Graphical Linear Algebra

Midlands Graduate School 2017

Pawel Sobocinski
University of Southampton

joint work with Filippo Bonchi, and Fabio Zanasi

contributions from Dusko Pavlovic,
Fabio Gadducci, Aleks Kissinger, Brendan Fong and Paolo Rapisarda

graphicallinearalgebra.net

What will you get out of this?

- An introduction to symmetric monoidal categories and *diagrammatic reasoning*, an important tool in
    - quantum computing and quantum information (Abramsky, Coecke, Pavlovic, Duncan, Kissinger, …)
    - asynchronous circuits (Ghica)
    - signal flow graphs (Bonchi, S., Zanasi)
    - linear time-invariant dynamical systems (Baez, Erbele, Fong, Rapisarda, S.)
- A different way to think about several concepts of linear algebra!
    - linear algebra is **everywhere** (quantum, machine learning, systems and control theory, graph theory, …)
    - a different language means different ways to think about applications, and it is sometimes more efficient than the classical concepts of matrices, bases, etc.
    - a nice calculus for recurrence relations
- An introduction to several algebraic structures that are quite common in computational models
    - e.g. monoids, comonoids, Frobenius monoids, bimonoids, Hopf algebras, …

More generally: resource sensitive syntax

- Ordinary, tree-like syntax, together with its operations (tuples, substitution) is good in situations where all operations have a coarity 1
  - i.e. operations of the form $X^n \to X$
- This often goes together with the implicit assumption that the resources in X are ***copyable*** and ***discardable***
- In many situations (e.g. quantum, many kinds of circuits) we have to be more careful with resources — here tree-like syntax can unnatural
  - the language of symmetric monoidal categories provides the algebra for this syntax
  - the language of string diagrams provides a nice notation

Plan

- **Lecture 1 - String diagrams and symmetric monoidal categories**

- **Lecture 2** - Resource-sensitive algebraic theories

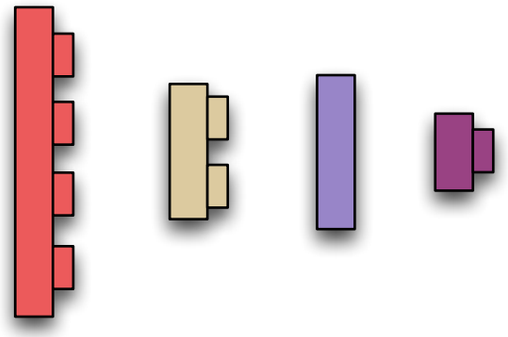
- **Lecture 3** - Interacting Hopf monoids and graphical linear algebra

- **Lecture 4** - Signal Flow Graphs and recurrence relations

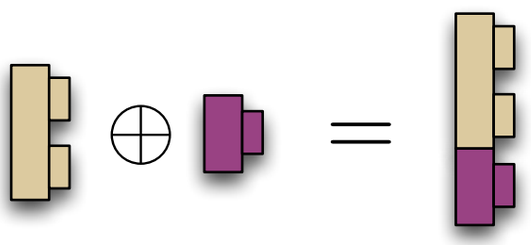
Lecture 1

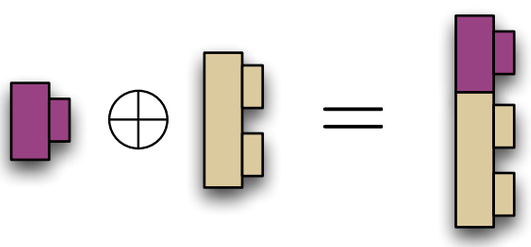
String diagrams and symmetric monoidal categories

“stacking things and plugging them together”



