# Categorical Dataflow

*Lenses and Optics as data structures for backpropagation*

*Bruno Gavranović*

*Cats4AI*
*24 October 2022*

# Recap

# Recap: Week 1

Category theory takes a bird’s eye view of mathematics. From high in the sky, details become invisible, but we can spot patterns that were impossible to detect from ground level.

Tom Leinster, Basic Category Theory

# Recap: Week 2

- We began to give concrete definitions

# Category: definition

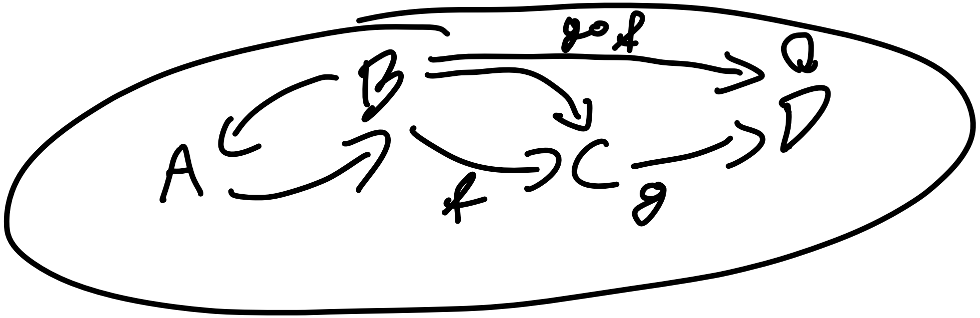
A category: a *universe* of **objects**, and **morphisms** between them, s.t.:

(The word *"universe"* is used here deliberately instead of *"set"*, to avoid *paradoxes*)

- For $f : A \rightarrow B$ and $g : B \rightarrow C$, there is a **composition**, $g \circ f : A \rightarrow C$
- For each object $A$, there is a unique **identity** morphism $\mathrm{id}_A : A \rightarrow A$
- For any morphism $f : A \rightarrow B$, it holds that $\mathrm{id}_B \circ f = f \circ \mathrm{id}_A = f$
- For any composable $f$, $g$, $h$, we have $h \circ (g \circ f) = (h \circ g) \circ f$

The collection of morphisms between $A$ and $B$ is often denoted $\mathrm{Hom}(A, B)$

(the "**hom-set**" from A to B)



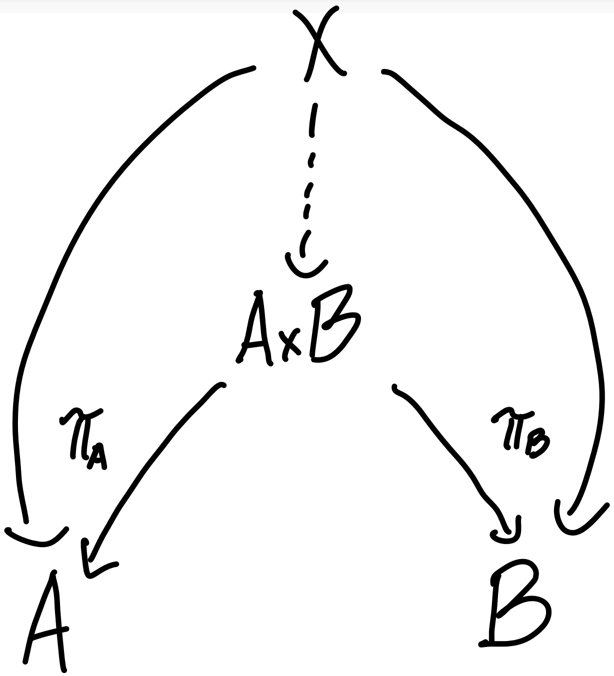
# Category: examples

- Set - sets and functions
- Rel - sets and relations
- Vect - vector spaces and linear transformations
- $\mathbb{R}$ - numbers and order relations
- Grp - single objects, group elements are morphisms

# Constructions inside categories

- Monomorphisms
- Epimorphisms
- Products
- Coproducts
- Exponential objects
- …

# Products



# *Pattern-hunting* for the product

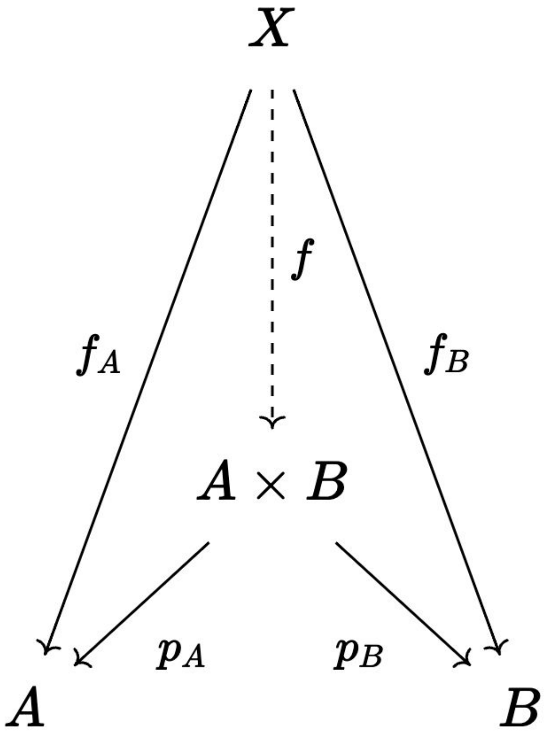
In some sense, $A×B$ is the "minimal" combination of data in $A$ and $B$. Therefore, if **any** other object $X$ presents projections, they must be somehow *decomposable* into a form that uses the "true" projections.

In other words, if $A×B$ is a product object, and any other object $X$ possesses morphisms $f_A : X \to A$ and $f_B : X \to B$, they **must** decompose through $A×B$, as:

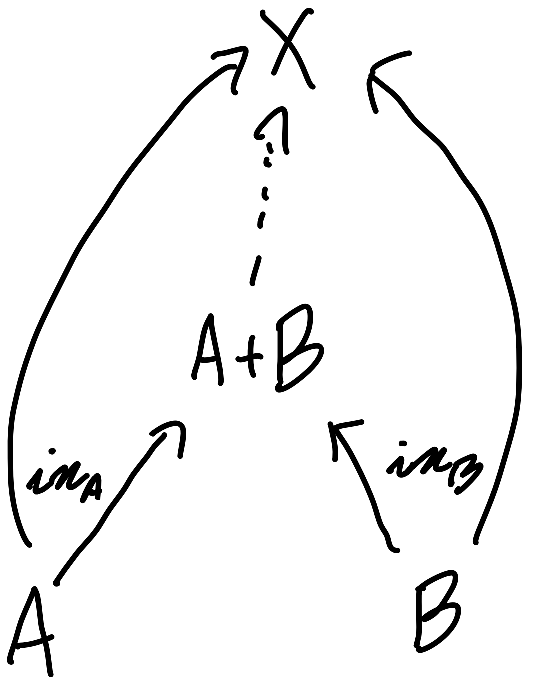
$f_A = p_A \circ f, f_B = p_B \circ f$

With $f : X \to A×B$ being a unique morphism.

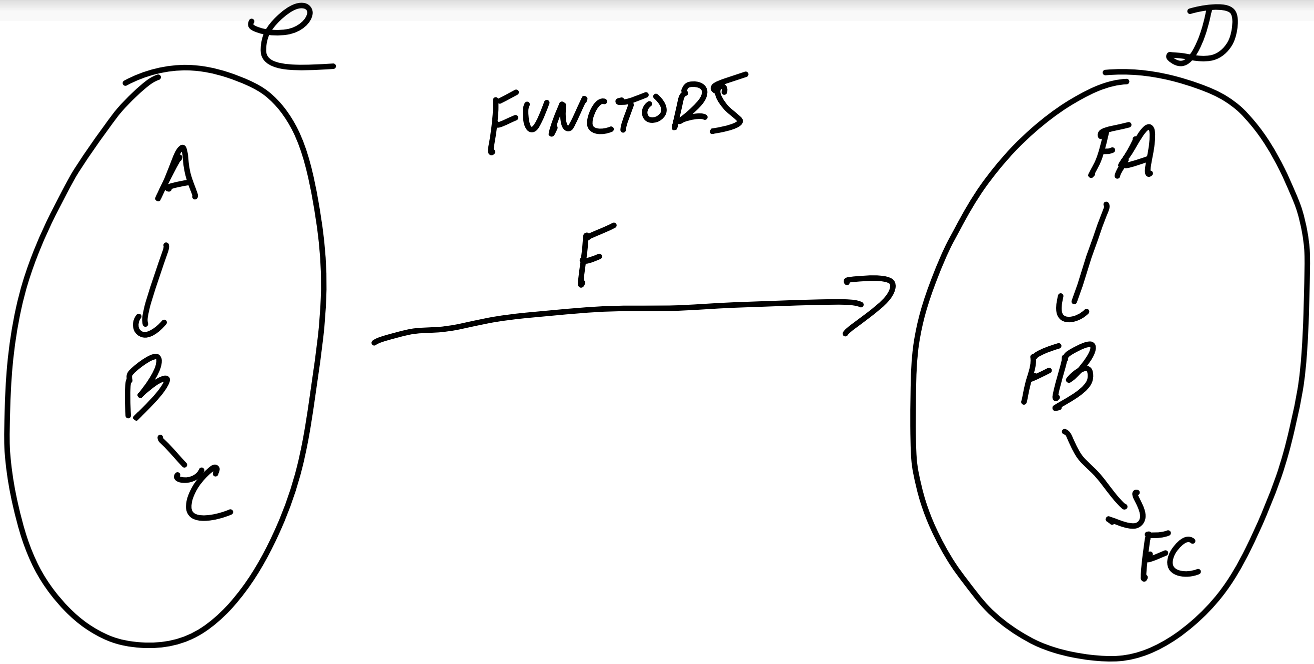
(Note, when $X=A×B$, trivially, $f = \mathrm{id}_{A×B}$)



