

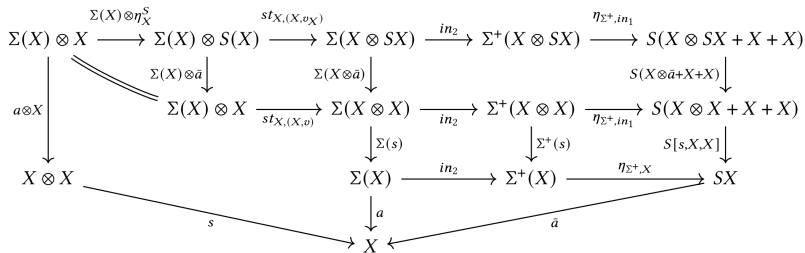
A categorical diagram editor to help formalising commutation proofs

Ambroise Lafont

University of Cambridge (postdoc)

GReTA-ExACT online workgroup, March 2022

- **Diagrammatic reasoning** is useful



✗ Theorem provers (e.g., Coq) are **text-based**

⇒ This talk: a tool to bridge the gap

Yet Another Diagram Editor (YADE)

Proof assistant
(Coq)



Lemma commutes :

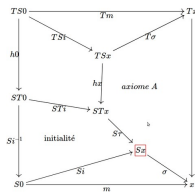
$$n_y \circ F f = G f \circ n_x.$$

Proof / statement display



Proof generation

Diagram editor
(YADE)



- Available online¹: run by your browser
- Written in **Elm** (~6000 LoC)



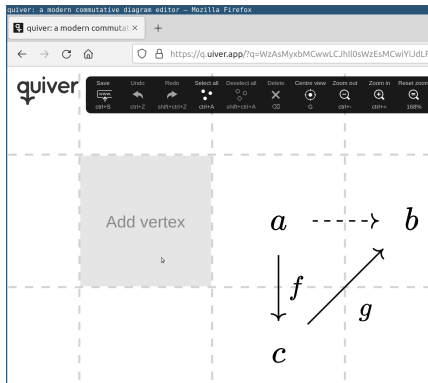
“A delightful language for reliable web applications.”

- Functional PL
- Compiles to js

¹<https://amblafont.github.io/graph-editor/index.html>

Comparison with Quiver

“quiver¹ is 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.”



Quiver features:

- More display options (colors, ...)
- Rigid grid layout
- Export to LaTeX
- ✗ No proof generation

Note: YADE exports to quiver

¹<https://q.uiver.app/>

- Short presentation of YADE's features that help formalisation
- Demo

Plan

- 1 Parse equations
- 2 Forward reasoning
- 3 Backward reasoning
- 4 Demo

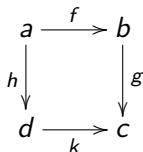
Plan

- 1 Parse equations
- 2 Forward reasoning
- 3 Backward reasoning
- 4 Demo

A simple format for equations

$$a \text{ -- } f \text{ --> } b \text{ -- } g \text{ --> } c = a \text{ -- } h \text{ --> } d \text{ -- } k \text{ --> } c$$

\Downarrow



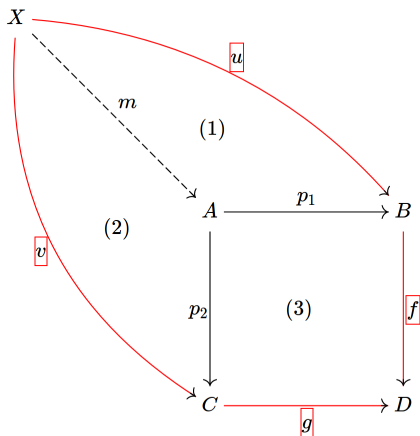
Load an equation from Coq

Specific Coq notations to comply to the above format

Plan

- 1 Parse equations
- 2 Forward reasoning
- 3 Backward reasoning
- 4 Demo

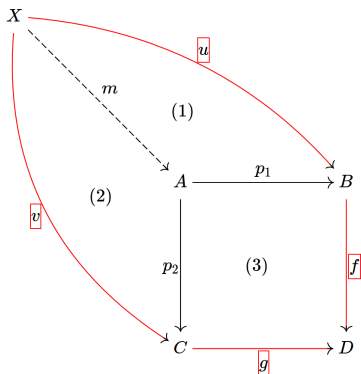
Diagrams are proofs



Generated formal proof

If inner subdiagrams (1)-(3) commute,
then the **outer diagram** commutes.

Algorithm for proof generation



- 1 Identify all subdiagrams
- 2 Start from one branch of the outer diagram
- 3 Repeatedly “apply” subdiagrams to progress, until reaching the other branch.