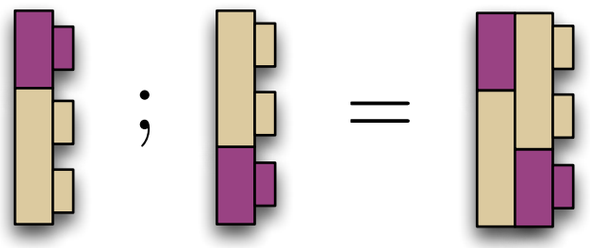
⊕ - stacking one on top of the other

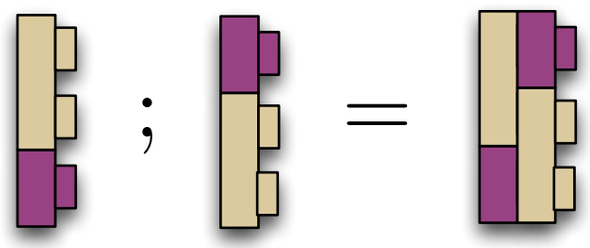




associative but not commutative

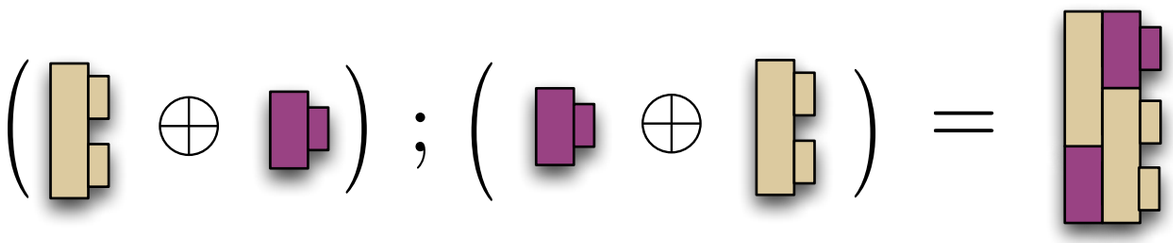
; - plugging the first into the second

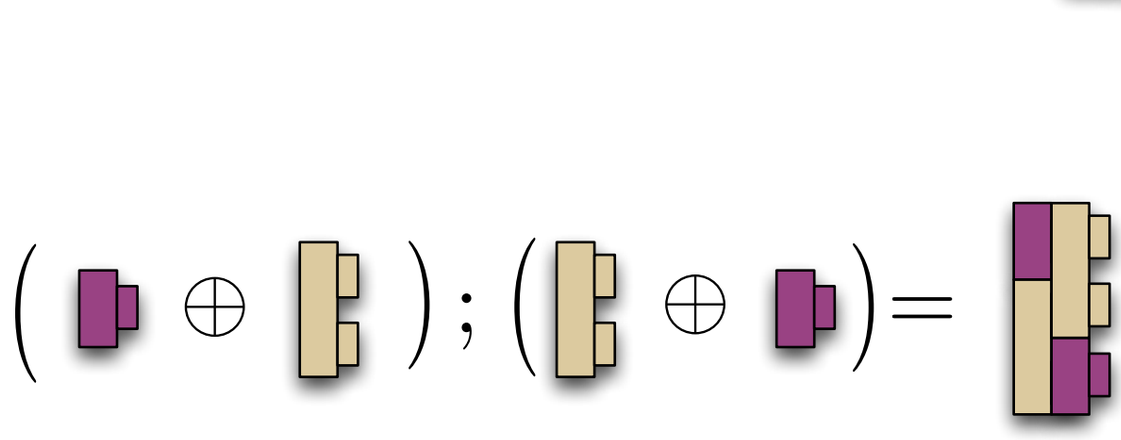




associative but not commutative

Combining operations

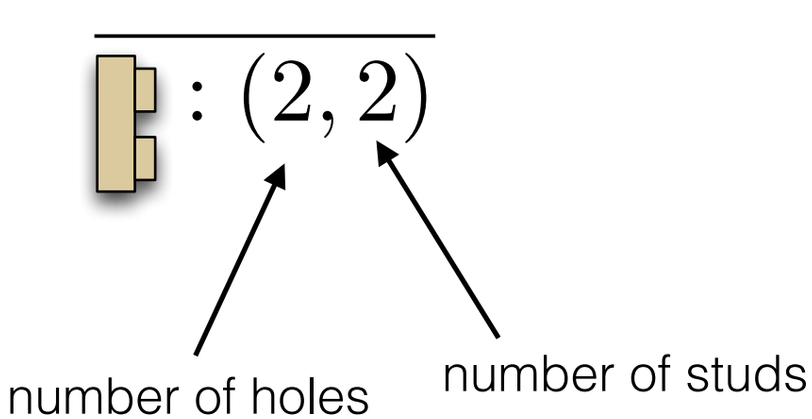


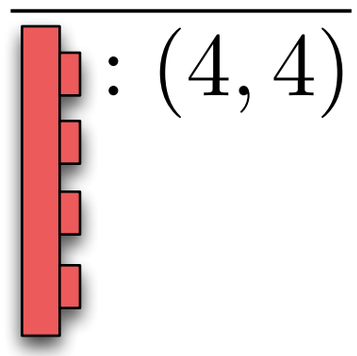


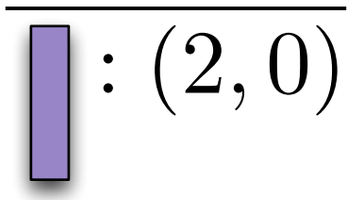
Plan

- **algebra of magic Lego**
- categories
- monoidal categories
- string diagrams
- symmetric monoidal categories