# Coproducts – REVERSING THE ARROWS



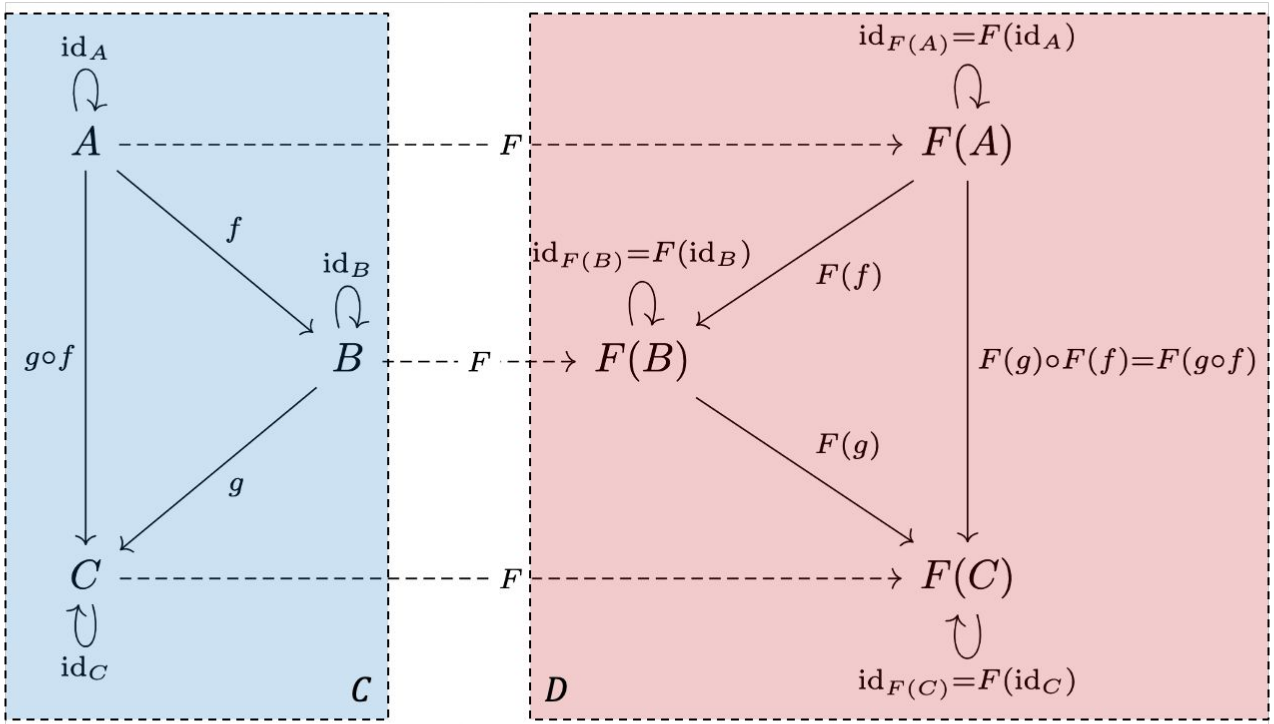
# Functors



# Functors, pictorially

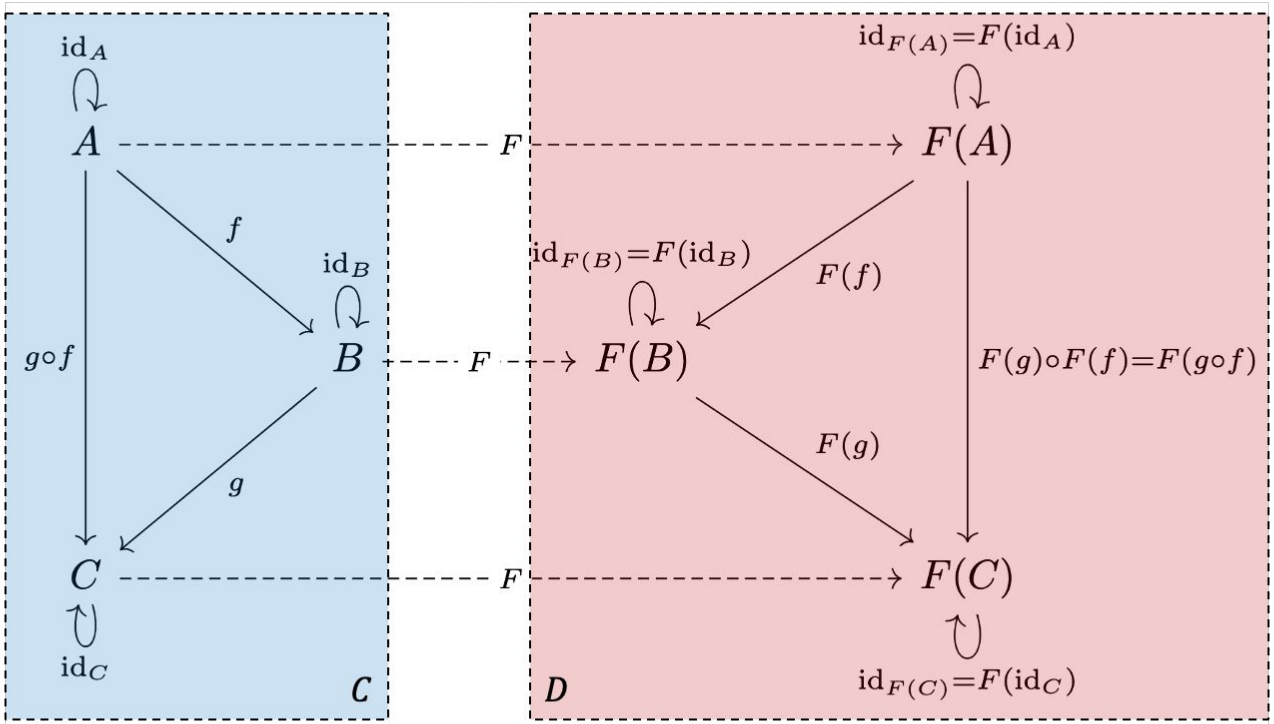
$F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$ for any object $A$ in **C**

$F(g \circ f) = F(g) \circ F(f)$ for any composable morphisms $f$ and $g$ in **C**

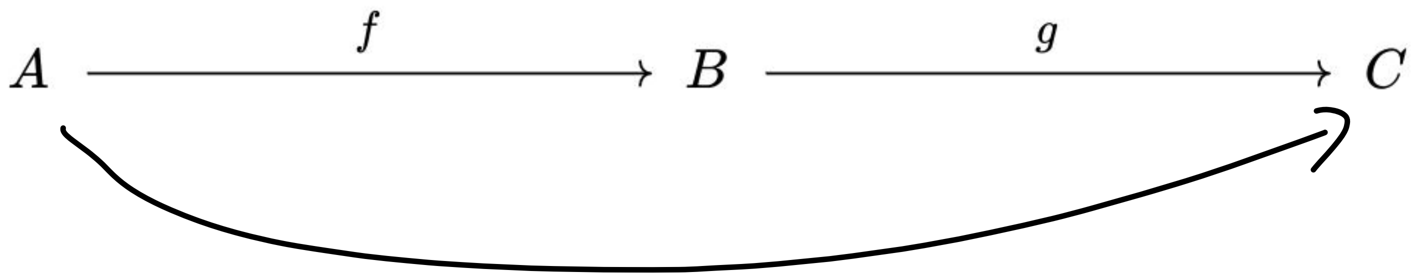


# Functors, pictorially

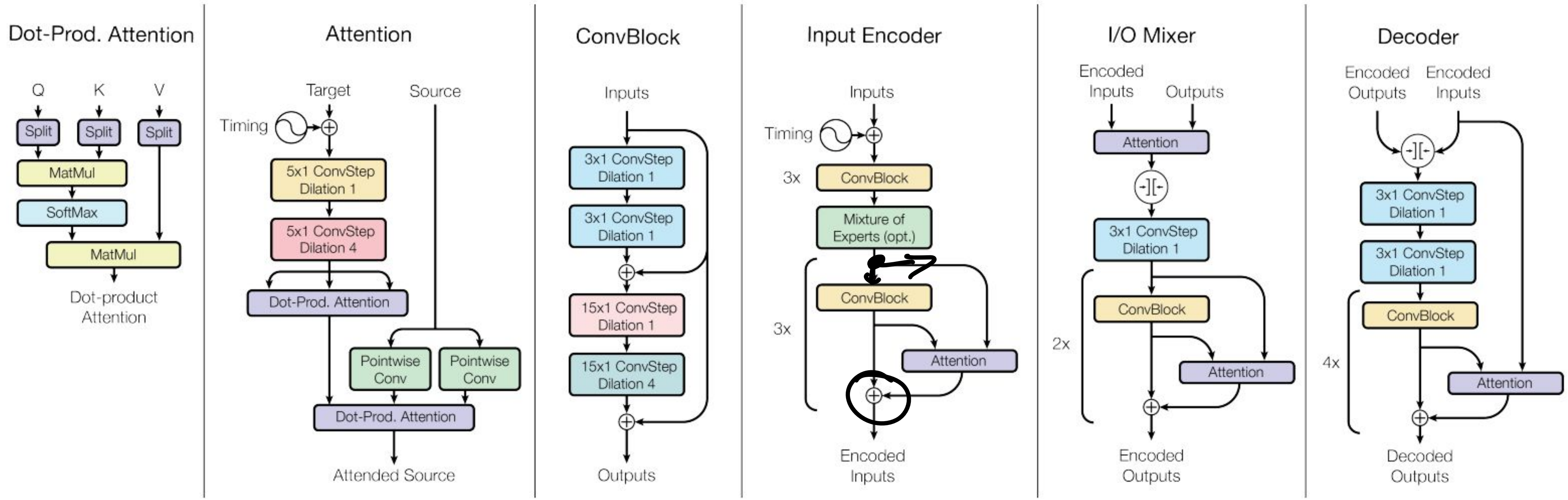
"Every sufficiently good **analogy** is yearning to become a *functor*."---John Baez



# Categories describe sequential processes

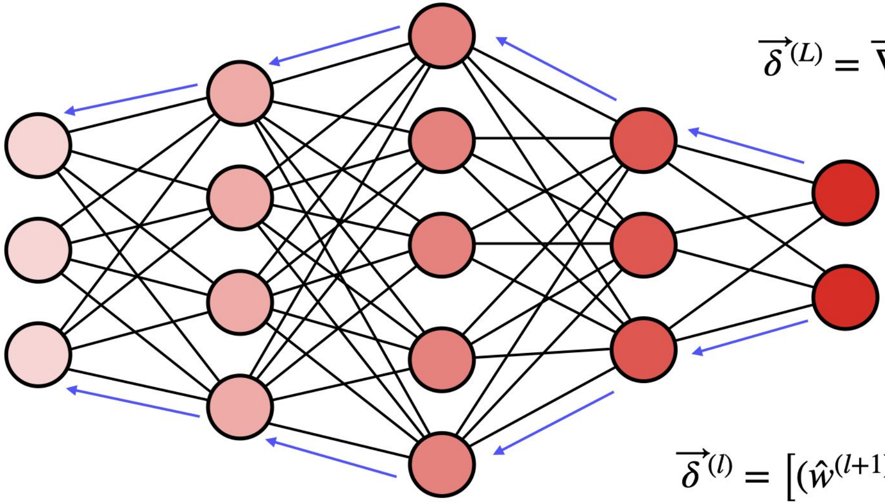


# What about N.N. architectures?




One Model To Learn Them All

# Backpropagation?



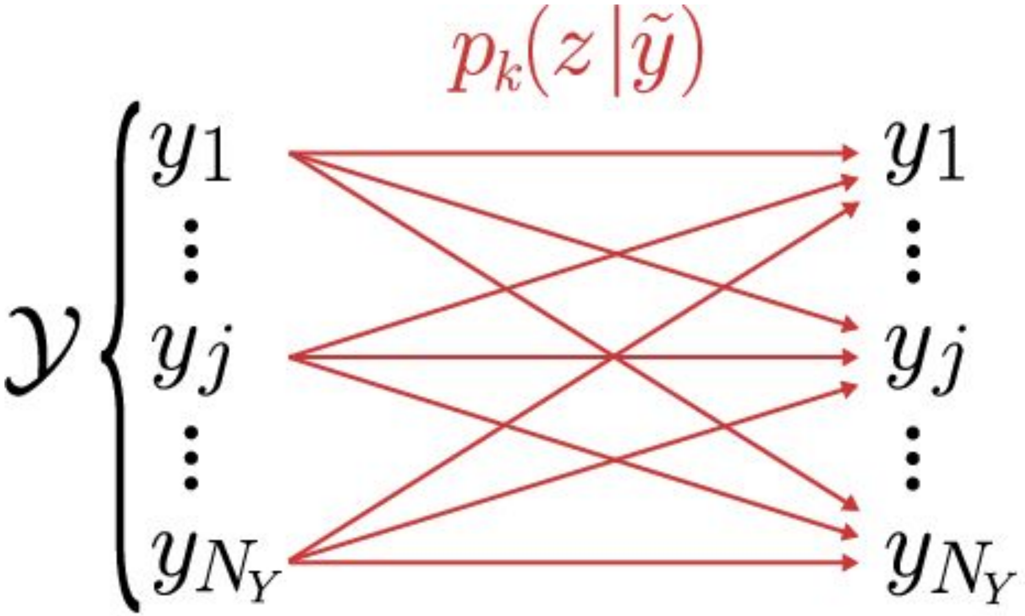
$$\vec{\delta}^{(L)} = \vec{\nabla}_{\vec{a}^{(L)}}\mathscr{L} \odot \sigma'(\vec{z}^{(L)})$$

$$\vec{\nabla}_{\vec{b}^{(l)}}\mathscr{L} = \vec{\delta}^{(l)}$$

$$\frac{\partial \mathscr{L}}{\partial w_{jk}^{(l)}} = a_k^{(l-1)}\delta_j^{(l)}$$

$$\vec{\delta}^{(l)} = \left[(\hat{w}^{(l+1)})^T \vec{\delta}^{(l+1)}\right] \odot \sigma'(\vec{z}^{(l)})$$

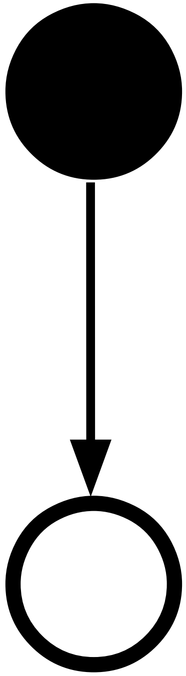
# Probability?




$$\tilde{Y}(k) \mapsto G(k, \tilde{Y}(k)) =: Z(k)$$

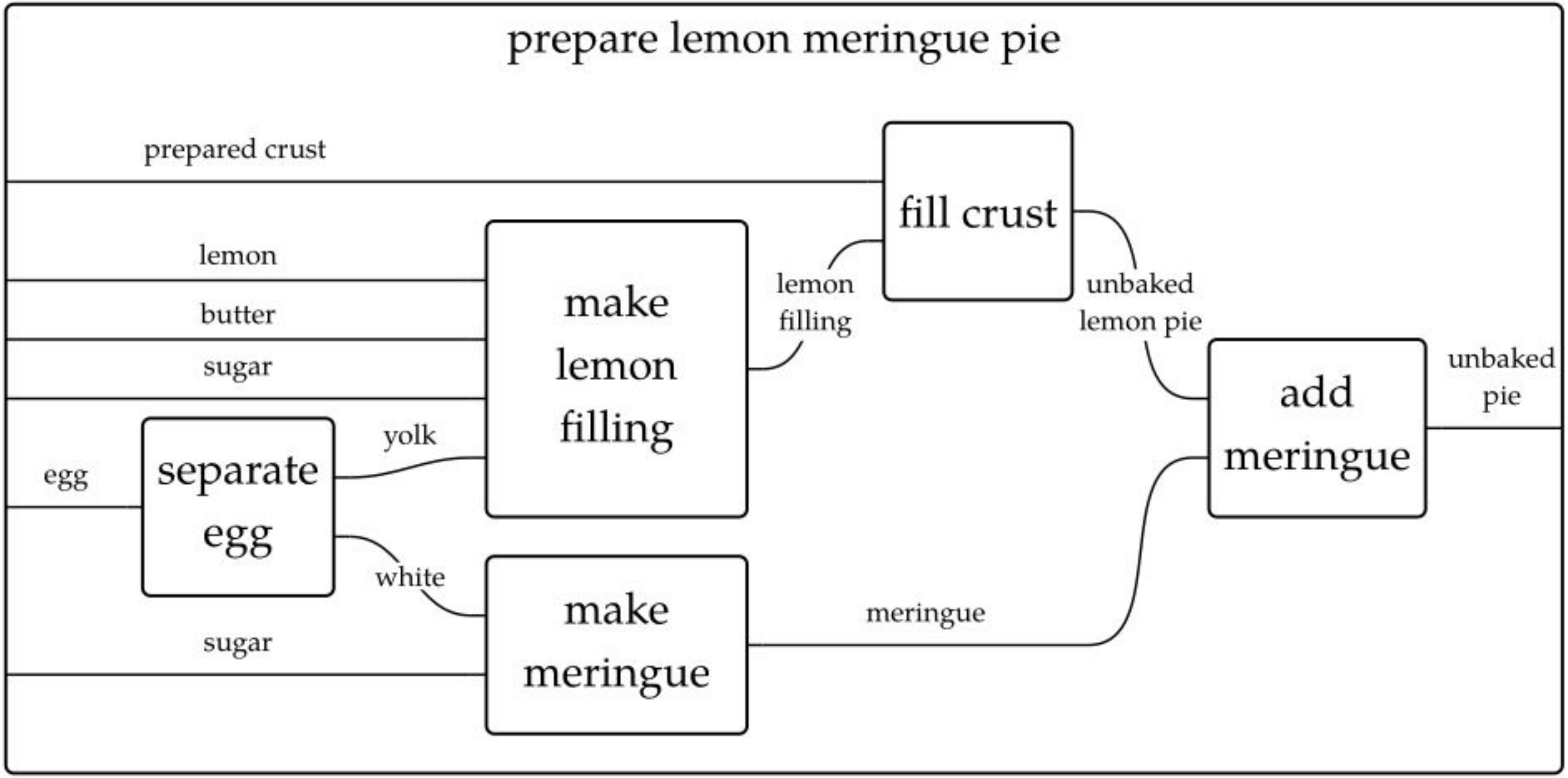
# Lecture plan

- How to model parallel processes? (Monoidal categories)
  - Graphical language of string diagrams
- Adding bells and whistles to monoidal categories
  - Ability to cross strings
  - Ability to copy/delete information
  - Ability to add/create information
- Deterministic parallel processes (Cartesian Monoidal categories)
- Deterministic bidirectional processes (Lenses)
- General bidirectional processes (Optics)