Target: the UniMath mathematical library

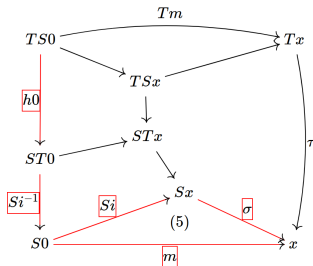
History

- started in 2014 with Voevodsky's repository Foundations
- Large Coq mathematical library (~300 000 lines)
 - ~ two thirds on (bi)category theory
 - Verbose style
- Inconsistent: universe level checks are disabled (\Rightarrow impredicative encoding of quotients)

Why targeting this library?

- I use UniMath in my formalisation projects.
- The implementation could be easily adapted to another target.

Commutation proofs in UniMath

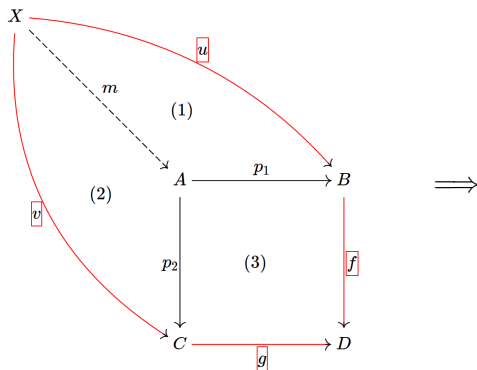


Applying¹ (5) to the bottom left branch

$$\begin{array}{lcl}
 (h_0 \cdot Si^{-1}) \cdot m & \left. \begin{array}{l} \\ \end{array} \right\} & \text{rewrite assoc'} \\
 h_0 \cdot (Si^{-1} \cdot m) & \left. \begin{array}{l} \\ \end{array} \right\} & \text{apply cancel_precomposition} \\
 Si^{-1} \cdot m & \left. \begin{array}{l} \\ \end{array} \right\} & \text{apply cancel_precomposition} \\
 m & \left. \begin{array}{l} \\ \end{array} \right\} & \text{apply (5)} \\
 Si \cdot \sigma & &
 \end{array}$$

¹rewrite (5) sometimes works but is hard to maintain.

Generated Coq script



```
Goal { u · f = v · g }.

assert(eq : { u = m · p1 }).
{ admit. }
etrans.
{
  apply cancel_postcomposition.
  apply eq.
}
clear eq.
assert(eq : { p1 · f = p2 · g }).
{ admit. }
etrans.
{
  repeat rewrite assoc'.
  apply cancel_precomposition.
  repeat rewrite assoc.
  apply eq.
}
repeat rewrite assoc.
clear eq.
assert(eq : { m · p2 = v }).
{ admit. }
etrans.
{
  apply cancel_postcomposition.
  apply eq.
}
clear eq.
apply idpath.
Qed.
```

Commutation of subdiagrams are explicitly asserted and **admitted**.

Plan

- 1 Parse equations
- 2 Forward reasoning
- 3 Backward reasoning
- 4 Demo

Forward vs Backward reasoning in Coq

$n : G \Rightarrow F$ natural.

How to rewrite $n_x \cdot Ff$ into $Gf \cdot n_y$?

Forward reasoning in Coq

- 1 State naturality specialised to $f : x \rightarrow y$

$$n_x \cdot Ff = Gf \cdot n_y$$

- 2 Prove the above equation by naturality
- 3 Apply the equation

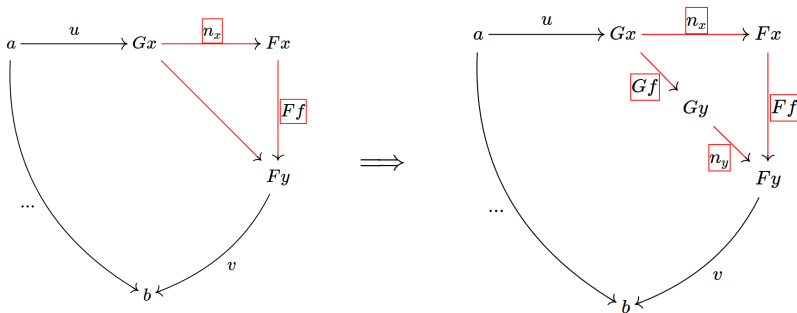
Backward reasoning in Coq

Directly apply naturality of n

\Rightarrow No need to provide the r.h.s $Gf \cdot n_y$

Backward reasoning with YADE

- 1 Generate Coq script that focuses on the subterm
 - 2 Manually apply the lemma, in Coq
 - 3 Copy and paste the result in YADE
- ⇒ The diagram is completed automatically



Plan

- 1 Parse equations
- 2 Forward reasoning
- 3 Backward reasoning
- 4 Demo

Demo: Associativity of the “composite” multiplication

Let

- $\delta : ST \rightarrow TS$ be a monadic distributive law.
- $R := TS$
- $\mu' : RR \rightarrow R$ defined in the only sensible way.

Then,

$$\begin{array}{ccc} RRR & \xrightarrow{R\mu'} & RR \\ \mu'R \downarrow & & \downarrow \mu' \\ RR & \xrightarrow{\mu'} & R \end{array}$$