Towards a proof assistant for diagrammatic reasoning

Ambroise Lafont

Coreact Meeting, 29 November 2024

Quick update on YADE

- Mechanisation features:

  YADE is now fully included in the vscode extension (no need to install it separately)

- New “Multiplayer” mode

  Requires running a server

  - Solution 1: vscode
    - run it locally (the server is embedded in YADE’s vscode extension)
    - share it over the network with LiveShare extension
  - Solution 2: run it on a web hosting platform (with node.js enabled)

    (e.g., glitch.io which has free tier)

- Future work (contributors welcome!)
  - Benchmark, e.g., formalise a monoidal coherence theorem
  - Improve the generation of Coq code
  - Make it more modular over the library of category
  - Make it work in vscode in the browser

    (coq-lsp now fully support coq in the browser!)

Project of a graphical proof assistant

(not implemented)

- No text interface
- Use it to prove that some statement holds in any *model*;
- The notion of model is entirely specified by the user

  Examples: a category with pullbacks, a 2-category …

  No built-in notion of models: the user must define what a category is

How to specify a model *graphically*?

How to specify a model?

1. Sorts

For graphs:

vertex and edge

For categories:

same + composition witness

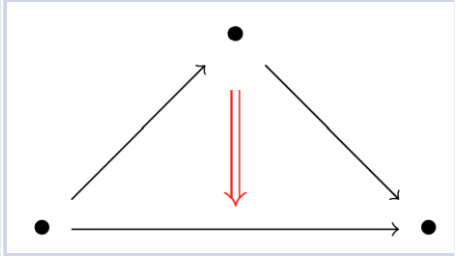
For categories with monomorphisms:

… + monomorphic tag

2. Rewriting rules

Operations (e.g. composition)

Equations

Sort Specification (drawn by the user)				

black: dependencies

red: graphical representation

Dependencies	(none)	2 vertices	1 edge	A triangle of edges
Graphical representation	Dot	Arrow	Hook	Double arrow

A model of this specification consists of 4 components:

a set Ob

for $x, y \in Ob$,
a set hom(x,y)

for f $\in \hom(x, y)$,
a set isMono(f)

for f $\in \hom(x, y)$,
g $\in \hom(y, z)$,
h $\in \hom(x, z)$,
a set isComp(f,g,h)

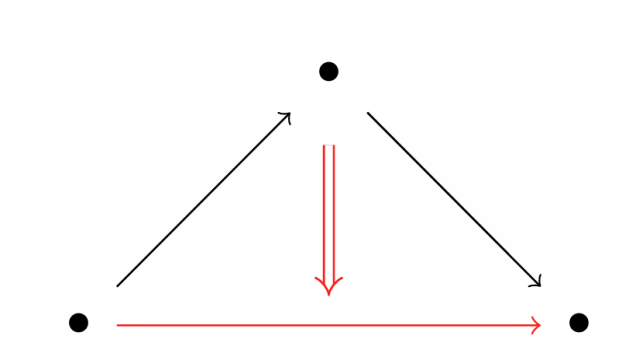
Specification of models

1. Sorts

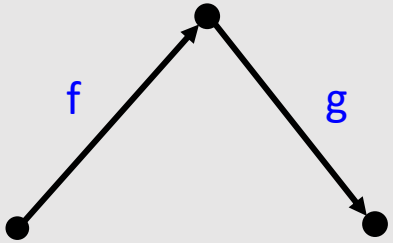
2. Rewriting rules

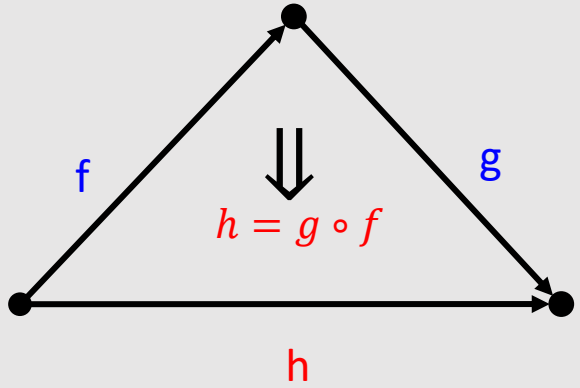
First example: composition of morphisms

Rewriting rule for composition



"Black → black + red"