# Monoidal categories

# Mon. cats. describe parallel processes



# String diagrams

- Originating with Penrose's graphical notation for tensor networks

$$D^{\alpha\gamma}_{\rho\tau} A^{\beta}_{\gamma\sigma} - 3D^{\gamma\beta}_{\gamma\sigma} A^{\alpha}_{\tau\rho} = \delta^{\alpha}_{\sigma}\, x^{\beta}\, y_{\rho}\, y_{\tau} \rightsquigarrow$$

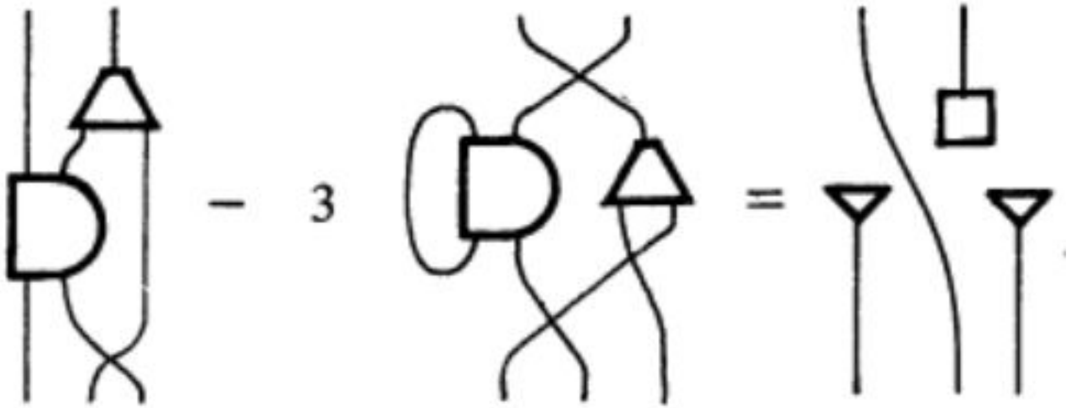


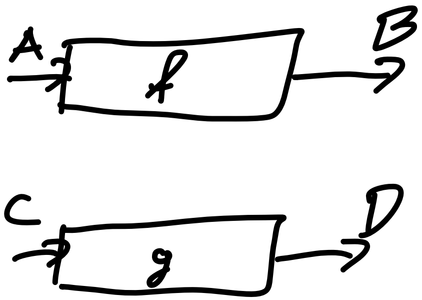

Fig. A-1. Diagrammatic representation of a tensor equation.

- Objects - strings
- Morphisms - boxes

# Moncats: What's the idea?

$$A \xrightarrow{f} B$$

$$C \xrightarrow{g} D$$




$$A \otimes C \xrightarrow{f \otimes g} B \otimes D$$

$$I : \mathcal{C}$$

$$\mathcal{C} \times \mathcal{C} \xrightarrow{\otimes} \mathcal{C}$$

$$(A, C) \mapsto A \otimes C$$

$$(B, D) \mapsto B \otimes D$$

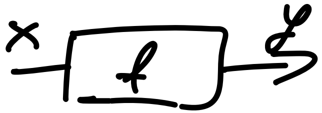
# Monoidal category: definition

**Definition 1.2.1.** A *monoidal category* is a tuple

$$(\mathsf{C}, \otimes, \mathbb{1}, \alpha, \lambda, \rho)$$

consisting of:

- a category $\mathsf{C}$;
- a functor $\otimes : \mathsf{C} \times \mathsf{C} \longrightarrow \mathsf{C}$ called the *monoidal product*;
- an object $\mathbb{1} \in \mathsf{C}$ called the *monoidal unit*;




$X \otimes I \cong X$

$I \otimes X \cong X$

$(X \otimes Y) \otimes Z \cong X \otimes (Y \otimes Z)$

# Monoidal category: definition

**Definition 1.2.1.** A *monoidal category* is a tuple

$$(\mathsf{C}, \otimes, \mathbb{1}, \alpha, \lambda, \rho)$$

consisting of:

- a category $\mathsf{C}$;
- a functor $\otimes : \mathsf{C} \times \mathsf{C} \longrightarrow \mathsf{C}$ called the *monoidal product*;
- an object $\mathbb{1} \in \mathsf{C}$ called the *monoidal unit*;
- a natural isomorphism

(1.2.2) $$(X \otimes Y) \otimes Z \xrightarrow[\cong]{\alpha_{X,Y,Z}} X \otimes (Y \otimes Z)$$

  for all objects $X, Y, Z \in \mathsf{C}$ called the *associativity isomorphism*;
- natural isomorphisms

(1.2.3) $$\mathbb{1} \otimes X \xrightarrow[\cong]{\lambda_X} X \quad \text{and} \quad X \otimes \mathbb{1} \xrightarrow[\cong]{\rho_X} X$$

  for all objects $X \in \mathsf{C}$ called the *left unit isomorphism* and the *right unit isomorphism*, respectively.

$X \cdot 1 = X$

Johnson, Yau, 2-Dimensional Categories

# …such that these axioms are satisfied

**Unity Axioms:** The *middle unity diagram*

(1.2.4)
$$\begin{array}{ccc} (X \otimes \mathbb{1}) \otimes Y & \xrightarrow{\alpha_{X,\mathbb{1},Y}} & X \otimes (\mathbb{1} \otimes Y) \\ {\scriptstyle \rho_X \otimes Y} \downarrow & & \downarrow {\scriptstyle X \otimes \lambda_Y} \\ X \otimes Y & = & X \otimes Y \end{array}$$

is commutative for all objects $X, Y \in \mathsf{C}$. Moreover, the equality

$$\lambda_{\mathbb{1}} = \rho_{\mathbb{1}} : \mathbb{1} \otimes \mathbb{1} \xrightarrow{\cong} \mathbb{1}$$

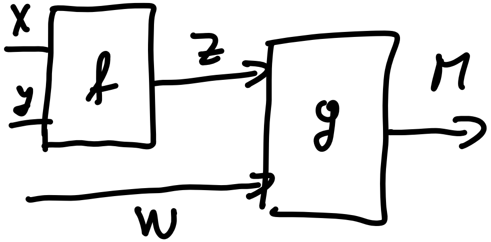
holds.

**Pentagon Axiom:** The pentagon

(1.2.5)
$$\begin{array}{ccc} & (W \otimes X) \otimes (Y \otimes Z) & \\ {\scriptstyle \alpha_{W \otimes X, Y, Z}} \nearrow & & \searrow {\scriptstyle \alpha_{W,X,Y \otimes Z}} \\ ((W \otimes X) \otimes Y) \otimes Z & & W \otimes (X \otimes (Y \otimes Z)) \\ {\scriptstyle \alpha_{W,X,Y} \otimes Z} \downarrow & & \uparrow {\scriptstyle W \otimes \alpha_{X,Y,Z}} \\ (W \otimes (X \otimes Y)) \otimes Z & \xrightarrow{\alpha_{W,X \otimes Y,Z}} & W \otimes ((X \otimes Y) \otimes Z) \end{array}$$

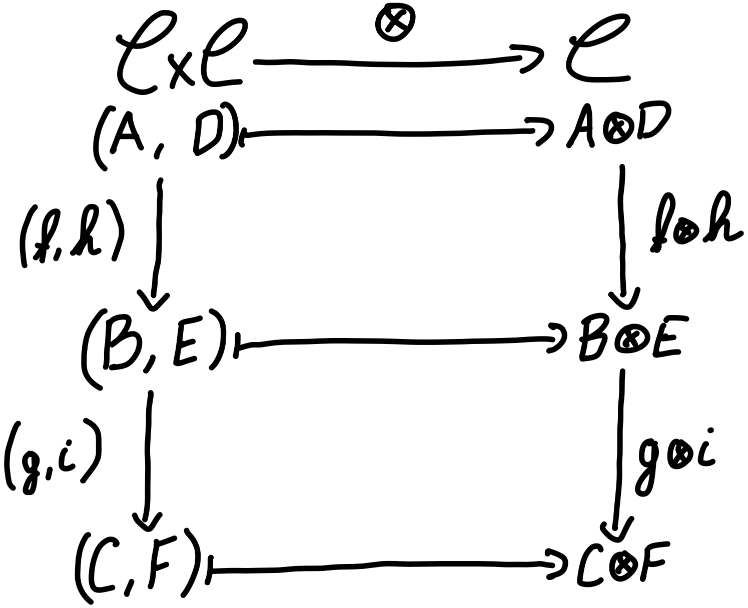
Johnson, Yau, 2-Dimensional Categories

# Laws of a monoidal category capture the geometry of the plane




$$(X\otimes Y)\otimes W \xrightarrow{f\otimes 1_W} Z\otimes W \xrightarrow{g} M$$

# Q: Why does ⊗ need to be a functor?




FUNCTORIALITY OF ⊗:

$$(f\otimes h);(g\otimes i) = (f;g)\otimes(h;i)$$

NOTATION

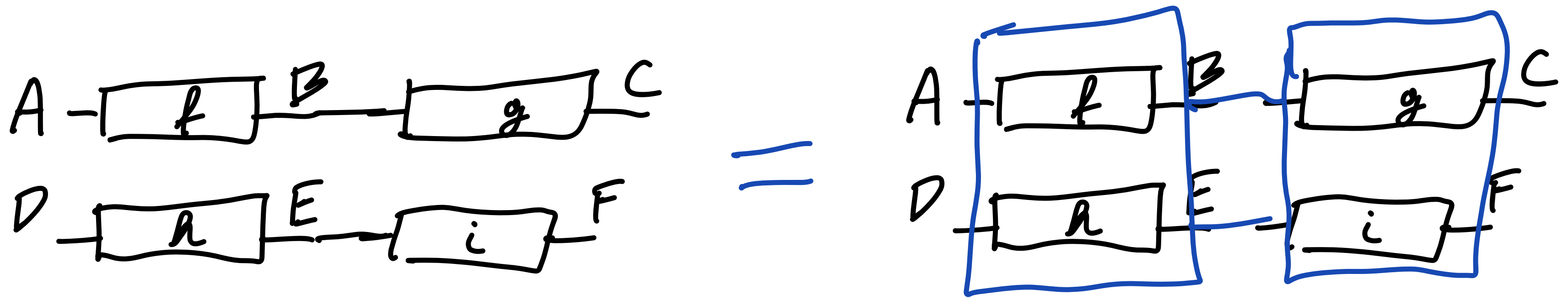
$$A \xrightarrow{f} B \xrightarrow{g} C$$

$$g\circ f = f;g$$

WE SOMETIMES USE 'DIAGRAMMATIC' NOTATION, INVOLVING A SEMICOLON

# Q: Why does ⊗ need to be a functor?

$$(f \otimes h) ; (g \otimes i) = (f ; g) \otimes (h ; i)$$



# String diagrams: sound and complete

A well-typed equation between morphisms in a monoidal category follows from the axioms *if and only if* it holds in the graphical language up to planar isotopy.