A simple type system





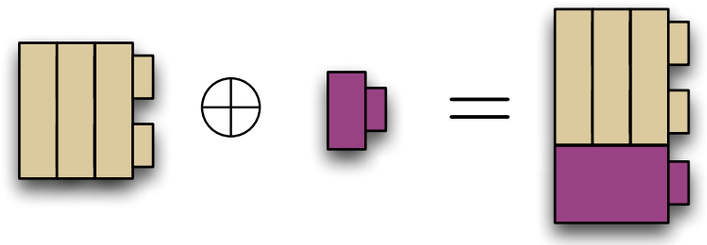


$$\frac{X : (k, l) \qquad Y : (m, n)}{X \oplus Y : (k+m, l+n)}$$

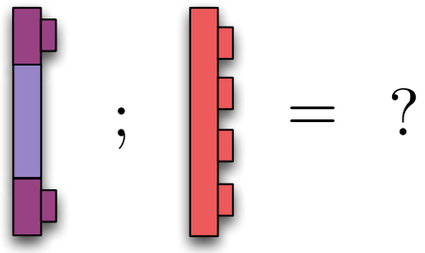
$$\frac{X : (k, l) \qquad Y : (l, m)}{X ; Y : (k, m)}$$

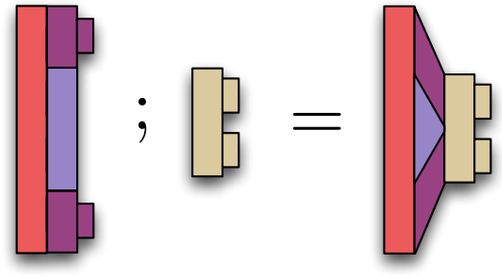
The magic in magic Lego

$$\frac{X : (k, l) \qquad Y : (m, n)}{X \oplus Y : (k+m, l+n)}$$



$$\frac{X : (k, l) \qquad Y : (l, m)}{X ; Y : (k, m)}$$





Pieces stretch and shrink as needed — the only important thing is the type (number of holes, number of studs)

The mathematics of magic Lego

- "stacking things on top of each other" and "plugging things together" are very common operations in science and mathematics

- the mathematical structure that's behind these operations is called a (strict) **monoidal category**

- let's start with categories…

Plan

- algebra of magic Lego
- **categories**
- monoidal categories
- string diagrams
- symmetric monoidal categories

Categories

- A *category* **C**, seen as a mathematical structure, consists of two sets: objects and arrows

- think of objects as **types**

- think of arrows as **operations**

  - every arrow C has two associated objects k, m, called **domain** and **codomain**, written $C : k \to m$

- Every object has an **identity arrow** (think of this as a trivial “do-nothing” operation)

$$\frac{}{I_k : k \to k}$$

Operations on arrows

- There is a partial operator on arrows ; called **composition**

$$\frac{C : k \to l \qquad D : l \to m}{C \,;\, D : k \to m}$$

— like the plugging in magic Lego

Associativity

- Composition is associative, i.e.

$$\frac{C : k \to l \qquad D : l \to m \qquad E : m \to n}{(C \,;\, D) \,;\, E = C \,;\, (D \,;\, E) : k \to n}$$

Composing identities

- Composing with identity arrows on both sides does not change the arrow, *i.e.*

$$\frac{C : m \to n}{I_m \,;\, C = C = C \,;\, I_n}$$

Examples of (small) categories

- **0** - the empty category
- **1** - the category with one object * and one arrow
  - *Exercise*: check that everything works
- **2** - the category with objects $\{0, 1\}$, one non-identity arrow $0 \to 1$.
  - *Exercise*: check that everything works
- Any preorder is a category with
  - objects the elements of X, and at most one arrow between any 2 objects: $x \to y$ iff $x \leq y$.
  - conversely, any category with at most one arrow from any object to another is a preorder.

Examples of (big) categories

- **Set**
  - *Objects*: sets
  - *Arrows*: functions
- **$\mathbf{Set}_f$**
  - *Objects*: finite sets
  - *Arrows*: functions
- **Rel**
  - *Objects*: sets
  - *Arrows*: relations
- **Mon**
  - *Objects*: monoids
  - *Arrows*: homomorphisms
- **Top**
  - *Objects*: topological spaces
  - *Arrows*: continuous functions
- **Cat**
  - *Objects*: categories
  - *Arrows*: functors

Plan

- algebra of magic Lego
- categories
- **monoidal categories**
- string diagrams
- symmetric monoidal categories

