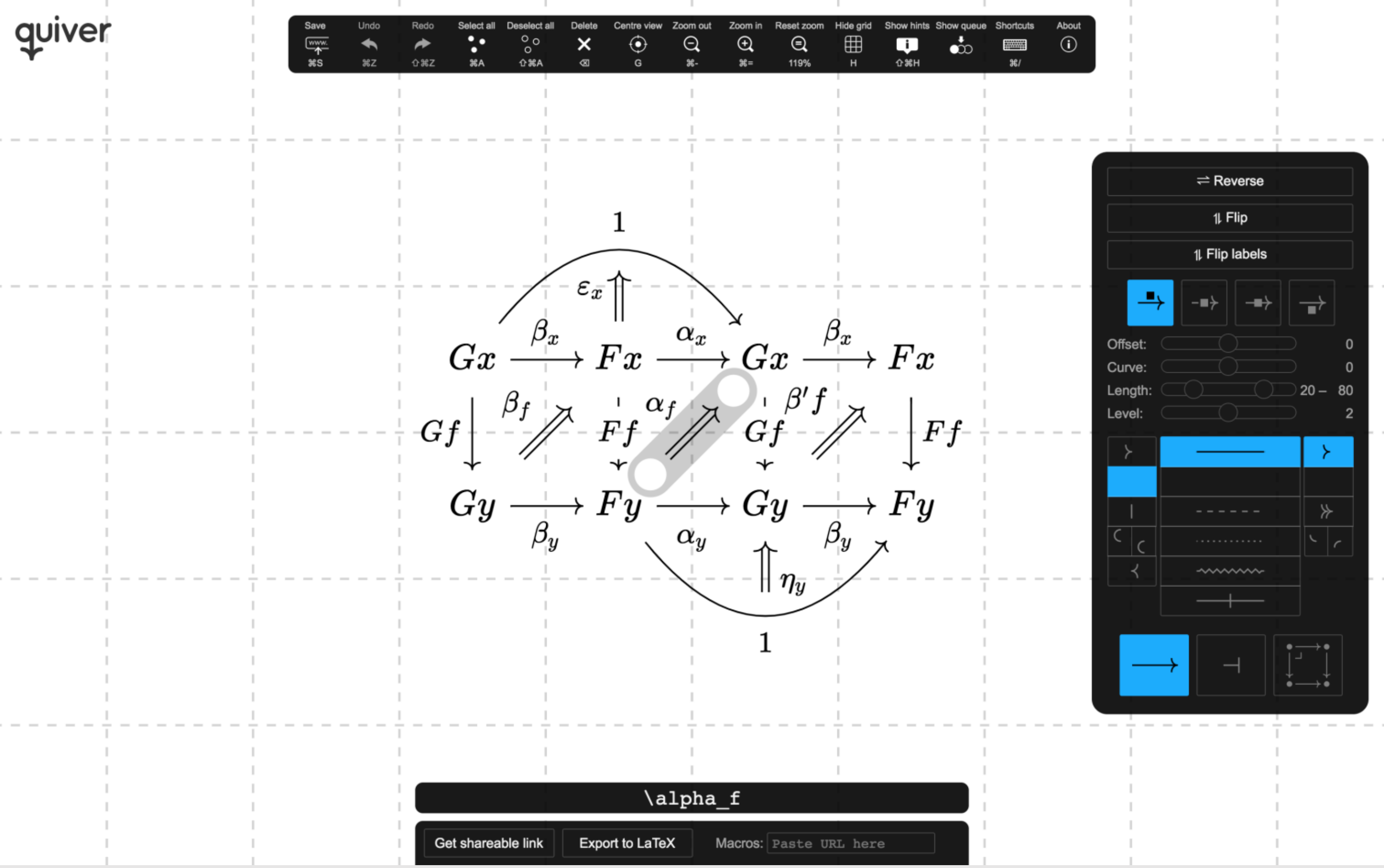
I will refer to the editor as **YADE**, or **Coreact-YADE**

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

[1] https://coreact.wiki/

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."