# Q: Which of these diagrams are equal...

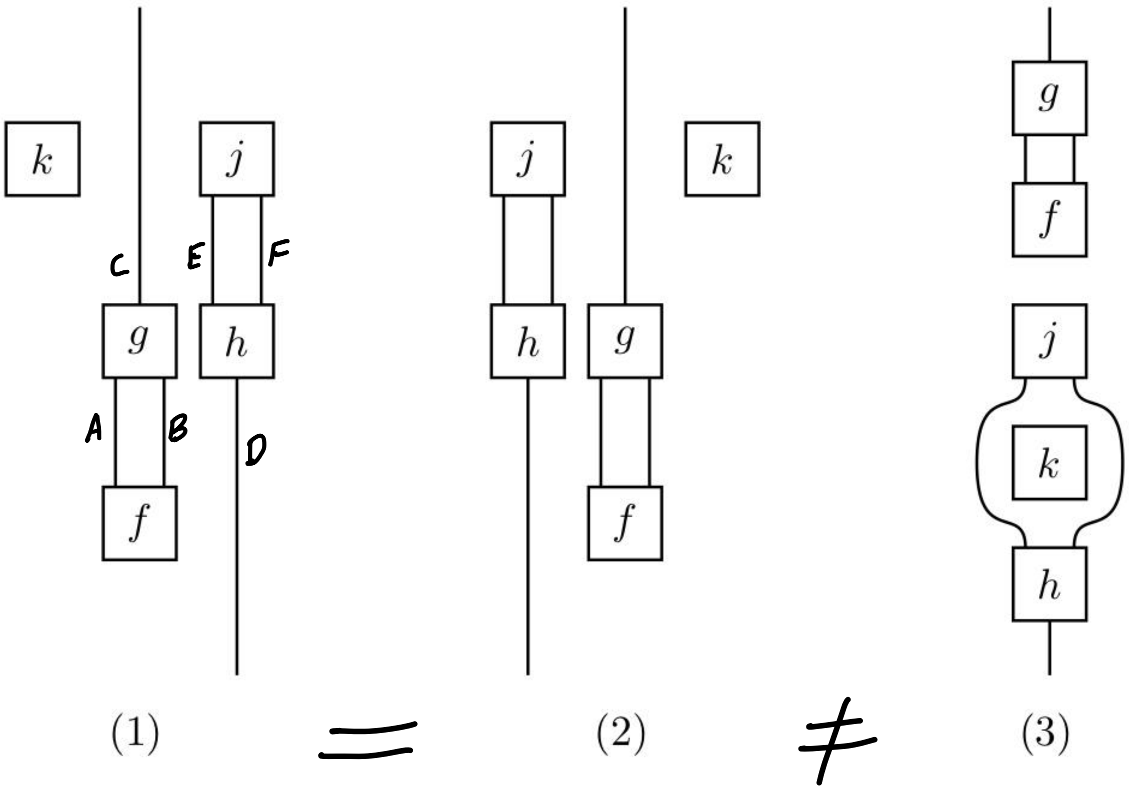
$k: I \longrightarrow I$

$f: I \longrightarrow A \otimes B$

$g: A \otimes B \longrightarrow C$

$h: D \longrightarrow E \otimes F$

$j: E \otimes F \longrightarrow I$




...in a monoidal category?

Heunen, Lecture Notes,
Categories and Quantum Informatics

ABOVE DIAGRAMS IN EQUATION FORM:

1) $D \cong I \otimes (I \otimes D) \xrightarrow{k \otimes f \otimes h} I \otimes ((A \otimes B) \otimes (E \otimes F)) \xrightarrow{I \otimes g \otimes j} I \otimes (C \otimes I) \cong C$

2) $D \cong (D \otimes I) \otimes I \xrightarrow{h \otimes f \otimes k} ((E \otimes F) \otimes (A \otimes B)) \otimes I \xrightarrow{j \otimes g \otimes I} (I \otimes C) \otimes I \cong C$

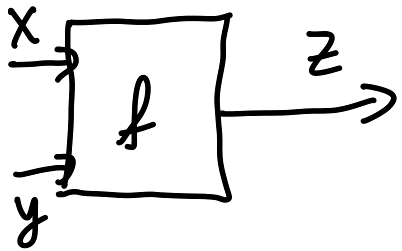
3) $D \xrightarrow{h} E \otimes F \cong E \otimes (I \otimes F) \xrightarrow{E \otimes (k \otimes F)} E \otimes (I \otimes F) \cong E \otimes F \xrightarrow{j} I \xrightarrow{f} A \otimes B \xrightarrow{g} C$

USING THE STRUCTURE OF A MONDIDAL CATEGORY $(\lambda, \rho, \alpha)$ WE CAN TRANSFORM 1) INTO 2) AND BACK.

BUT THERE IS NO WAY TO TRANSFORM EITHER ONE INTO 3)!

# Example: (Set, ×, 1)

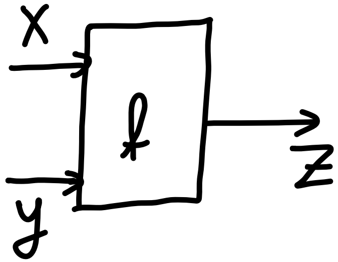
Set




$X \times 1 \cong X$

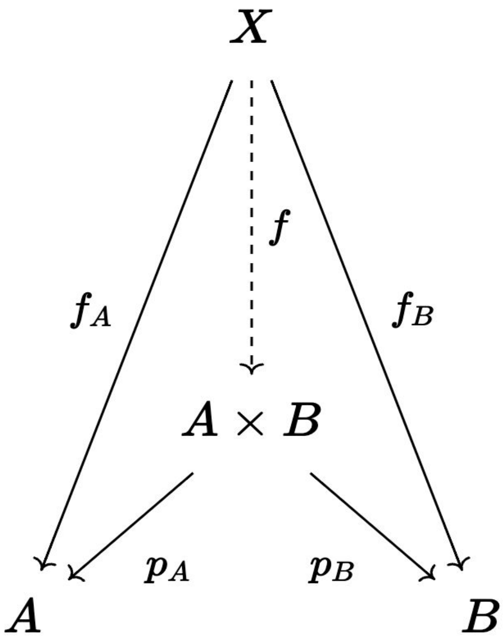
$f : X \times Y \longrightarrow Z$

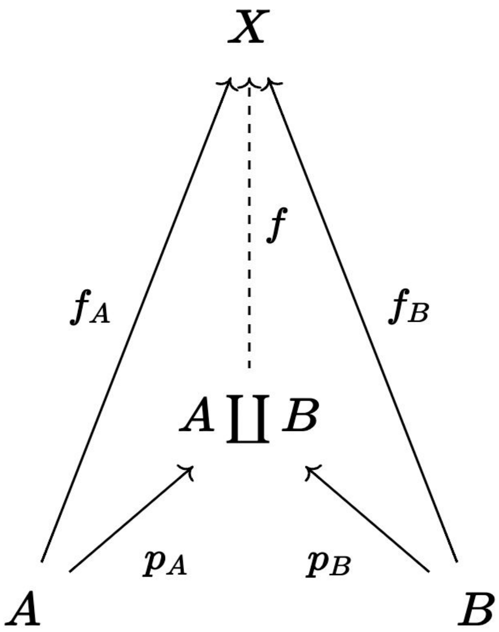
# Example: (Set, ⊔, ∅)




$$f : X \sqcup Y \longrightarrow Z$$

- Monoidal categories generalise products and coproducts!



# A category can have many monoidal products

# Example: (Vect, $\otimes$, $\mathbb{R}$), (Vect, $\oplus$, 1)

$\mathbb{R}$

# Example: (Rel, ⊗, 1)

$$A \xrightarrow{R} B \qquad\qquad A \times C \xrightarrow{R \otimes P} B \times D$$

$$C \xrightarrow{P} D \qquad\qquad (a,c)\, R \otimes P\, (b,d) = \begin{matrix} aRb \text{ AND} \\ cPd \end{matrix}$$

# Example: any monoid

- A monoid is a monoidal category with only identity morphisms

$A : \mathrm{Set}$

$\cdot : A \times A \longrightarrow A$

$1 : A$

- IS ASSOCIATIVE $\quad \forall a, b, c \quad (a \cdot b) \cdot c = a \cdot (b \cdot c)$
- IS UNITAL $\quad \forall a \quad a \cdot 1 = a$

$1 \cdot a = a$

# A monoid is a discrete monoidal category

- $(\mathbb{R}, +, 0)$
- $(\mathbb{R}, *, 1)$
- (List X, concat, []), where X is any set
- $(\mathbb{B}$, AND, True)
- $([X, X], \circ, id_X)$, where X is any set
- …

List $\mathbb{N}$

[1,2,7]++[4,5]=[1,2,7,4,5]

# Example: (Euc, ×, 1)

$$\mathbb{R}^N \xrightarrow{f} \mathbb{R}^M \qquad \mathbb{R}^N \times \mathbb{R}^K \xrightarrow{f \times g} \mathbb{R}^M \times \mathbb{R}^L$$

$$\mathbb{R}^K \xrightarrow{g} \mathbb{R}^L$$

# Our categories need not be deterministic!

- Category FinStoch
- Objects are finite sets
- Morphisms are Markov kernels

$$X \xrightarrow{f} Y \xrightarrow{g} Z \qquad f : \mathbb{R}^{X \times Y}$$

$$f(y|x)$$

FOR EACH $x : X$

$$\sum_{y:Y} f(y|x) = 1$$

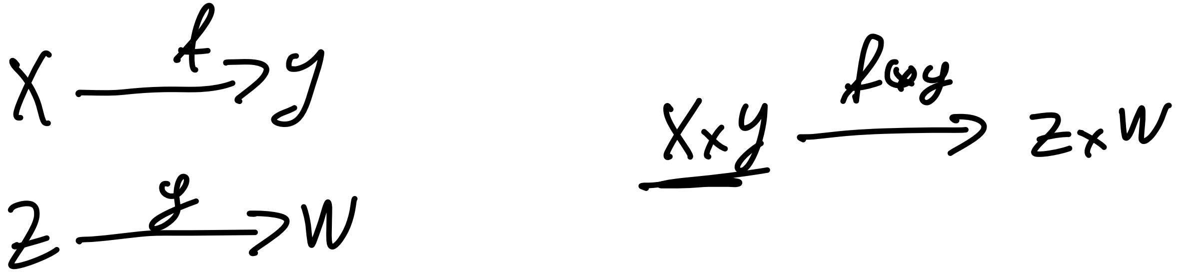
$$f(y|x) \geq 0$$

$$2 \overset{3}{\begin{bmatrix} 0.1 & 0.6 & 0.3 \\ 0.4 & 0.4 & 0.2 \end{bmatrix}}$$

Markov Categories

# (FinStoch, ⊗, 1)

- Implements stochastic independence
- At the level of objects given by cartesian product
- At the level of morphisms given by the Kronecker product of matrices

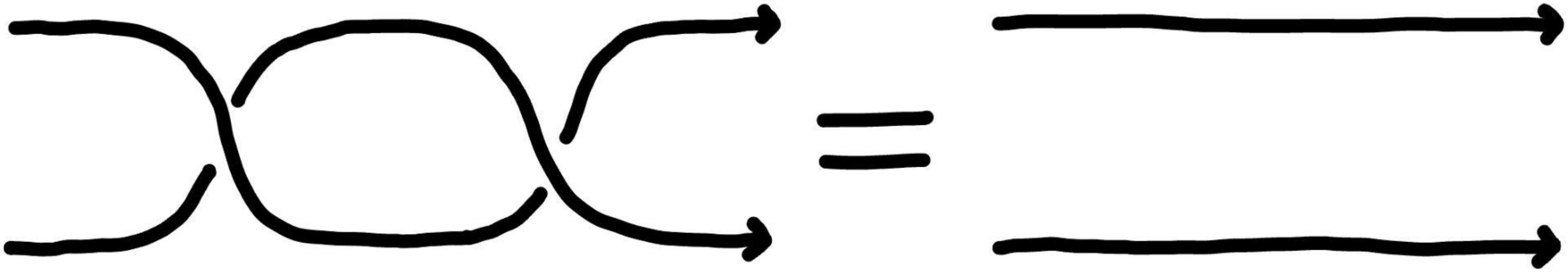


# All of these categories have a lot of structure!

**…but an *arbitrary* monoidal category doesn't necessarily possess that structure.**

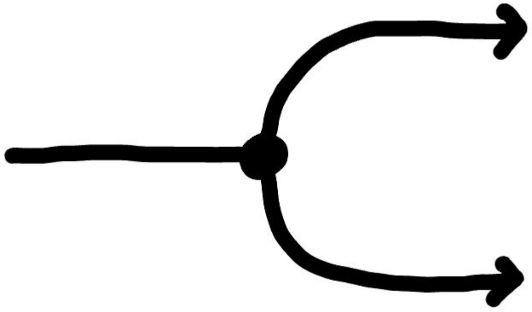
# What can’t we do in an *arbitrary* monoidal category?

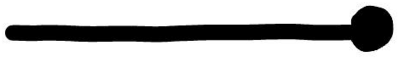
# Pass strings through each other



(Symmetric monoidal category)

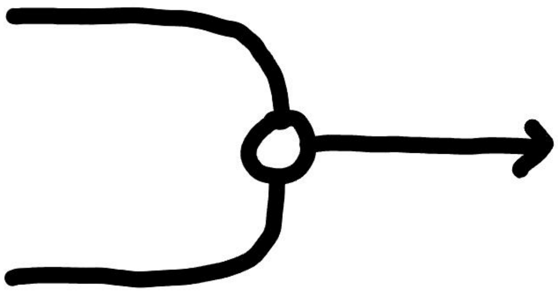
# Split/end strings





**(Symmetric monoidal category with a *supply* of comonoids)**

# Add/start strings



**(Symmetric monoidal category with a *supply* of monoids)**

# Idea - interfaces

```python
class Animal(ABC):
    @abstractmethod
    def say(self):
        pass

    @abstractmethod
    def do(self):
        pass

class Dog(Animal):
    def say(self):
        print("Woof!")

    def do(self):
        run_in_park()

class Cat(Animal):
    def say(self):
        print("Meow.")

    def do(self):
        sleep()
```

- Interface based design
- As in computer science, an abstract interface tells us what operations are available for a particular object

- In CT we take this idea rigorously!

