A categorical diagram editor to help formalising commutation proofs

Ambroise Lafont

University of Cambridge (postdoc)

GReTA-ExACT online workgroup, March 2022

Motivation

- **Diagrammatic reasoning** is useful

$$\begin{tikzcd}
\Sigma(X) \otimes X \arrow[r, "\Sigma(X) \otimes \eta^S_X"] \arrow[dd, "a \otimes X"'] \arrow[dr, equal] & \Sigma(X) \otimes S(X) \arrow[r, "st_{X,(X,v_X)}"] \arrow[d, "\Sigma(X) \otimes \bar{a}"'] & \Sigma(X \otimes SX) \arrow[r, "in_2"] \arrow[d, "\Sigma(X \otimes \bar{a})"'] & \Sigma^+(X \otimes SX) \arrow[r, "\eta_{\Sigma^+, in_1}"] & S(X \otimes SX + X + X) \arrow[d, "S(X \otimes \bar{a} + X + X)"'] \\
 & \Sigma(X) \otimes X \arrow[r, "st_{X,(X,v)}"'] & \Sigma(X \otimes X) \arrow[r, "in_2"'] \arrow[d, "\Sigma(s)"] & \Sigma^+(X \otimes X) \arrow[r, "\eta_{\Sigma^+, in_1}"'] \arrow[d, "\Sigma^+(s)"] & S(X \otimes X + X + X) \arrow[d, "S[s,X,X]"'] \\
X \otimes X \arrow[drr, "s"'] & & \Sigma(X) \arrow[r, "in_2"] \arrow[d, "a"] & \Sigma^+(X) \arrow[r, "\eta_{\Sigma^+, X}"] & SX \arrow[dll, "\bar{a}"] \\
 & & X & &
\end{tikzcd}$$

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

$\Rightarrow$ need a device to bridge the gap

Yet Another Diagram Editor (YADE)

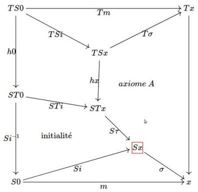
Proof assistant (Coq)

Diagram editor (YADE)

Proof / statement display

Proof generation

```
Lemma commutes :
  n_y ∘ F f = G f ∘ n_x.
```



$$\begin{tikzcd}
TS0 \arrow[rrr, "Tm"'] \arrow[dr, "TSi"] \arrow[dd, "h0"'] & & & Tx \arrow[dddd] \\
& TSx \arrow[urr, "T\sigma"'] \arrow[d, "hx"'] \arrow[dr, phantom, "axiome\ A"] & & \\
ST0 \arrow[r, "STi"'] \arrow[dd, "Si^{-1}"'] \arrow[dr, phantom, "\text{initialité}"] & STx \arrow[dr, "S\tau"'] & & \\
& & \boxed{Sx} \arrow[dr, "\sigma"] & \\
S0 \arrow[urr, "Si"'] \arrow[rrr, "m"'] & & & x
\end{tikzcd}$$

More about YADE

- Available online[1]
- Written in **Elm** (~6000 LoC)

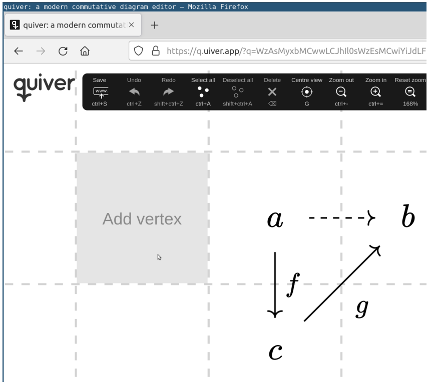
"A delightful language for reliable web applications."

- Functional PL
- Compiles to js

[1] `https://amblafont.github.io/graph-editor/index.html`

Comparison with Quiver

"quiver[1] 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: a modern commutative diagram editor — Mozilla Firefox
quiver: a modern commutat
https://q.uiver.app/?q=WzAsMyxbMCwwLCJhIl0sWzEsMCwiYiJdLF
quiver
Save ctrl+S
Undo ctrl+Z
Redo shift+ctrl+Z
Select all ctrl+A
Deselect all shift+ctrl+A
Delete
Centre view G
Zoom out ctrl+-
Zoom in ctrl+=
Reset zoom 168%
Add vertex

$$\begin{tikzcd} a \arrow[r, dashed] \arrow[d, "f"] & b \\ c \arrow[ur, "g"'] & \end{tikzcd}$$

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/

This talk

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

Plan

Plan

A simple format for equations

```
a -- f -> b -- g -> c  =  a -- h -> d -- k -> c
```

$$\Downarrow$$

$$\begin{tikzcd} a \arrow[r, "f"] \arrow[d, "h"'] & b \arrow[d, "g"] \\ d \arrow[r, "k"'] & c \end{tikzcd}$$