In the spirit of HOTT

- A 2-category has
  - objects “0-cells”
  - arrows “1-cells”
    - 2-cells (arrows between arrows)
- so that (1 and 2) composition is associative and has identities
- e.g. Topological space: objects = points, arrows = paths, 2-cells = homotopies
- two different ways of composing 2-cells!
- **A (strict) monoidal category is a 2-category with one object**

Monoidal categories

- A (strict) *monoidal category* **C**, seen as a mathematical structure, consists of two sets: objects and arrows
  - think of objects as (products of) **types**
  - think of arrows as **operations**
    - every arrow C has two associated objects k, m, called **domain** and **codomain**, written $C : k \to m$

- there is an associative operation on objects $\oplus$ called **monoidal product**

$$(k \oplus m) \oplus n = k \oplus (m \oplus n)$$

- Every object has an **identity arrow** (think of this as a trivial “do-nothing” operation)

$$\frac{}{I_k : k \to k}$$

- There is an **identity object** I for monoidal product (think of this as a unit type). This makes the set of object a **monoid**.

$$\frac{}{k \oplus I = I \oplus k = k}$$

Operations on arrows

- There is a partial operation or arrows ; called **composition**

$$\frac{C : k \to l \qquad D : l \to m}{C ; D : k \to m}$$

— like the plugging in magic Lego

- There is a total operation on arrows $\oplus$ (abusing notation) also called **monoidal product**.

$$\frac{C : k \to l \qquad D : m \to n}{C \oplus D : k \oplus m \to l \oplus n}$$

— like the stacking in magic Lego

Associativity

- Composition is associative, i.e.

$$\frac{C : k \to l \quad D : l \to m \quad E : m \to n}{(C ; D) ; E = C ; (D ; E) : k \to n}$$

- Monoidal product is associative, also on arrows, i.e.

$$\frac{C : m \to n \quad D : m' \to n' \quad E : m'' \to n''}{(C \oplus D) \oplus E = C \oplus (D \oplus E) : m \oplus m' \oplus m'' \to n \oplus n' \oplus n''}$$

Monoidal product and functoriality

- The monoidal product is actually a **functor**, a morphism of categories

$$F: \mathbf{X} \to \mathbf{Y}$$

- Functors consist of two functions, one each for objects and arrows, and preserve **domains** and **codomains** (*c.f.* morphism of digraphs)

$$\frac{C : m \to n}{F(C) : F(m) \to F(n)}$$

- as well as **composition** and **identities**

$$F(C \mathbin{;} D) = F(C) \mathbin{;} F(D)$$

$$F(I_k) = I_{F(k)}$$

How is monoidal product a functor?

- A *monoid* is a set M equipped with a **function**

$$\cdot : M \times M \to M$$

(M×M is the cartesian product of sets)

- … with a unit element, satisfying associativity

- Similarly, a *strict monoidal category* is a category C equipped with a **functor**

$$\oplus : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$$

(**C**×**C** is the cartesian product of categories)

- … with a unit object, satisfying associativity

Exercise

- Suppose that **C** and **D** are categories. How do we define of the cartesian product **C**×**D** of **C** and **D**?
  - *What are the objects?*
  - *What are the arrows?*
  - *How does composition work?*
  - *What are the identity arrows?*

Monoidal product and composition

- Suppose that A can be composed with C, and B can be composed with D. Then since $\oplus$, being a functor, preserves composition, we have:

$$\oplus(A;B,\ C;D) = \ \oplus(\ (A,C)\ ;\ (B,D)\ ) = \oplus(A,C)\ ;\ \oplus(B,D)$$

i.e.

$$(A\ ;\ B) \oplus (C\ ;\ D) = (A \oplus C)\ ;\ (B \oplus D)$$

This equation is sometimes called “middle-four interchange”

Exercise

- Suppose that A: $p \rightarrow q$ and B: $r \rightarrow s$. Show that

$$(A \oplus I_r) ; (I_q \oplus B) = A \oplus B = (I_p \oplus B) ; (A \oplus I_s)$$

Composing identities

- Composing with identity arrows on both sides does not change the arrow, *i.e.*

$$C\colon m \to n$$

$$I_m \,;\, C = C = C \,;\, I_n$$

- Since $\oplus$ must preserve identities, we also require

$$\oplus(I_{(m,n)}) = \oplus(I_m,\ I_n) = I_{m \oplus n} \qquad \text{i.e.}$$

$$I_m \oplus I_n = I_{m \oplus n}$$

Examples of monoidal categories

- **Set**$_f$ with × (cartesian product) or with + (disjoint union) as monoidal product
  - it's not quite strict… e.g. monoidal product is not associative on the nose
  - it is a (non-strict) monoidal category, where associativity is up to coherent isomorphism
- **F** - the strict monoidal version of **Set**$_f$
  - *Objects*: finite ordinals $\underline{m} := \{1,2,\ldots,m\}$
  - *Arrows*: functions
  - Monoidal product 1: on objects $\underline{m} + \underline{n} := \underline{m + n}$
    - what is the monoidal identity object?
    - how to define monoidal product on morphisms?
  - Monoidal product 2: on objects $\underline{m} \times \underline{n} := \underline{m \times n}$
    - what is the monoidal identity object?
    - how to define monoidal product on morphisms?