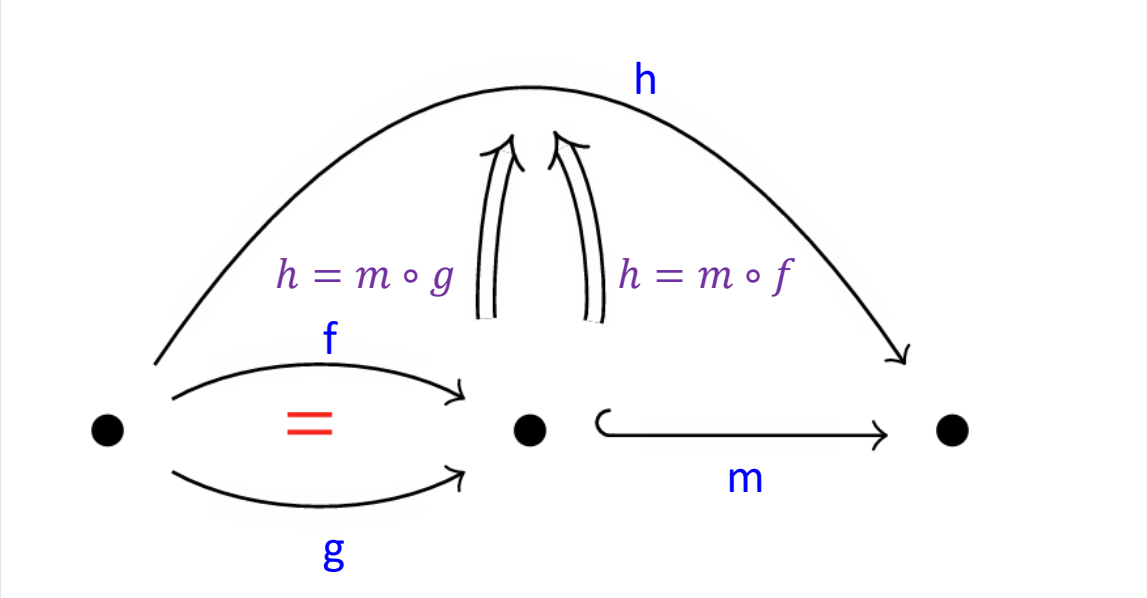




A model of this rule satisfies: "∀ black ∃ red"

More explicitly, $\forall x, y, z \in Ob, f \in \hom(x, y), g \in \hom(y, z),$
$\exists h \in \hom(x, z), p \in isComp(f, g, h)$

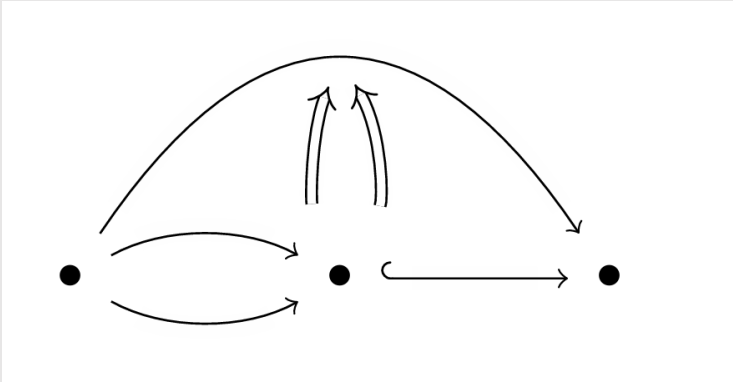
Rule for the monomorphic property



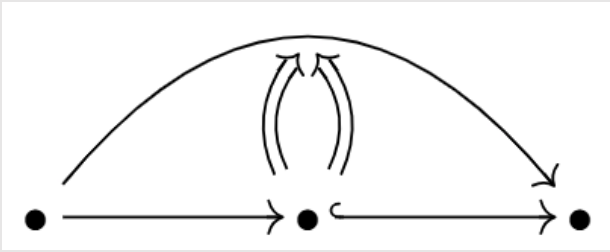
"Black → black + red"

equality merges elements

"$m \circ f = m \circ g \quad \Rightarrow \quad f = g$"