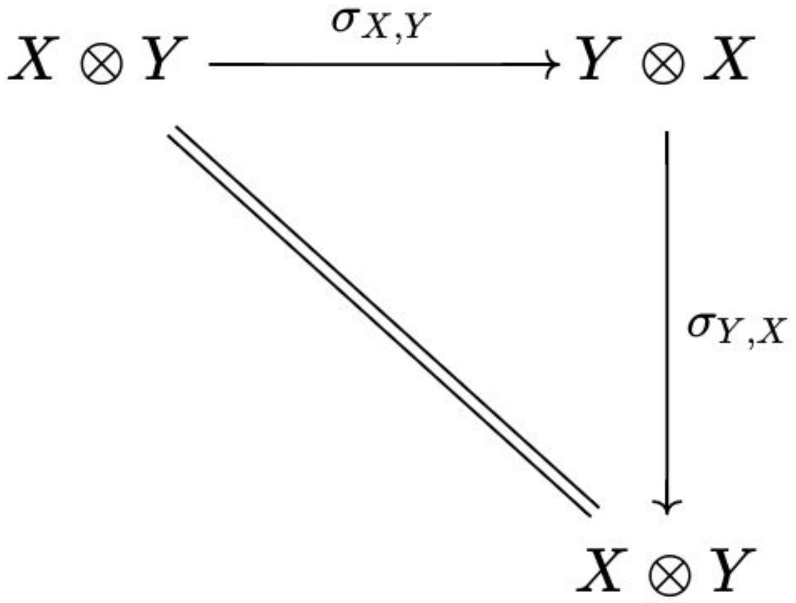
# Symmetric Monoidal Category

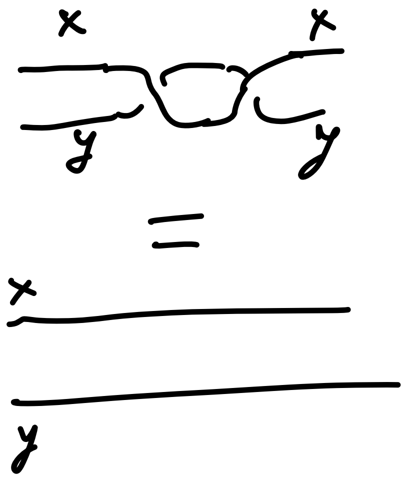
# Symmetric Monoidal Category: Definition

- A monoidal category $\mathcal{C}$ equipped with

$$\sigma_{X,Y} : X \otimes Y \longrightarrow Y \otimes X$$

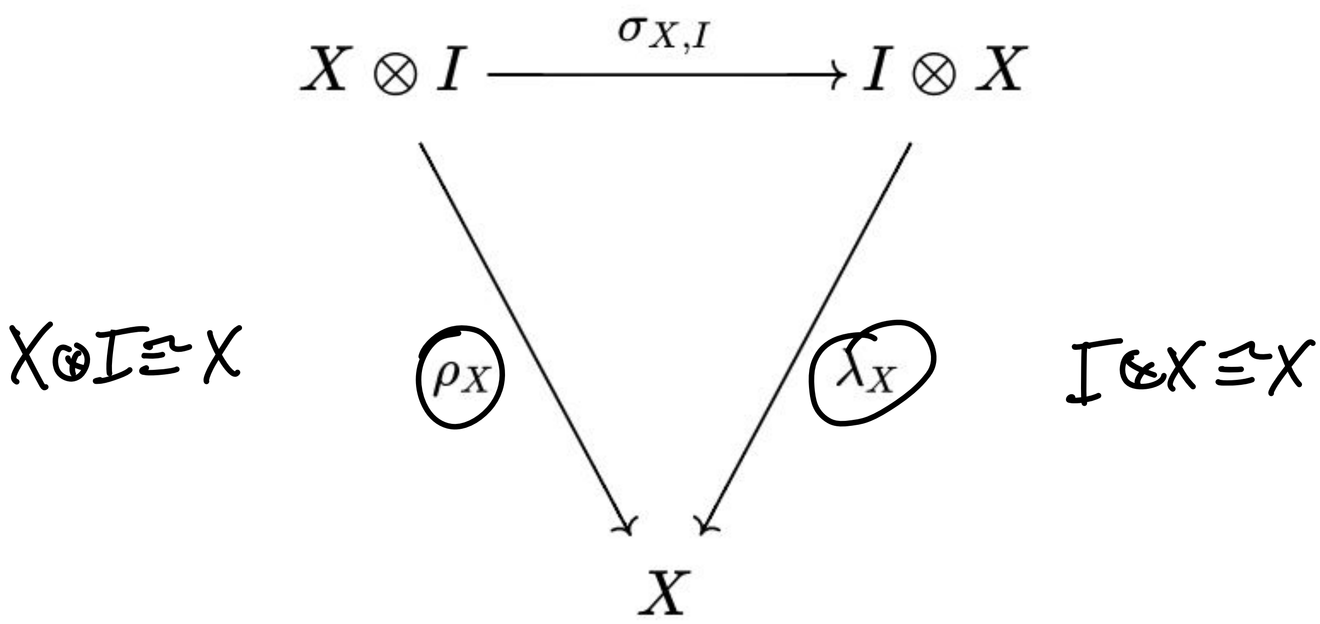
# …such that these axioms hold:



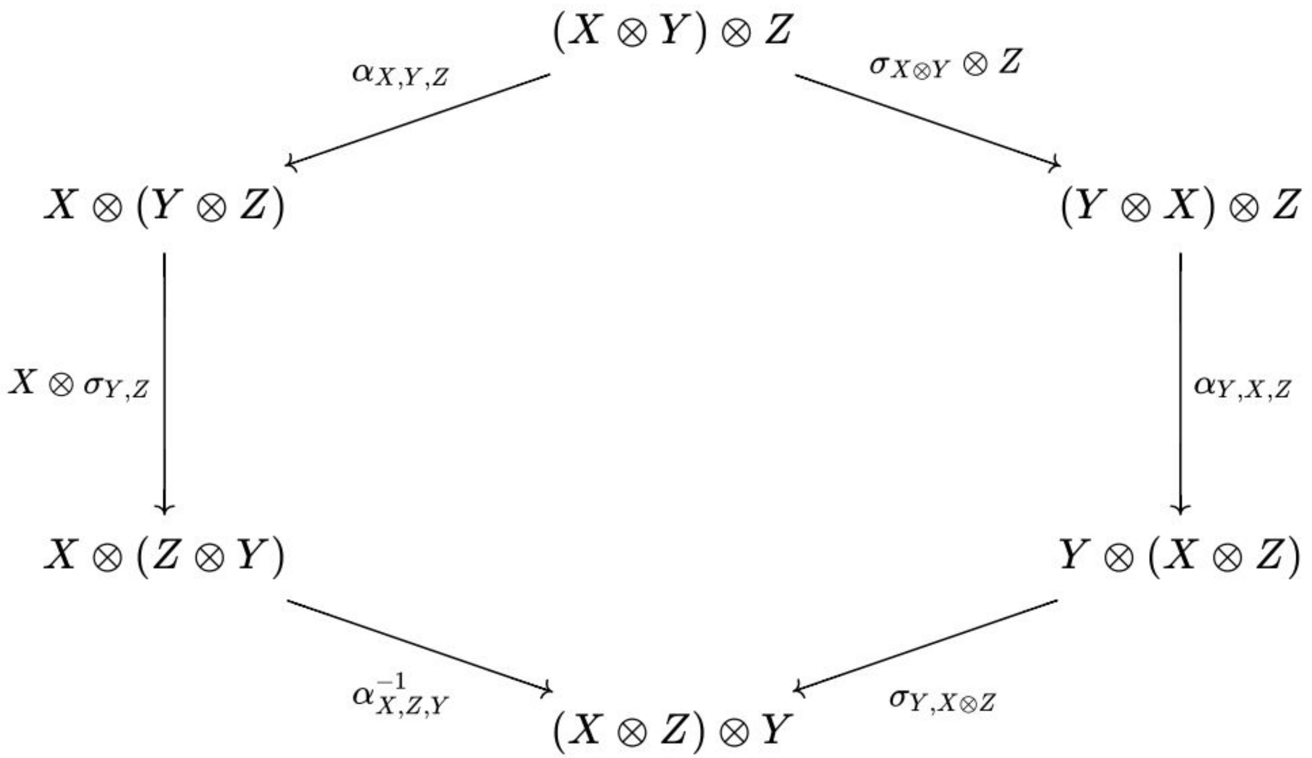

- For all X,Y:C

# …such that these axioms hold:




- For all X:C

# …such that these axioms hold:




- For all X, Y, Z:C

# Example of a non-symmetric mon. cat.

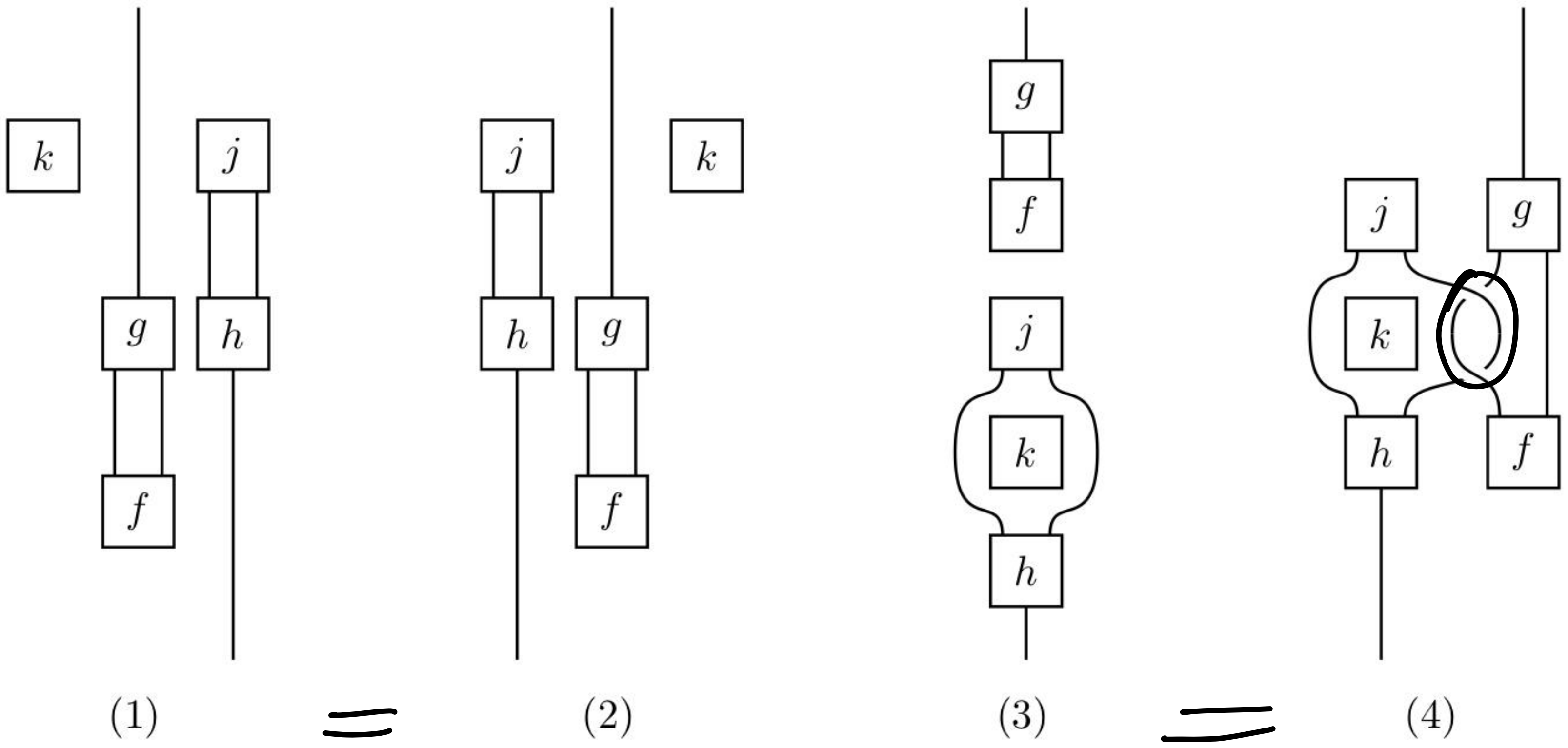
- $([X, X], \circ, id_X)$, where X is any set

$[X, X]$ $\quad X \xrightarrow{f} X$

$X \xrightarrow{g} X$

$X \xrightarrow{f \circ g} X$

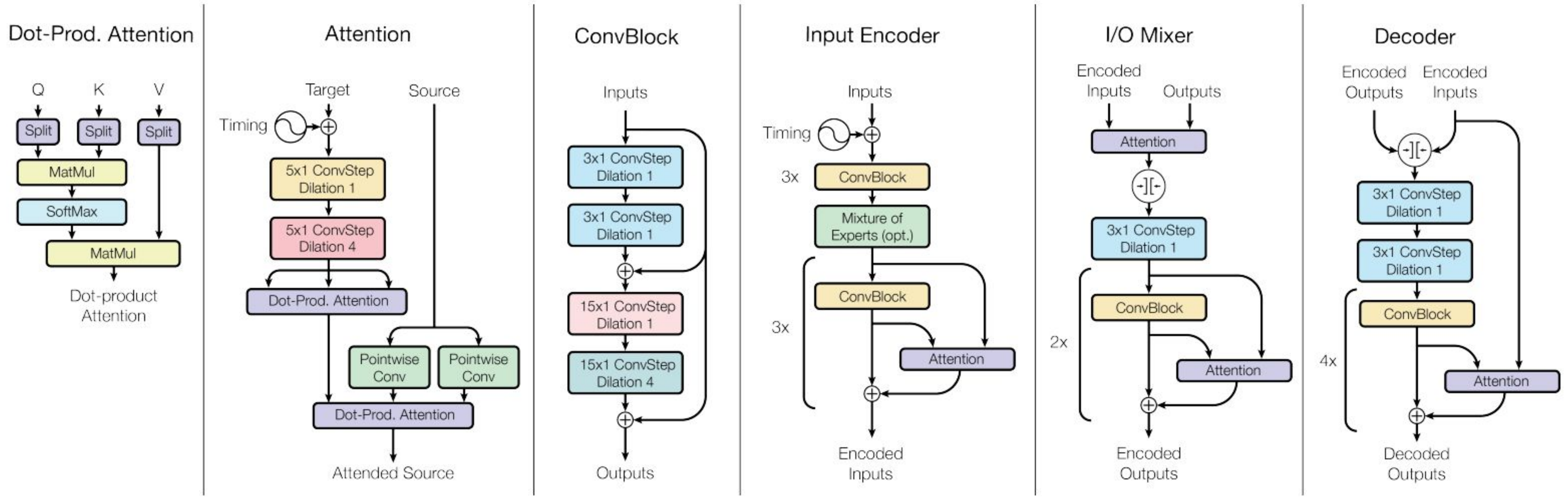
# Q: Which of these diagrams are equal…

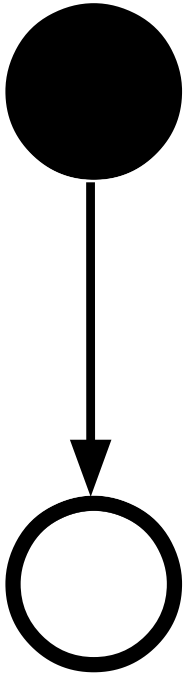



...in a symmetric monoidal category?