From Coq

Specific notations to comply to the above format

Plan

Diagrams are proofs

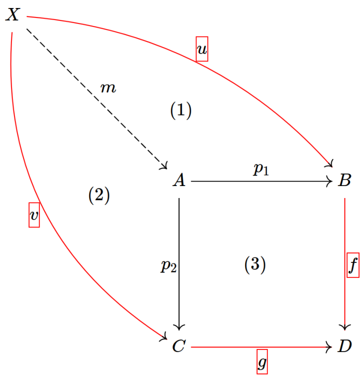
$$\begin{tikzcd}
X \arrow[drr, bend left, "\boxed{u}"] \arrow[ddr, bend right, "\boxed{v}"'] \arrow[dr, dashed, "m"] \arrow[drr, phantom, "(1)"] \arrow[ddr, phantom, "(2)"] & & \\
& A \arrow[r, "p_1"] \arrow[d, "p_2"'] \arrow[dr, phantom, "(3)"] & B \arrow[d, "\boxed{f}"] \\
& C \arrow[r, "\boxed{g}"'] & D
\end{tikzcd}$$

Generated formal proof

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

Algorithm for proof generation

$$\begin{tikzcd}
X \arrow[dr, dashed, "m"] \arrow[drr, bend left, "\boxed{u}"] \arrow[ddr, bend right, "\boxed{v}"'] & & \\
 & A \arrow[r, "p_1"] \arrow[d, "p_2"'] & B \arrow[d, "\boxed{f}"] \\
 & C \arrow[r, "\boxed{g}"'] & D
\end{tikzcd}$$



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! (Type : Type)

Why targeting this library?

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

Commutation proofs in UniMath

$$\begin{tikzcd}
TS0 \arrow[rrr, bend left, "Tm"] \arrow[dr] \arrow[dd, "\boxed{h0}"'] & & & Tx \arrow[dddd, bend left, "\tau"] \\
& TSx \arrow[urr] \arrow[d] & & \\
ST0 \arrow[r] \arrow[dd, "\boxed{Si^{-1}}"'] & STx \arrow[dr] & & \\
& & Sx \arrow[dr, "\boxed{\sigma}"] & \\
S0 \arrow[urr, "\boxed{Si}"] \arrow[rrr, "\boxed{m}"'] & & (5) & x
\end{tikzcd}$$

Applying[1] (5) to the bottom left branch

$$(h_0 \cdot Si^{-1}) \cdot m$$

rewrite assoc'

$$h_0 \cdot (Si^{-1} \cdot m)$$

apply cancel_precomposition

$$Si^{-1} \cdot m$$

apply cancel_precomposition

$$m$$

apply (5)

$$Si \cdot \sigma$$

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

Generated Coq script

$$\begin{tikzcd}
X \arrow[dr, dashed, "m"] \arrow[drr, bend left, "\boxed{u}"] \arrow[ddr, bend right, "\boxed{v}"'] \arrow[drr, phantom, "(1)"] \arrow[ddr, phantom, "(2)"] & & \\
& A \arrow[r, "p_1"] \arrow[d, "p_2"'] \arrow[dr, phantom, "(3)"] & B \arrow[d, "\boxed{f}"] \\
& C \arrow[r, "\boxed{g}"'] & D
\end{tikzcd}$$

$\Longrightarrow$

```
Goal { u · f = v · g }.

assert(eq : { u = m · p_1 }).
{ admit. }
etrans.
{
  apply cancel_postcomposition.
  apply eq.
}
clear eq.
assert(eq : { p_1 · f = p_2 · g }).
{ admit. }
etrans.
{
  repeat rewrite assoc'.
  apply cancel_precomposition.
  repeat rewrite assoc.
  apply eq.
}
repeat rewrite assoc.
clear eq.
assert(eq : { m · p_2 = v }).
{ admit. }
etrans.
{
  apply cancel_postcomposition.
  apply eq.
}
clear eq.
apply idpath.
Qed.
```

Commutation of subdiagrams are explicitly asserted and admitted.

Plan

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. Explicitely state the equation
2. Prove it by naturality of n
3. Apply the equation

Backward reasoning in Coq

1. Focus on $n_x \cdot Ff$
2. Apply naturality of n

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

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

$\Rightarrow$ The diagram is completed automatically

$$\begin{tikzcd}
a \arrow[r, "u"] \arrow[drr, bend right=60, "\dots"'] & Gx \arrow[r, "\boxed{n_x}"] \arrow[dr] & Fx \arrow[d, "\boxed{Ff}"] \\
 & & Fy \arrow[dl, bend left, "v"] \\
 & b &
\end{tikzcd}
\quad \Longrightarrow \quad
\begin{tikzcd}
a \arrow[r, "u"] \arrow[dddr, bend right=60, "\dots"'] & Gx \arrow[rr, "\boxed{n_x}"] \arrow[dr, "\boxed{Gf}"'] & & Fx \arrow[dd, "\boxed{Ff}"] \\
 & & Gy \arrow[dr, "\boxed{n_y}"'] & \\
 & & & Fy \arrow[dll, bend left, "v"] \\
 & b & &
\end{tikzcd}$$

Plan