→



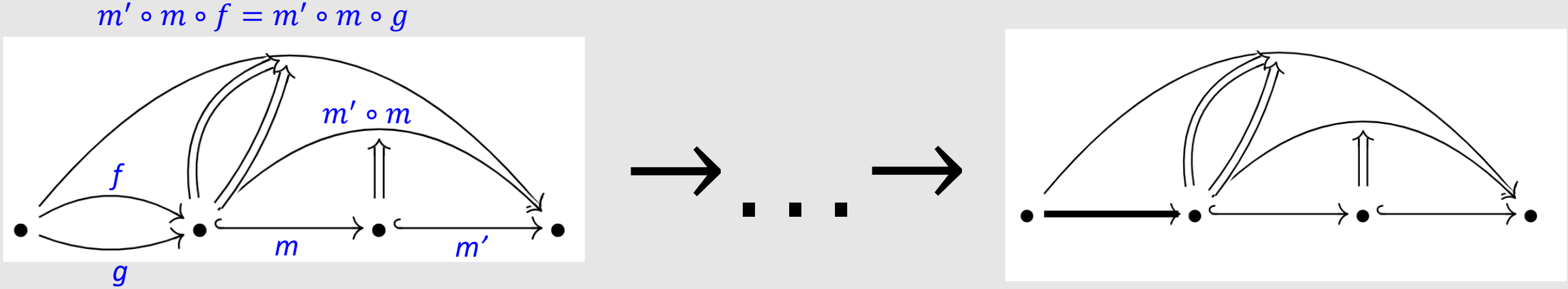
A proof that monomorphisms compose

$$\left.\begin{array}{l} m : x \hookrightarrow y \\ m' : y \hookrightarrow z \end{array}\right\} \text{ mono} \quad \Rightarrow \quad m' \circ m \text{ mono}$$

i.e., $\forall f, g : a \to x,$

$$m' \circ m \circ f = m' \circ m \circ g \quad \Rightarrow \quad f = g$$

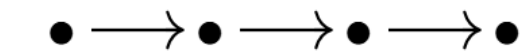
Proof:



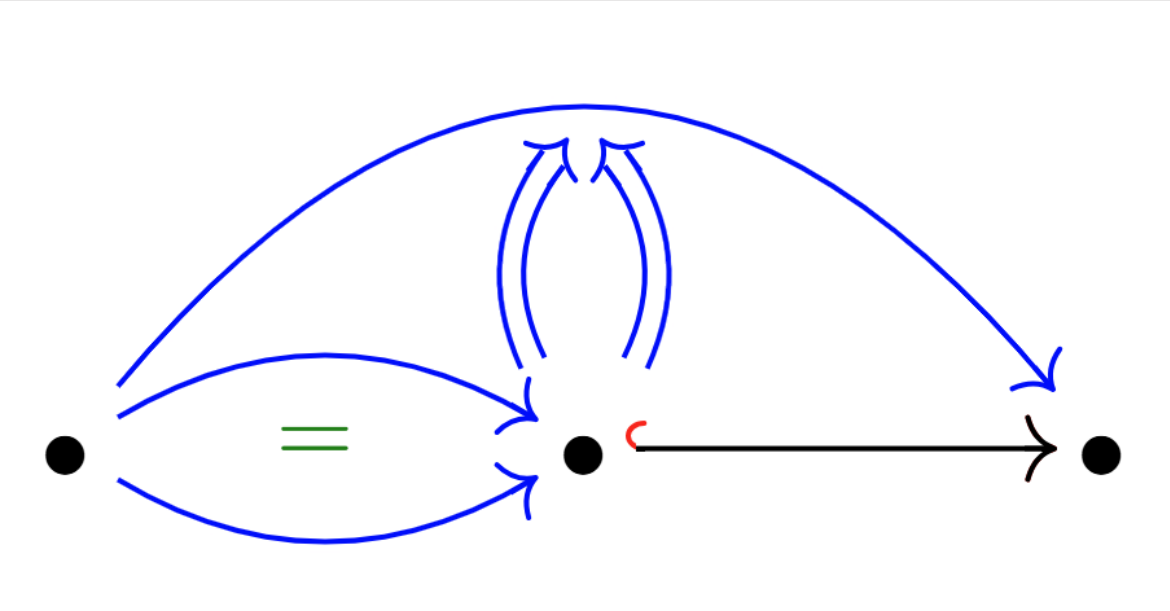
Logical semantics

- Formal logic: calculus of conditions
- Theory: all the declared rewriting rules, e.g.,

- A state is represented by a formula which expresses what has been proved so far



Rewriting rule for tagging a morphism as mono



"∀ black, (∀ blue ∃ green) ⇒ ∃ red"

If black + blue → black + blue + green,

Then, black → black + red

<u>Thesis</u>: This notion of (second-order) rule is enough to capture any desirable model specification.

Diagram chasing

Joint work with Assia Mahboubi and Matthieu Piquerez

Requires alternative semantics for rewriting rules!

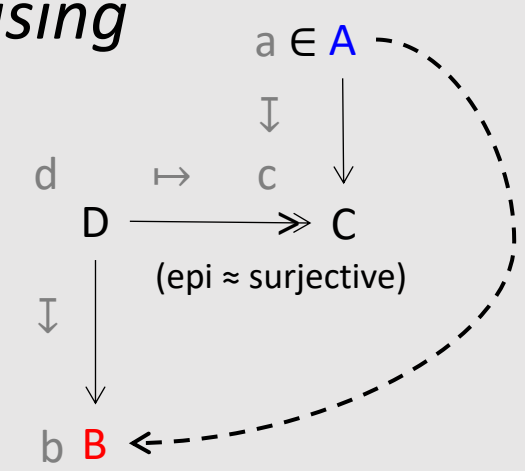
What is diagram chasing?