Magic Lego

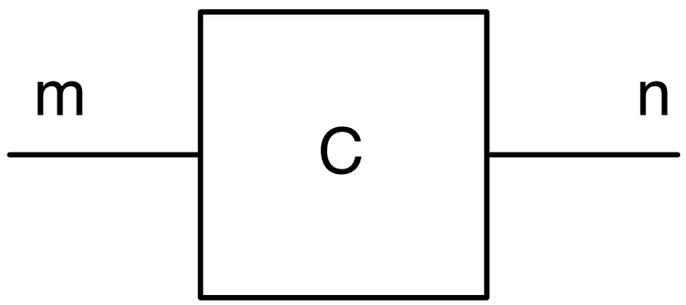
- Is a monoidal category with
  - *Objects*: natural numbers (keeping track of numbers of holes and studs)
  - *Arrows*: lego constructions

Plan

- algebra of magic Lego
- categories
- monoidal categories
- **string diagrams**
- symmetric monoidal categories

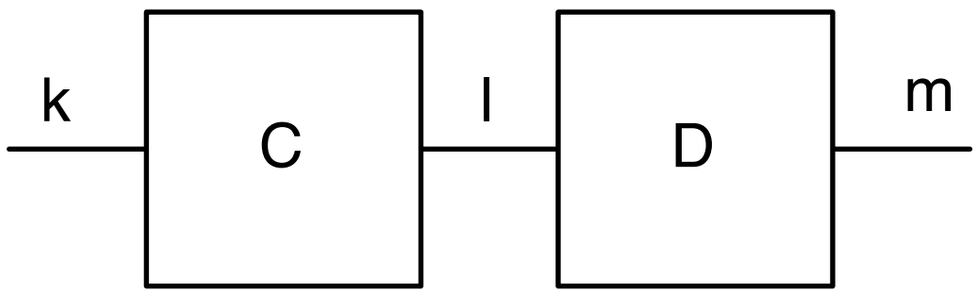
String diagrams

- A graphical notation for the arrows of monoidal categories
  - We have been writing $C\colon m \to n$
  - We will now **draw**



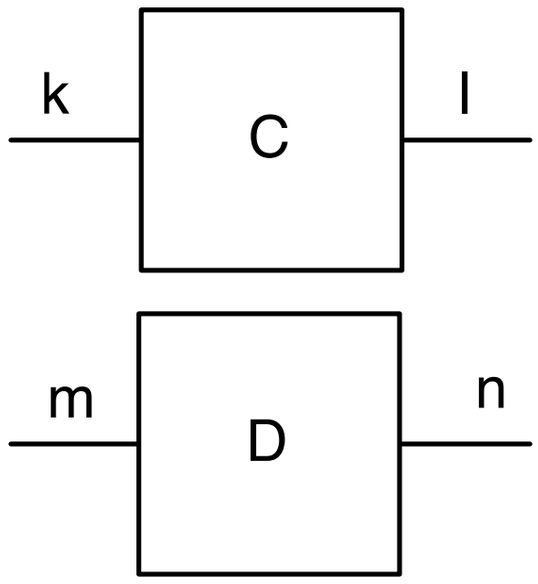
Composition

$$\frac{C : k \rightarrow l \qquad D : l \rightarrow m}{C ; D : k \rightarrow m}$$



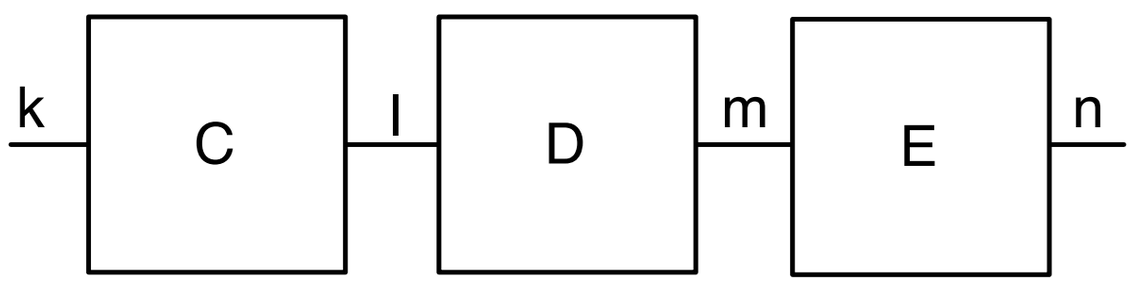
Monoidal product

$$\frac{C : k \to l \qquad D : m \to n}{C \oplus D : k \oplus m \to l \oplus n}$$