$$\begin{tikzcd}
{} & {} & {} & {} \\
Gx \arrow[r, "\beta_x"] \arrow[d, "Gf"'] \arrow[rr, bend left, "1"] & Fx \arrow[r, "\alpha_x"] \arrow[d, "Ff"] \arrow[u, Rightarrow, "\varepsilon_x"] & Gx \arrow[r, "\beta_x"] \arrow[d, "Gf"] & Fx \arrow[d, "Ff"] \\
Gy \arrow[r, "\beta_y"'] \arrow[ur, Rightarrow, "\beta_f"] & Fy \arrow[r, "\alpha_y"'] \arrow[rr, bend right, "1"'] \arrow[ur, Rightarrow, "\alpha_f"] & Gy \arrow[r, "\beta_y"'] \arrow[ur, Rightarrow, "\beta' f"] & Fy \\
{} & {} & {} \arrow[u, Rightarrow, "\eta_y"'] & {}
\end{tikzcd}$$

Offset: 0

Curve: 0

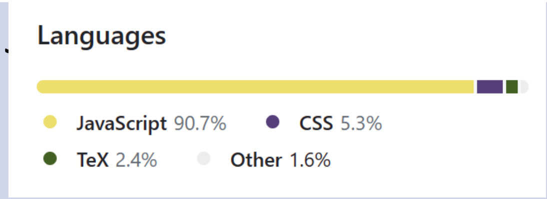
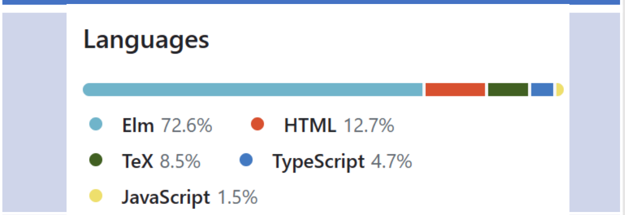
Length: 20 – 80

Level: 2

```
\alpha_f
```

Comparison with quiver

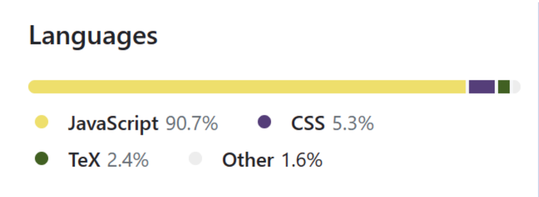
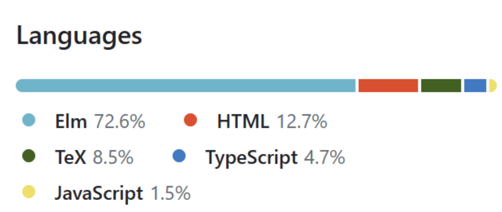
About the same size (around 10k of LoC)

	Quiver	YADE
Programming Languages	 Languages ● JavaScript 90.7% ● CSS 5.3% ● TeX 2.4% ● Other 1.6%	 Languages ● Elm 72.6% ● HTML 12.7% ● TeX 8.5% ● TypeScript 4.7% ● JavaScript 1.5%
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	 Languages ● JavaScript 90.7% ● CSS 5.3% ● TeX 2.4% ● Other 1.6%	 Languages ● Elm 72.6% ● HTML 12.7% ● TeX 8.5% ● TypeScript 4.7% ● JavaScript 1.5%
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

YADE
(standalone version)

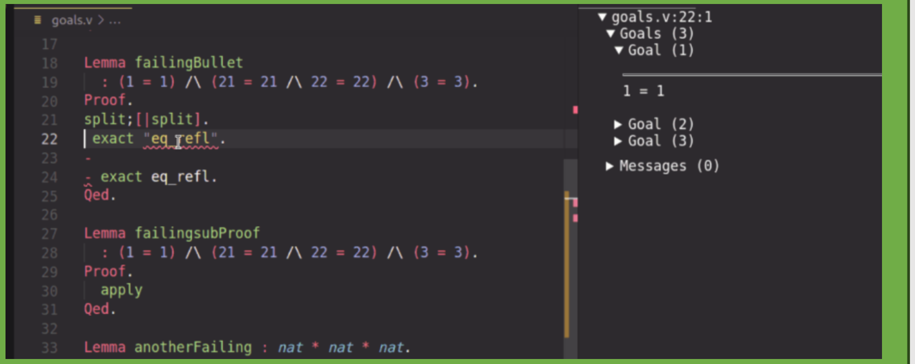
$$\begin{tikzcd}
Fa \arrow[r, "Ff"] \arrow[d, "n_a"'] \arrow[dr, phantom, "\checkmark"] & Fb \arrow[r, "Fg"] \arrow[d, "n_b"] \arrow[dr, phantom, "\checkmark"] & Fc \arrow[d, "n_c"] \\
Ga \arrow[r, "Gf"'] & Gb \arrow[r, "Gg"'] & Gc
\end{tikzcd}$$