Pretending that any abelian category is a category of concrete modules[1].

Example:

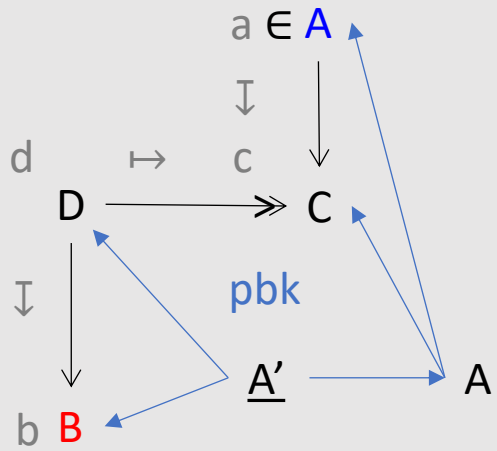
Construction of a morphism A → B in an (abstract) abelian category:

Reason as if A has elements, and find a way to map elements of A to B by *element chasing*



[1] Thanks to Mitchell's embedding theorem

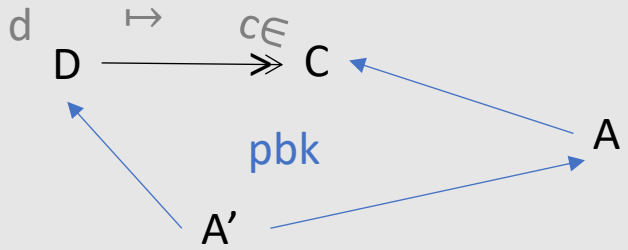
Element chasing can be interpreted as the (implicit) construction of a cone



How to model diagram chasing?

A new sort: for each C ∈ Ob, a set Elements(C)

Intuition:
an abelian category with a chosen object A (the apex of the cone) induces a model
with Elements(C) = hom(A,C)



Problem:
The chosen object can change during element chasing. We do not stay in the same model!

Solution (WIP):
A model comes with its notion of acceptable extensions.
(here, the epi A' ↠ A makes it acceptable)

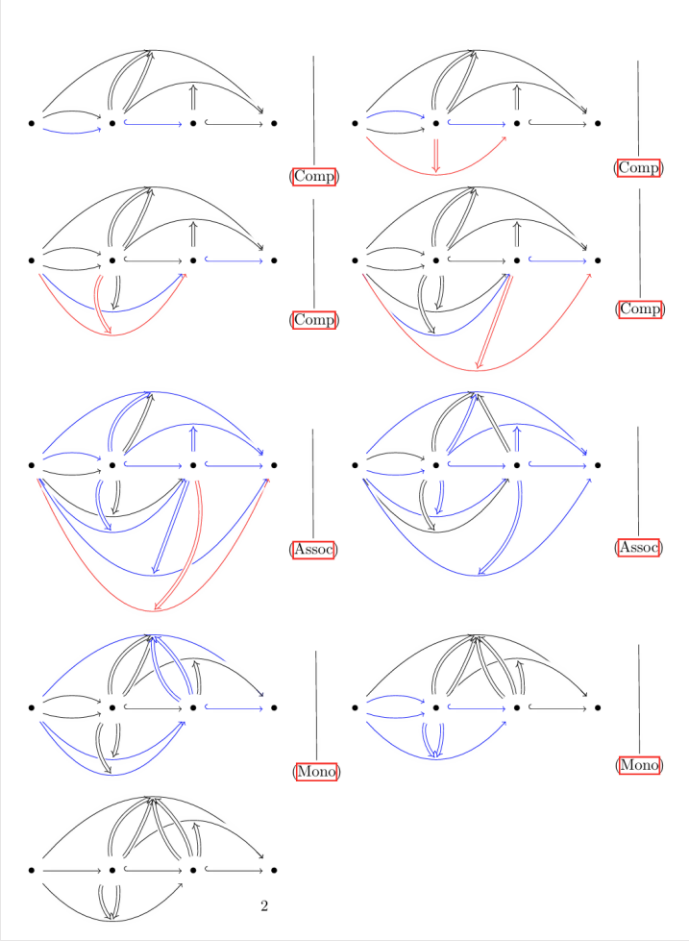
Conclusion

More details in the pdf draft[1] (WIP)

No implementation

How to make the context easier to manipulate?

proof that monomorphisms compose



[1] http://github.com/amblafont/diagrammatic-resoning-spec