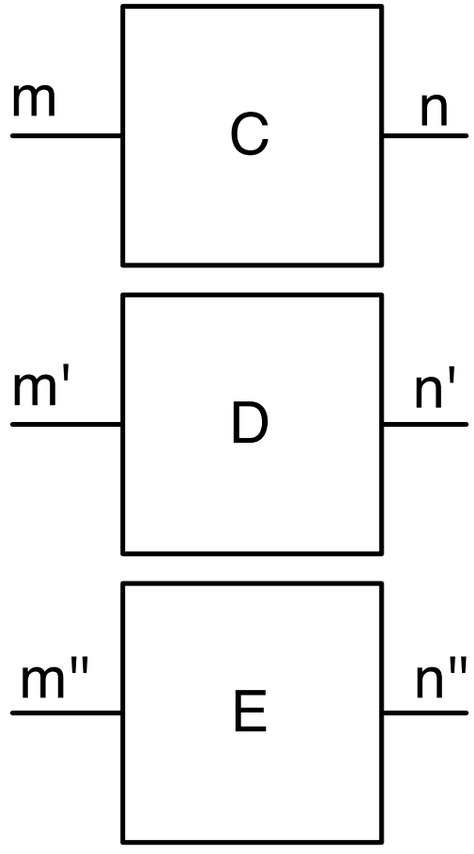
Perks of the notation I

$$\frac{C : k \to l \qquad D : l \to m \qquad E : m \to n}{(C \,;\, D) \,;\, E = C \,;\, (D \,;\, E) : k \to n}$$



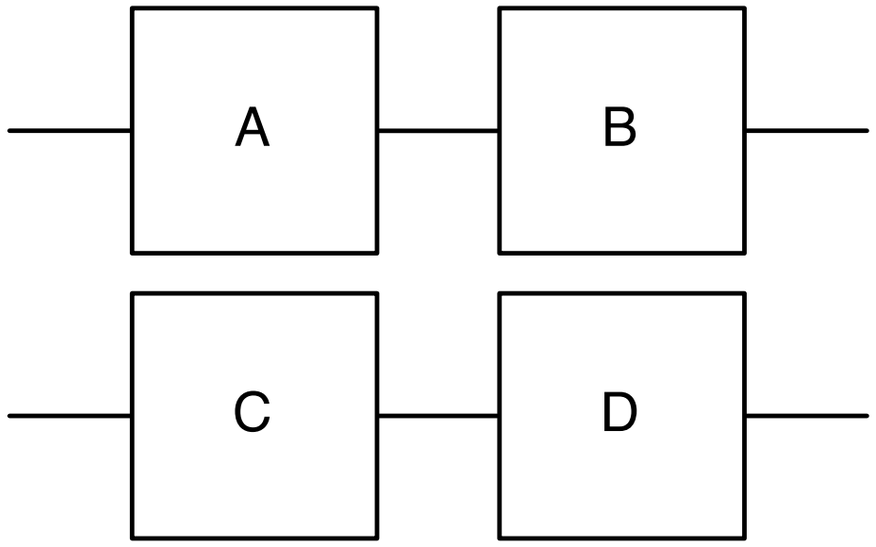
Perks of the notation II

$$\frac{C : m \to n \quad D : m' \to n' \quad E : m'' \to n''}{(C \oplus D) \oplus E = C \oplus (D \oplus E) : m \oplus m' \oplus m'' \to n \oplus n' \oplus n''}$$



Perks of the notation III

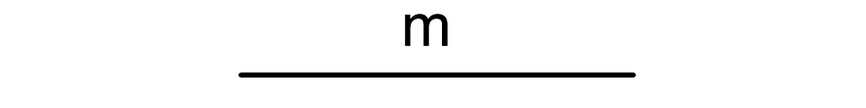
$$(A \,;\, B) \oplus (C \,;\, D) = (A \oplus C) \,;\, (B \oplus D)$$