Heunen, Lecture Notes,
Categories and Quantum Informatics

# Recall:


Dot-Prod. Attention
Q
K
V
Split
Split
Split
MatMul
SoftMax
MatMul
Dot-product Attention
Attention
Target
Source
Timing
5x1 ConvStep Dilation 1
5x1 ConvStep Dilation 4
Dot-Prod. Attention
Pointwise Conv
Pointwise Conv
Dot-Prod. Attention
Attended Source
ConvBlock
Inputs
3x1 ConvStep Dilation 1
3x1 ConvStep Dilation 1
15x1 ConvStep Dilation 1
15x1 ConvStep Dilation 4
Outputs
Input Encoder
Inputs
Timing
3x
ConvBlock
Mixture of Experts (opt.)
3x
ConvBlock
Attention
Encoded Inputs
I/O Mixer
Encoded Inputs
Outputs
Attention
3x1 ConvStep Dilation 1
2x
ConvBlock
Attention
Encoded Outputs
Decoder
Encoded Outputs
Encoded Inputs
3x1 ConvStep Dilation 1
3x1 ConvStep Dilation 1
4x
ConvBlock
Attention
Decoded Outputs



# SMCs with *supplies*

# SMC with supplies

- Symmetric monoidal category with a specific structure on each object

Two examples:

- Symmetric monoidal category with *a (homomorphic) supply of comonoids*
- Symmetric monoidal category with *a (homomorphic) supply of monoids*

# What is a comonoid? $\mathbb{N}(\mathrm{Set}, \times, 1)$

$A : \mathrm{Set}$

$\bullet : A \times A \longrightarrow A$

$N : 1 \longrightarrow A$

$B : \mathrm{Set}$

COPY $\Delta_B : B \longrightarrow B \times B$

$b \longmapsto (b, b)$

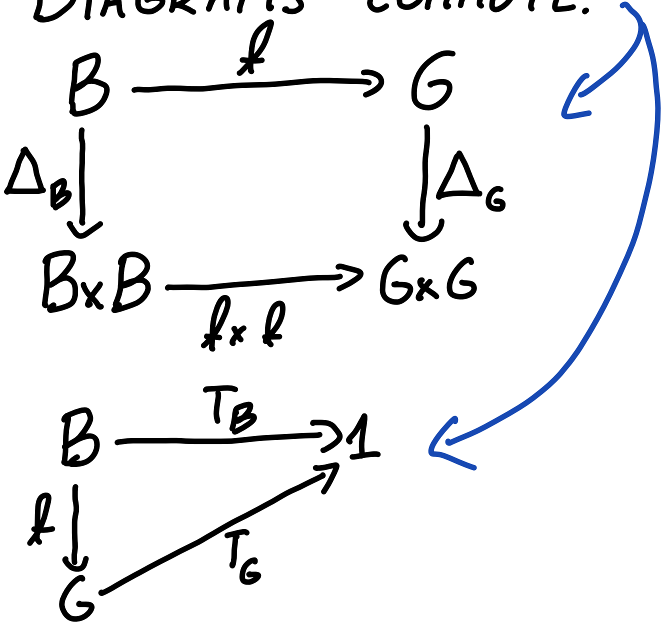
DELETE $T_B : B \longrightarrow 1$

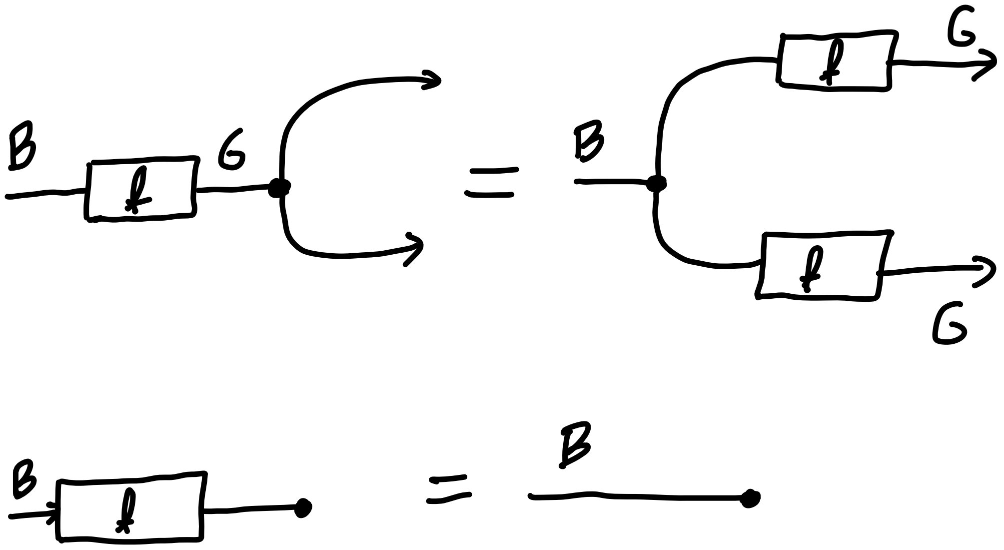
$(B, \Delta_B, T_B) \longrightarrow (G, \Delta_G, T_G)$

# What is a comonoid homomorphism?

$$(B, \Delta_B, T_B) \longrightarrow (G, \Delta_G, T_G)$$

IT IS A FUNCTION $f: B \longrightarrow G$ SUCH THAT THE FOLLOWING DIAGRAMS COMMUTE:

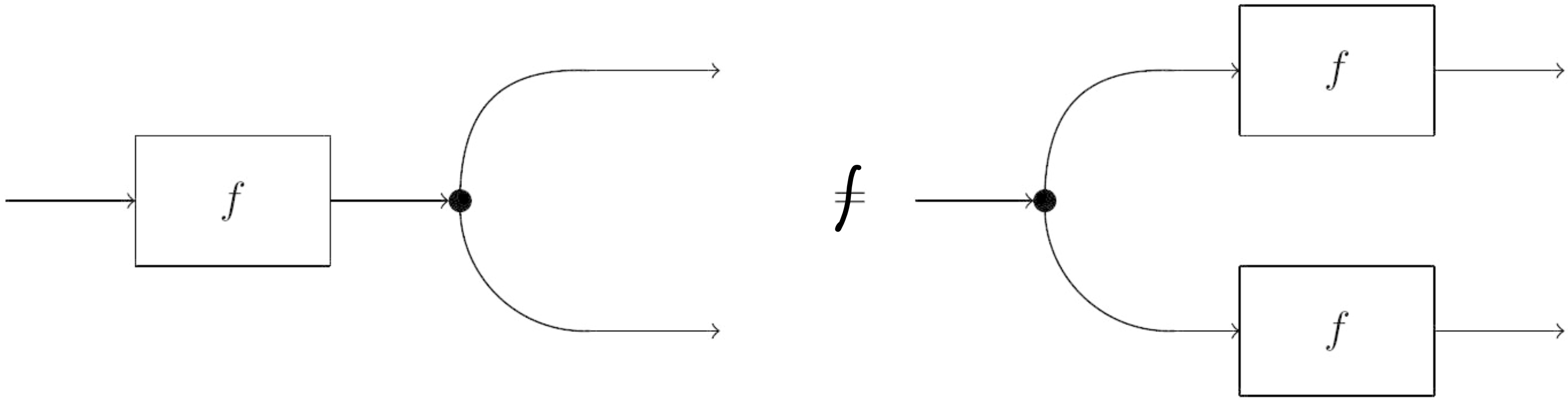


# Comonoid homomorphisms
# =
# Deterministic maps

# Non-example: FinStoch!

- We cannot slide copy through arbitrary maps!




- Rolling a dice and copying the result is not the same as rolling two dice

# A homomorphic supply of comonoids...

- $(\mathrm{Set}, \times, 1)$
- $(\mathrm{Vect}, \otimes, \mathbb{R})$
- $(\mathrm{Vect}, \oplus, 1)$

$(\mathrm{Euc}, \times, 1)$

# …gives a category with products!

Symmetric monoidal category with a homomorphic supply of comonoids

is isomorphic to

a monoidal category whose monoidal product is given by the category-theoretic product

# Operational view

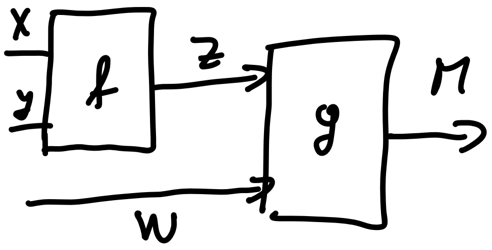
- Gives us an operational characterization of a category with products
- Ability to systematically copy and delete information
- Every map in a cartesian monoidal category is deterministic

# Exercise:

- What does a homomorphic supply of monoids give us?

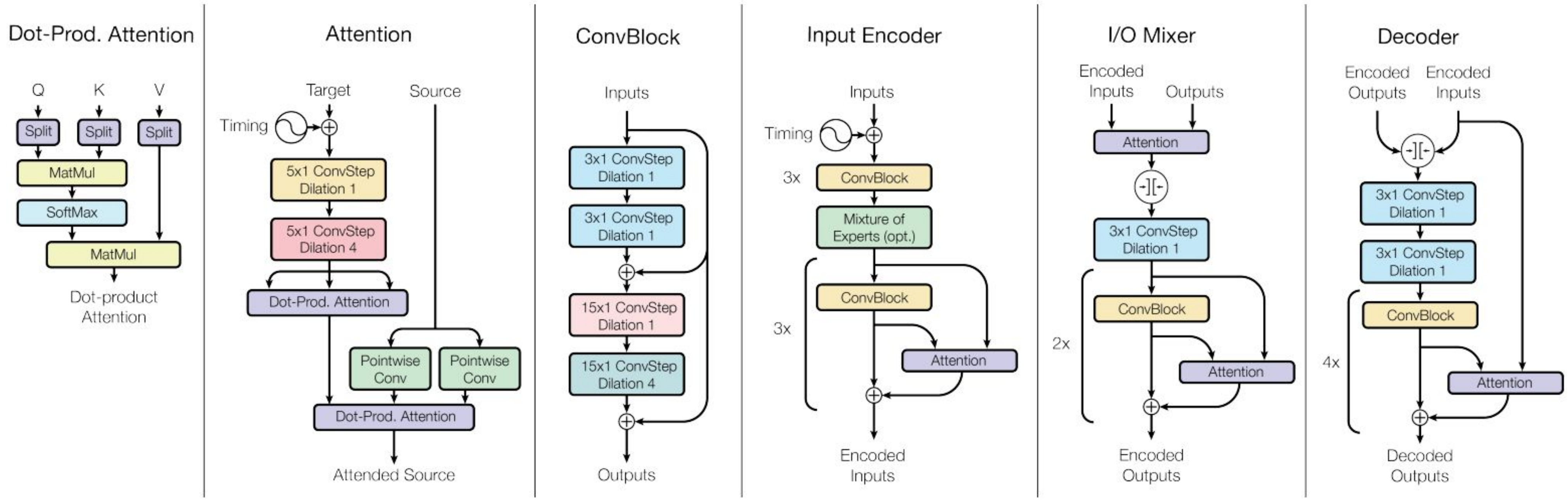
# So far:

- Parallel processes (Monoidal categories)
- Crossing of strings (Symmetric monoidal categories)
- Copying/deleting information (Cartesian categories)




$$(X \otimes Y) \otimes W \xrightarrow{f \otimes 1_W} Z \otimes W \xrightarrow{g} M$$

# This gives us the basic building blocks!




One Model To Learn Them All

# Reverse Derivative Ascent:
# A Categorical Approach to Learning Boolean Circuits

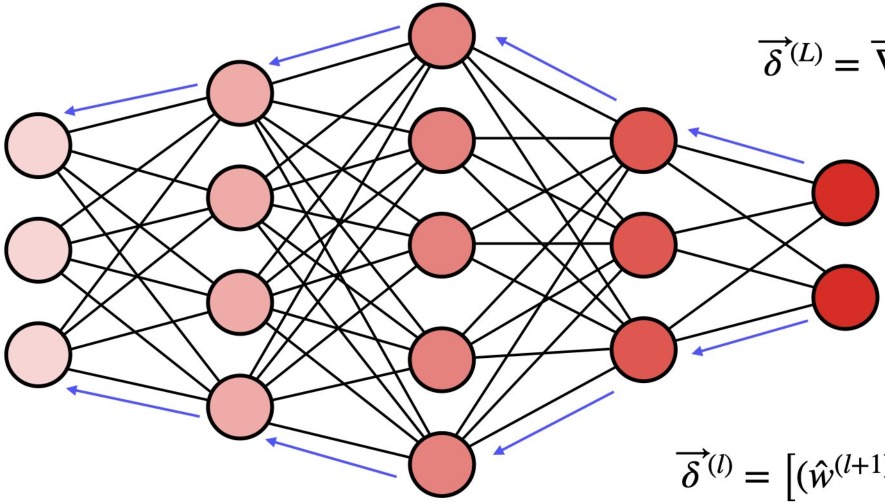
Paul Wilson
University College London
University of Southampton
paul@statusfailed.com

Fabio Zanasi
University College London
f.zanasi@ucl.ac.uk

We introduce *Reverse Derivative Ascent*: a categorical analogue of gradient based methods for machine learning. Our algorithm is defined at the level of so-called *reverse differential categories*. It can be used to learn the parameters of models which are expressed as morphisms of such categories. Our motivating example is boolean circuits: we show how our algorithm can be applied to such circuits by using the theory of reverse differential categories. Note our methodology allows us to learn the parameters of boolean circuits *directly*, in contrast to existing binarised neural network approaches. Moreover, we demonstrate its empirical value by giving experimental results on benchmark machine learning datasets.