Show diagram under cursor

Generate proofs

Custom vscode extension (building upon coq-lsp)

Visual Studio Code



```
goals.v > ...

Lemma failingBullet
  : (1 = 1) /\ (21 = 21 /\ 22 = 22) /\ (3 = 3).
Proof.
split;[|split].
exact "eq_refl".
-
- exact eq_refl.
Qed.

Lemma failingsubProof
  : (1 = 1) /\ (21 = 21 /\ 22 = 22) /\ (3 = 3).
Proof.
  apply
Qed.

Lemma anotherFailing : nat * nat * nat.
```

```
▼ goals.v:22:1
 ▼ Goals (3)
  ▼ Goal (1)
   ─────────────
   1 = 1

  ▶ Goal (2)
  ▶ Goal (3)
 ▶ Messages (0)
```

(+ Coq library for custom notations)

Natural transformations compose:

Diagrammatic proof

$$\begin{tikzcd}
Fc_1 \arrow[r, "\alpha_{c_1}"] \arrow[d, "Ff"'] & Gc_1 \arrow[r, "\beta_{c_1}"] \arrow[d, phantom, "Hf \circ \beta_{c_1} \circ \alpha_{c_1} \overset{?}{=} \beta_{c_2} \circ \alpha_{c_2} \circ Ff"] & Hc_1 \arrow[d, "Hf"] \\
Fc_2 \arrow[r, "\alpha_{c_2}"'] & Gc_2 \arrow[r, "\beta_{c_2}"'] & Hc_2
\end{tikzcd}$$

$$\begin{tikzcd}
{\color{blue}Fc_1} \arrow[r, "{\color{blue}\alpha_{c_1}}"] \arrow[d, "{\color{blue}Ff}"'] \arrow[dr, phantom, "="] & {\color{purple}Gc_1} \arrow[r, "{\color{red}\beta_{c_1}}"] \arrow[d, dashed, "{\color{purple}Gf}"] \arrow[dr, phantom, "="] & {\color{red}Hc_1} \arrow[d, "{\color{red}Hf}"] \\
{\color{blue}Fc_2} \arrow[r, "{\color{blue}\alpha_{c_2}}"'] & {\color{purple}Gc_2} \arrow[r, "{\color{red}\beta_{c_2}}"'] & {\color{red}Hc_2}
\end{tikzcd}$$

Automatic generation?

Computer-friendly proof

$$
\begin{aligned}
&Hf \circ \beta_{c_1} \circ \alpha_{c_1} \\
&= \beta_{c_2} \circ Gf \circ \alpha_{c_1} \\
&= \beta_{c_2} \circ \alpha_{c_2} \circ Ff
\end{aligned}
$$

Architecture

Diagram editor
(standalone version)

$$\begin{tikzcd}
Fa \arrow[r, "Ff"] \arrow[d, "n_a"'] \arrow[dr, phantom, "\checkmark"] & Fb \arrow[r, "Fg"] \arrow[d, "n_b"'] \arrow[dr, phantom, "\checkmark"] & Fc \arrow[d, "n_c"'] \\
Ga \arrow[r, "Gf"'] & Gb \arrow[r, "Gg"'] & Gc
\end{tikzcd}$$

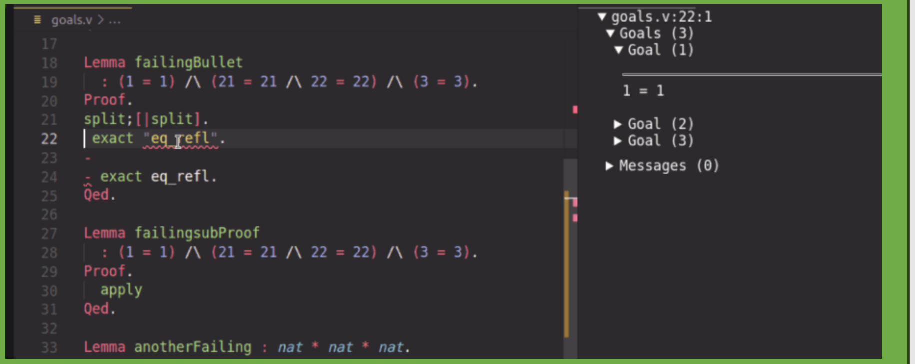
Show diagram under cursor

Generate proofs

Construct diagrams step by step

Custom vscode extension (building upon coq-lsp)