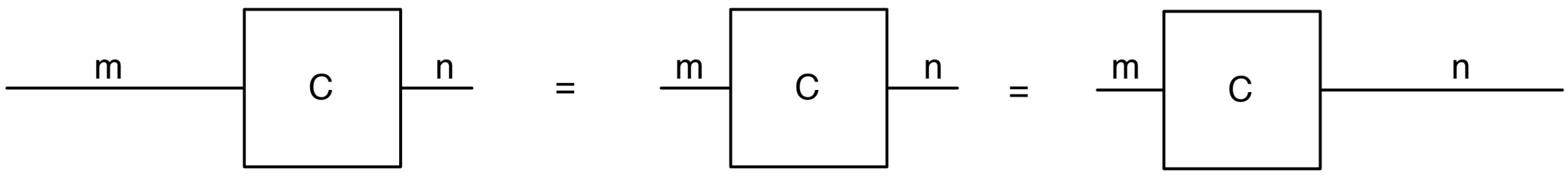
Diagrammatic reasoning I

Stretching

- Identity on *m* is simply drawn as a wire



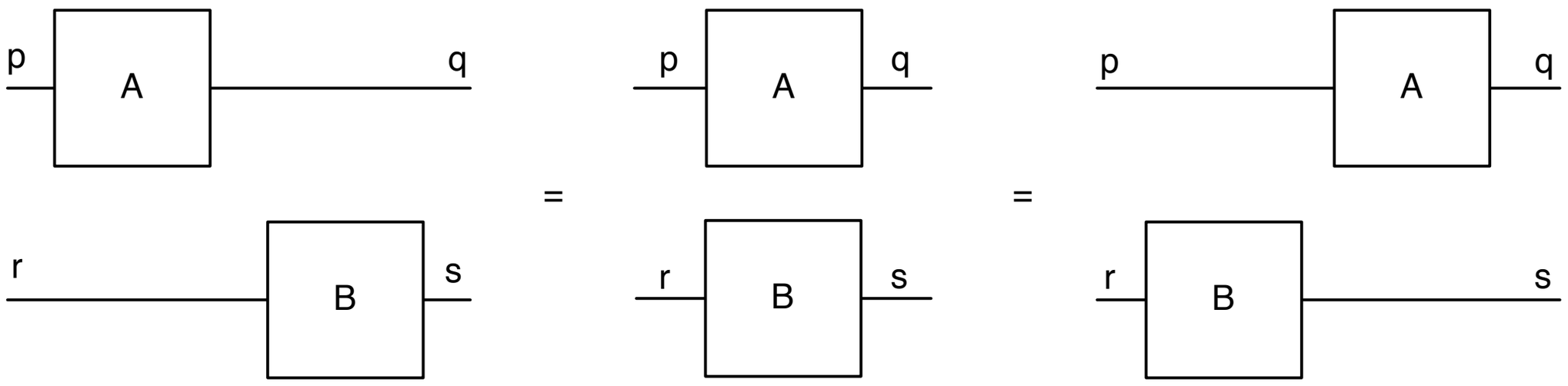
$$\frac{C : m \to n}{I_m \,;\, C = C = C \,;\, I_n}$$



Diagrammatic reasoning II

Sliding

$$(A \oplus I_r) ; (I_q \oplus B) = A \oplus B = (I_p \oplus B) ; (A \oplus I_s)$$



General story

- A strict monoidal category is the same thing as a 2-category with one object, a particularly simple kind of *higher category*
    - string diagrams are a kind of graph theoretical dual, i.e.
        - zero dimensional things (objects) become two dimensional things
        - one dimensional things (arrows in a 2-cat = objects in a strict monoidal cat) stay as one dimensional things — wires
        - two dimensional things (2-cells in a 2-cat = arrows in a strict monoidal cat) become zero dimensional things (points, or boxes as we have been drawing)
- See Globular (Vicary, Kissinger, Bar): http://globular.science

Plan

- algebra of magic Lego
- categories
- monoidal categories
- string diagrams
- **symmetric monoidal categories**

Symmetric monoidal categories

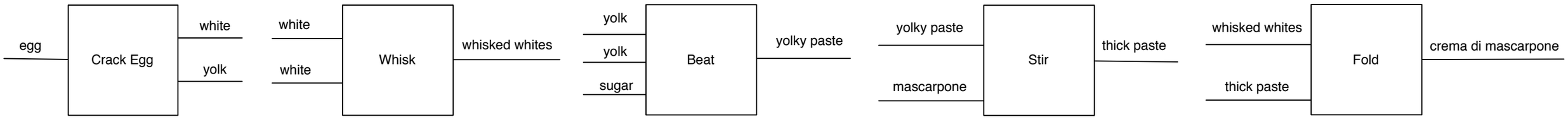
- When wiring things up using the algebra of connecting and stacking, we often want to permute the wires