**What about a category where morphisms have a *forward* and a *backward* component?**

# Recall

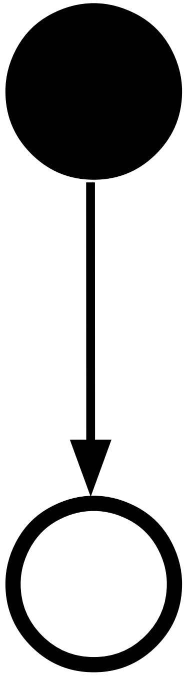


$$\vec{\delta}^{(L)} = \vec{\nabla}_{\vec{a}^{(L)}}\mathscr{L} \odot \sigma'(\vec{z}^{(L)})$$

$$\vec{\nabla}_{\vec{b}^{(l)}}\mathscr{L} = \vec{\delta}^{(l)}$$

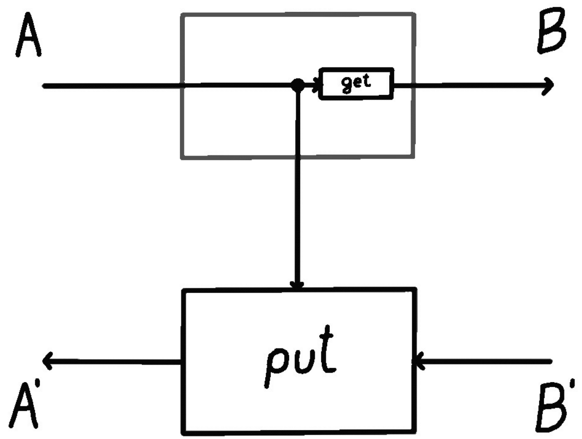
$$\frac{\partial \mathscr{L}}{\partial w_{jk}^{(l)}} = a_k^{(l-1)}\delta_j^{(l)}$$

$$\vec{\delta}^{(l)} = \left[(\hat{w}^{(l+1)})^T \vec{\delta}^{(l+1)}\right] \odot \sigma'(\vec{z}^{(l)})$$



# Lenses

# What is a lens?

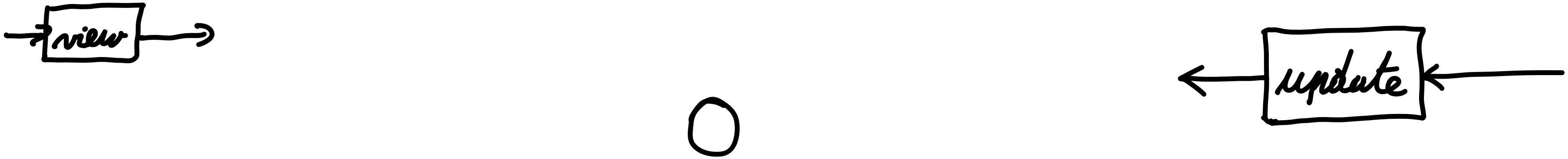



- Lenses mo
- Give us a high-level view of the bidirectional computation pattern

# A lens consists of two parts

Forward map

Backward map



# Lenses form a category

- Starting with any cartesian category C…

  … we can form a category Lens(C) defined as follows…

# Category of Lenses: Definition

DEF. THE CATEGORY $\mathrm{Lens}(\mathcal{C})$ $\qquad A, A' : \mathcal{C}$

- OBJECTS - PAIRS OF OBJECTS IN $\mathcal{C}$ $\begin{pmatrix} A \\ A' \end{pmatrix}$
- MORPHISM $\begin{pmatrix} A \\ A' \end{pmatrix} \longrightarrow \begin{pmatrix} B \\ B' \end{pmatrix}$ IS A PAIR (view, upd) WHERE:

$\mathrm{view} : A \longrightarrow B$

AND $\qquad$ ARE MORPHISMS IN $\mathcal{C}$.

$\mathrm{upd} : A \times B' \longrightarrow A'$

# Lens composition

THE COMPOSITE OF

$$\begin{pmatrix} A \\ A' \end{pmatrix} \xrightarrow{(view_1,\, upd_1)} \begin{pmatrix} B \\ B' \end{pmatrix} \quad \text{AND} \quad \begin{pmatrix} B \\ B' \end{pmatrix} \xrightarrow{(view_2,\, upd_2)} \begin{pmatrix} C \\ C' \end{pmatrix} \quad \text{WHERE}$$

$view_1 : A \longrightarrow B$

$upd_1 : A \times B' \longrightarrow A'$

$view_2 : B \longrightarrow C$

$upd_2 : B \times C' \longrightarrow B'$

$\Delta$

$id_A : A \to A$

IS

$$\begin{pmatrix} A \\ A' \end{pmatrix} \xrightarrow{(v,\, u)} \begin{pmatrix} C \\ C' \end{pmatrix}$$

$$v := A \xrightarrow{v_1} B \xrightarrow{v_2} C$$

$$A \times C' \xrightarrow{\Delta_A \times C'} A \times A \times C' \xrightarrow{A \times view_1 \times C'} A \times B \times C' \xrightarrow{A \times upd_2} A \times B' \xrightarrow{upd_1} A'$$

# Lens composition

$$A \xrightarrow{v_1} B \xrightarrow{v_2} C$$

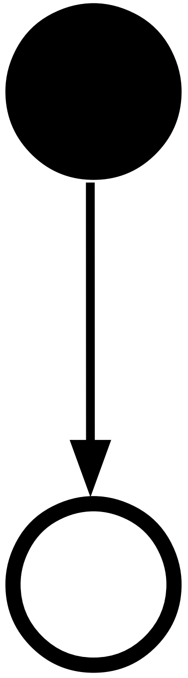
$$A \times C' \xrightarrow{\Delta_A \times C'} A \times A \times C' \xrightarrow{A \times view_1 \times C'} A \times B \times C' \xrightarrow{A \times upd_2} A \times B' \xrightarrow{upd_1} A'$$

$$view(a) = view_2(view_1(a))$$

$$upd(a, c') = upd_1(a, upd_2(view_1(a), c))$$

# Exercise:

Starting from a monoidal category, where in the definition of Lens(C) are we using the fact that morphism of C are comonoid homomorphisms?



# Examples of Lenses

# Derivatives as lenses

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 3x_1^2 + 7x_2$$

$$\nabla f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} 6x_1 \\ 7 \end{bmatrix}$$

$$\nabla f : \mathbb{R}^2 \times \mathbb{R} \longrightarrow \mathbb{R}^2$$

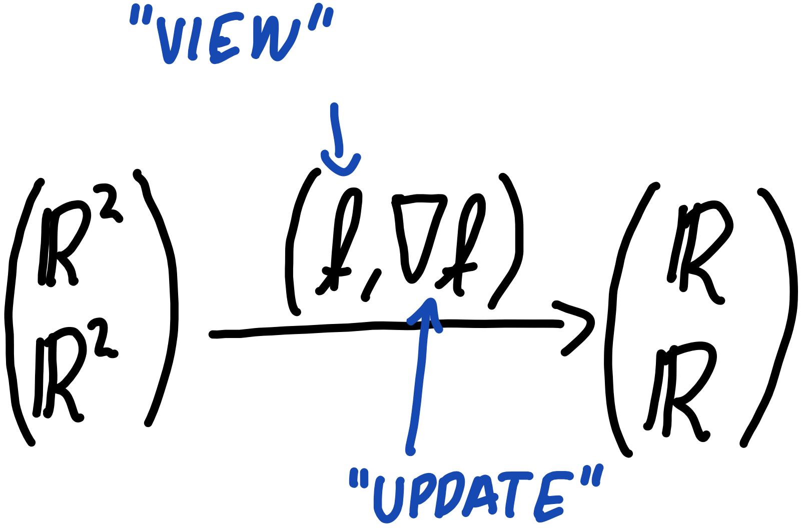
$$\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, dy\right) \longmapsto (6x_1 dy, 7dy)$$

BACKWARDS PART OF A LENS

# Derivatives as lenses

THIS IS A LENS!

A MORPHISM IN $\mathbf{Lens}(\mathbf{Euc})$ $(\mathbf{Euc}, \times, 1)$



# Chain rule as lens composition

$$\mathbb{R}^2 \xrightarrow{\;f\;} \mathbb{R} \xrightarrow{\;\cos\;} \mathbb{R}$$

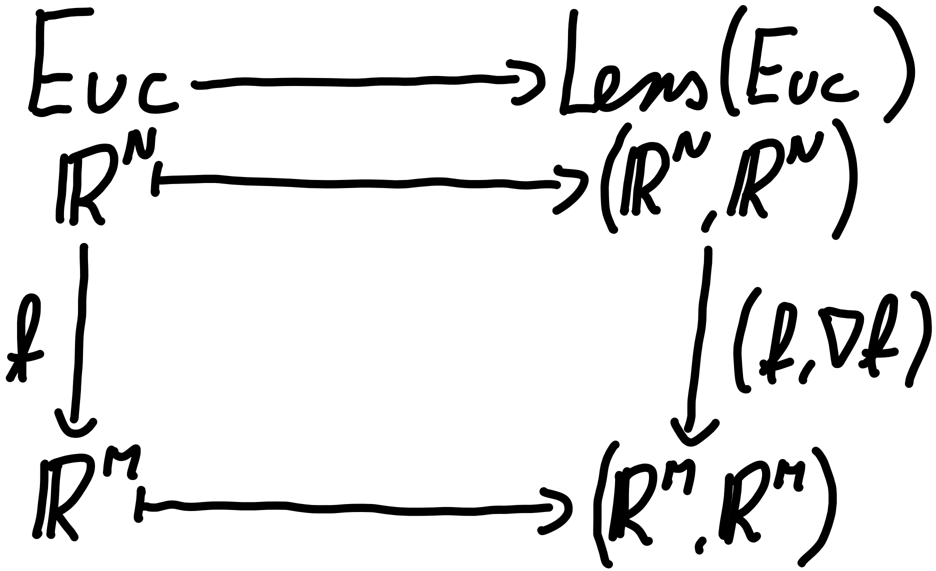
$$\nabla\cos : \mathbb{R}\times\mathbb{R} \longrightarrow \mathbb{R}$$
$$(x, dz) \longmapsto (\sin x)\cdot dz$$

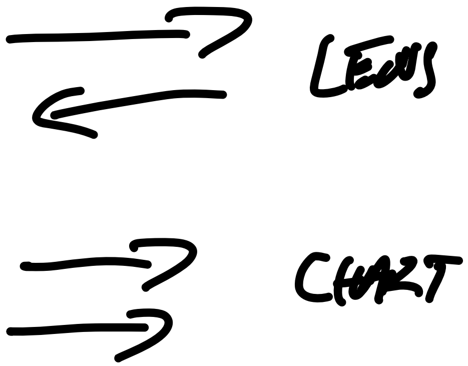
$$\begin{pmatrix}\mathbb{R}^2\\ \mathbb{R}^2\end{pmatrix} \xrightarrow{\;(f,\nabla f)\;} \begin{pmatrix}\mathbb{R}\\ \mathbb{R}\end{pmatrix} \xrightarrow{\;(\cos,\nabla\cos)\;} \begin{pmatrix}\mathbb{R}\\ \mathbb{R}\end{pmatrix}$$

$$upd(a, c') = upd_1(a, upd_2(view_1(a), c'))$$
$$\nabla f\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}, \nabla\cos\left(f\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right), dy\right)\right)$$
$$= \nabla f\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}, \sin(3x_1^2+7x_2)\cdot dy\right)$$
$$= \left(6x_1 \sin(3x_1^2+7x_2)\cdot dy,\; 7\sin(3x_1^2+7x_2)\cdot dy\right)$$

$$\nabla f : \mathbb{R}^2\times\mathbb{R} \longrightarrow \mathbb{R}^2$$
$$\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}, dy\right) \longmapsto (6x_1 dy, 7dy)$$

# Backprop: functor Euc → Lens(Euc)




On Euclidean spaces. For manifolds, we need dependent lenses.