Visual Studio Code



goals.v > ...

```
Lemma failingBullet
  : (1 = 1) /\ (21 = 21 /\ 22 = 22) /\ (3 = 3).
Proof.
split;[|split].
exact "eq_refl".
-
- exact eq_refl.
Qed.

Lemma failingsubProof
  : (1 = 1) /\ (21 = 21 /\ 22 = 22) /\ (3 = 3).
Proof.
  apply
Qed.

Lemma anotherFailing : nat * nat * nat.
```

```
▼ goals.v:22:1
 ▼ Goals (3)
  ▼ Goal (1)

    1 = 1

  ▶ Goal (2)
  ▶ Goal (3)
 ▶ Messages (0)
```

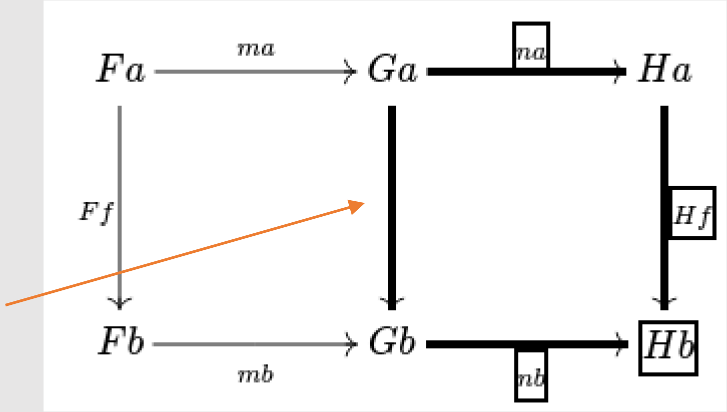
(+ Coq library for custom notations)

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:



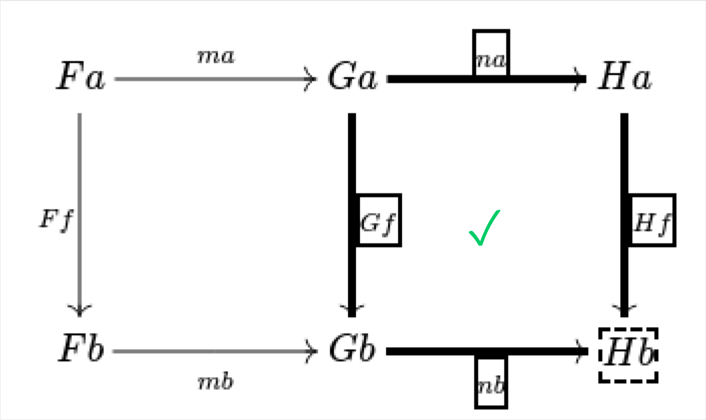
$$\begin{tikzcd}
Fa \arrow[r, "ma"] \arrow[d, "Ff"'] & Ga \arrow[r, "\boxed{na}"] \arrow[d] & Ha \arrow[d, "\boxed{Hf}"] \\
Fb \arrow[r, "mb"'] & Gb \arrow[r, "\boxed{nb}"'] & \boxed{Hb}
\end{tikzcd}$$

unnamed arrow

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

⇒ The diagram gets completed in YADE:

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



$$\begin{tikzcd}
Fa \arrow[r, "ma"] \arrow[d, "Ff"'] & Ga \arrow[r, "\boxed{na}"] \arrow[d, "\boxed{Gf}"] \arrow[dr, phantom, "\checkmark"] & Ha \arrow[d, "\boxed{Hf}"] \\
Fb \arrow[r, "mb"'] & Gb \arrow[r, "\boxed{nb}"'] & \boxed{Hb}
\end{tikzcd}$$

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