# Optimisers as lenses

GRADIENT DESCENT

$$\begin{pmatrix} \mathbb{R}^p \\ \mathbb{R}^p \end{pmatrix} \xrightarrow{(\mathrm{view},\, \mathrm{upd})} \begin{pmatrix} \mathbb{R}^p \\ \mathbb{R}^p \end{pmatrix}$$

$$\mathrm{view}(w) = w$$

$$\mathrm{upd}(w, \Delta w) = w - \alpha \cdot \Delta w$$

# Optimisers as lenses

MOMENTUM

$$\begin{pmatrix}\mathbb{R}^p \times \mathbb{R}^p \\ \mathbb{R}^p \times \mathbb{R}^p\end{pmatrix} \overset{(view,\, upd)}{\rightleftarrows} \begin{pmatrix}\mathbb{R}^p \\ \mathbb{R}^p\end{pmatrix}$$

$$view(v_{t-1}, w_t) = w_t$$

$$upd(v_{t-1}, w_t, \Delta w_t) = (v_t, w_{t+1})$$

$$\text{where } v_t = \gamma v_{t-1} + \alpha \Delta w_t$$

$$w_{t+1} = w_t - v_t$$

# Optimisers as lenses

NESTEROV MOMENTUM

$$\begin{pmatrix} \mathbb{R}^p \times \mathbb{R}^p \\ \mathbb{R}^p \times \mathbb{R}^p \end{pmatrix} \underset{\longleftarrow}{\xrightarrow{(\mathrm{view},\, \mathrm{upd})}} \begin{pmatrix} \mathbb{R}^p \\ \mathbb{R}^p \end{pmatrix}$$

$$\mathrm{view}(v_{t-1}, w_t) = w_t - \gamma v_{t-1}$$

$$\mathrm{upd}(v_{t-1}, w_t, \Delta w_t) = (v_t, w_{t+1})$$

$$\text{where } v_t = \gamma v_{t-1} + \alpha \Delta w_t$$

$$w_{t+1} = w_t - v_t$$

# Moore machines as lenses

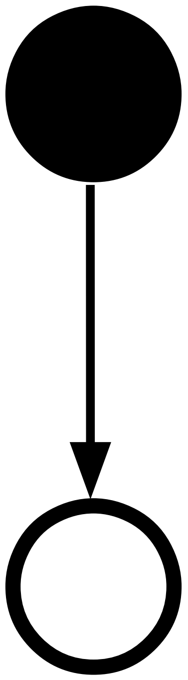
$$o : S \longrightarrow O$$

$$u : S \times I \longrightarrow S$$

$$\begin{pmatrix} S \\ S \end{pmatrix} \xrightarrow{(o,\, u)} \begin{pmatrix} O \\ I \end{pmatrix}$$

# But we can do more

- There's a particular way of categorically looking at bidirectional processes
- More general form, not restricted to deterministic processes
- Gives us an insight into the *internal* workings of lenses



# Optics

# Optics

- Optics model *contextual* bidirectional transformations
  - Probabilistic bidirectional transformations
  - Bidirectional transformations with side-effects
  - Bidirectional transformations that operate on containers

LENSES –||– CARTESIAN MONOIDAL CATEGORY

- Optics assume the base is a symmetric monoidal category C

# Optics: Definition

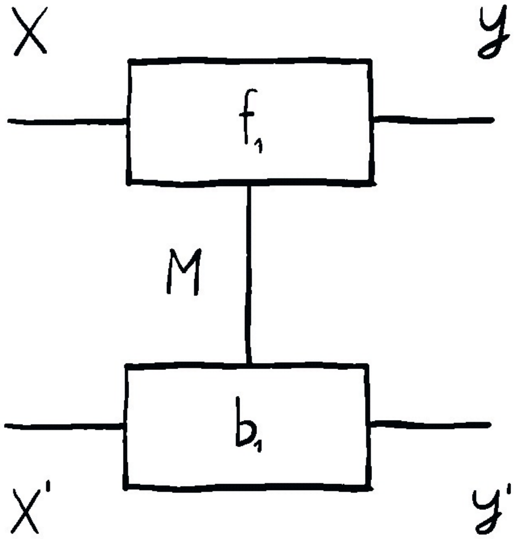
DEF. THE CATEGORY $\mathbf{Optic}(\mathcal{C})$

- OBJECTS - PAIRS OF OBJECTS IN $\mathcal{C}$ $\binom{A}{A'}$
- MORPHISM $\binom{A}{A'} \longrightarrow \binom{B}{B'}$ IS A TRIPLE $(M, f, b)$ WHERE:

$M : \mathcal{C}$ IS AN OBJECT OF $\mathcal{C}$

$$\left.\begin{array}{l} f : A \longrightarrow M \otimes B \\ b : M \otimes B' \longrightarrow A' \end{array}\right\} \text{ ARE MORPHISMS IN } \mathcal{C}.$$

# How do optics compose?



# Composition formula

THE COMPOSITE OF

$$\begin{pmatrix} A \\ A' \end{pmatrix} \xrightarrow{(M_1, f_1, b_1)} \begin{pmatrix} B \\ B' \end{pmatrix} \text{ AND } \begin{pmatrix} B \\ B' \end{pmatrix} \xrightarrow{(M_2, f_2, b_2)} \begin{pmatrix} C \\ C' \end{pmatrix} \text{ WHERE}$$

$M_1 : \mathcal{C}$ $\qquad$ $M_2 : \mathcal{C}$

$f_1 : A \longrightarrow M \otimes B$ $\qquad$ $f_2 : B \longrightarrow M \otimes C$

$b_1 : M \otimes B' \longrightarrow A'$ $\qquad$ $b_2 : M \otimes C' \longrightarrow B'$

IS

# Optic(Set)

# Optic(C) ≅ Lens(C) when C is Cartesian

# Optic(C) ≅ Lens(C) when C is Cartesian

- Exercise: Prove that there is a functor G:Lens(C) -> Optic(C) such that F and G form an isomorphism.

# Optics allow us to do more

# Optic(FinStoch)

## Bayesian open games

Joe Bolt, Jules Hedges, and Philipp Zahn

This paper generalises the treatment of compositional game theory as introduced by the second and third authors with Ghani and Winschel, where games are modelled as morphisms of a symmetric monoidal category. From an economic modelling perspective, the existing notion of an open game is not expressive enough for many applications. This includes stochastic environments, stochastic choices by players, as well as incomplete information regarding the game being played. The current paper addresses these three issue all at once. To achieve this we make significant use of category theory, especially the 'coend optics' of Riley.

# Motivation for Optics is 2-categorical!

## Space-time tradeoffs of lenses and optics via higher category theory

Bruno Gavranović

September 21, 2022

**Abstract**

Optics and lenses are abstract categorical gadgets that model systems with bidirectional data flow. In this paper we observe that the denotational definition of optics – identifying two optics as equivalent by observing their behaviour *from the outside* – is not suitable for operational, software oriented approaches where optics are not merely observed, but built with their internal setups in mind. We identify operational differences between denotationally isomorphic categories of cartesian optics and lenses: their different composition rule and corresponding space-time tradeoffs, positioning them at two opposite ends of a spectrum. With these motivations we lift the existing categorical constructions and their relationships to the 2-categorical level, showing that the relevant operational concerns become visible. We define the 2-category **2-Optic**($\mathcal{C}$) whose 2-cells explicitly optics' internal configuration. We show that the 1-category **Optic**($\mathcal{C}$) arises by locally quotienting out the connected components of this 2-category. We show that the embedding of lenses into cartesian optics gets weakened from a functor to an oplax functor whose oplaxator now detects the different composition rule. We determine the difficulties in showing this functor forms a part of an adjunction in any of the standard 2-categories. We establish a conjecture that the well-known isomorphism between cartesian lenses and optics arises out of the lax 2-adjunction between their double-categorical counterparts. In addition to presenting new research, this paper is also meant to be an accessible introduction to the topic.

# Value iteration

Value iteration is optic composition

Jules Hedges Riu Rodríguez Sakamoto

Dynamic programming is a class of algorithms used to compute optimal control policies for Markov decision processes. Dynamic programming is ubiquitous in control theory, and is also the foundation of reinforcement learning. In this paper, we show that value improvement, one of the main steps of dynamic programming, can be naturally seen as composition in a category of optics, and intuitively, the optimal value function is the limit of a chain of optic compositions. We illustrate this with three classic examples: the gridworld, the inverted pendulum and the savings problem. This is a first step towards a complete account of reinforcement learning in terms of parametrised optics.

# Bayes' Law!

## Bayesian Updates Compose Optically

Toby St. Clere Smithe

Department of Experimental Psychology,
University of Oxford
arxiv@tsmithe.net

July 29, 2020

Bayes' rule tells us how to invert a causal process in order to update our beliefs in light of new evidence. If the process is believed to have a complex compositional structure, we may ask whether composing the inversions of the component processes gives the same belief update as the inversion of the whole. We answer this question affirmatively, showing that the relevant compositional structure is precisely that of the *lens* pattern, and that we can think of Bayesian inversion as a particular instance of a state-dependent morphism in a corresponding fibred category. We define a general notion of (mixed) Bayesian lens, and discuss the (un)lawfulness of these lenses when their contravariant components are exact Bayesian inversions. We prove our main result both abstractly and concretely, for both discrete and continuous states, taking care to illustrate the common structures.

# Summary!

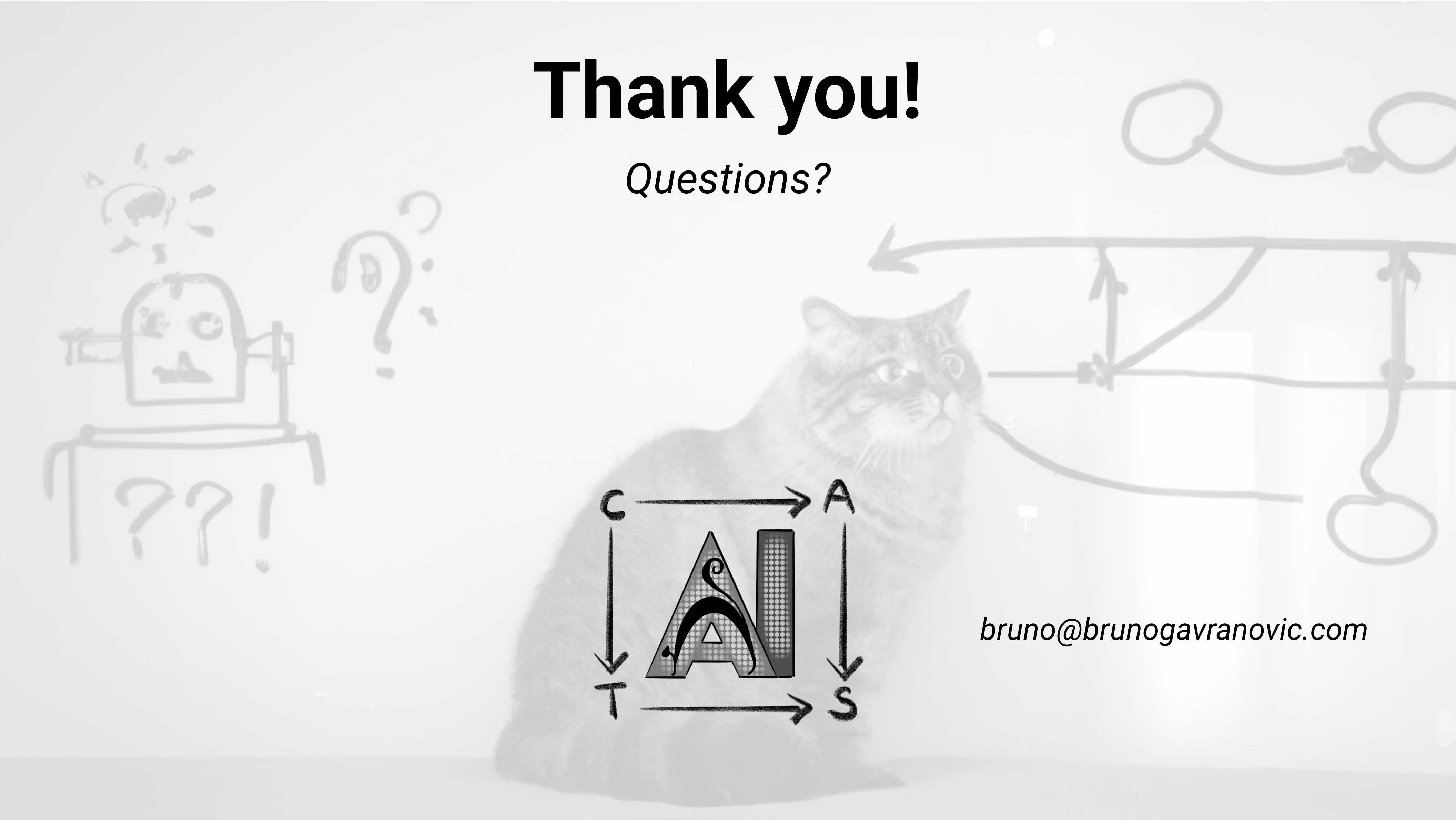
- Category theory gives us an rich language for describing processes found in neural networks

- Monoidal categories with supplies of comonoids, monoids

- With Optics, uniform way to model
    - Backpropagation
    - Bayes' Law
    - Value iteration

# Thank you!

*Questions?*



*bruno@brunogavranovic.com*