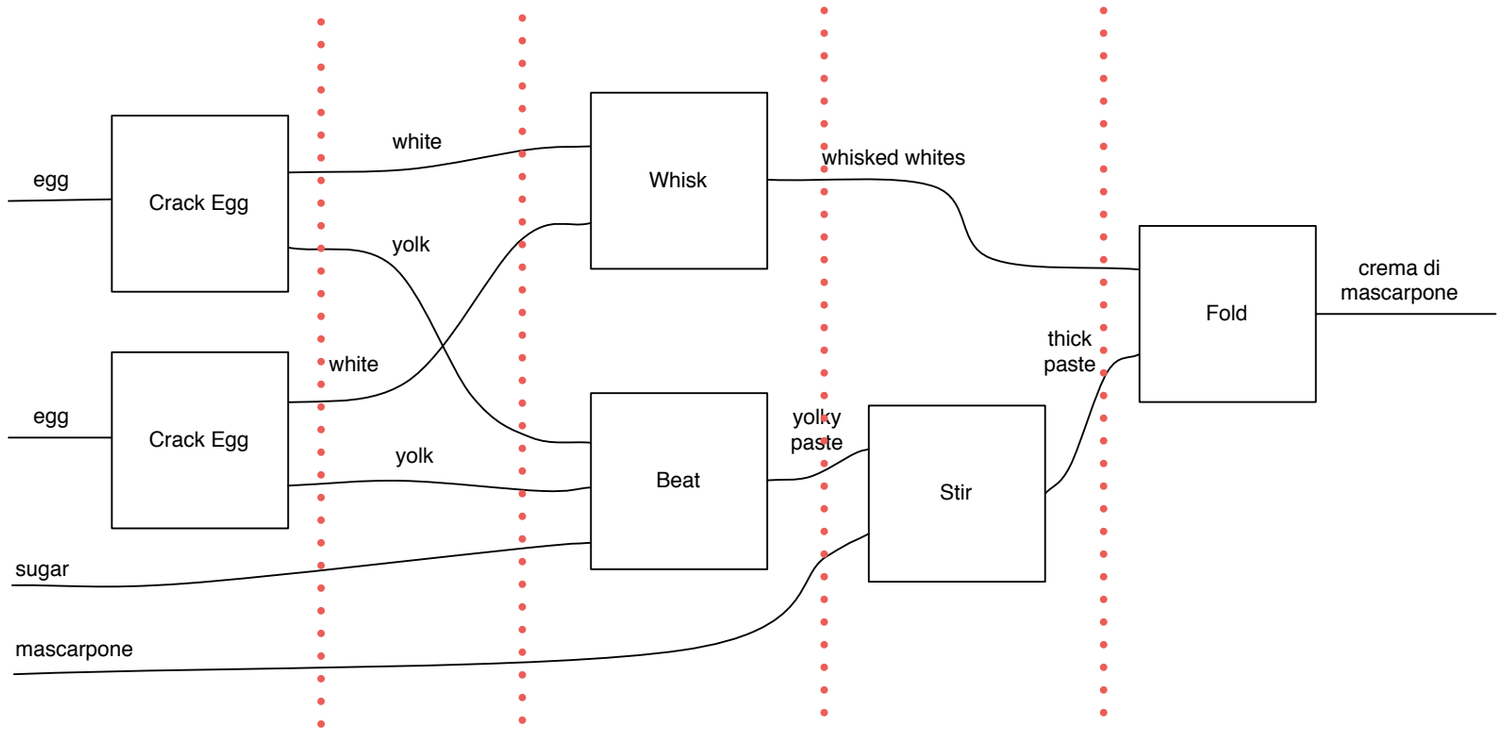
- Mathematically, this means moving from monoidal categories to *symmetric* monoidal categories

- In a symmetric monoidal category, for any two objects m, n, there is a *symmetry*, or *twist*

$$\mathrm{tw}_{m,n} \colon m \oplus n \to n \oplus m$$

Example - Crema di Mascarpone





$$(C \oplus C \oplus id_2) \; ; \; (id \oplus tw \oplus id_3) \; ; \; (W \oplus B \oplus id) \; ; \; (id \oplus S) \; ; \; F$$

Natural transformations

- We have seen categories, and functors: morphisms between categories
- A natural transformation is a *morphism between functors*

$$F \Rightarrow G: \mathbf{X} \rightarrow \mathbf{Y}$$

- A natural transformation $\alpha : F \Rightarrow G$ is a collection of arrows of **Y**, one for each object in $m \in \mathbf{X}$

$$\alpha_m : Fm \rightarrow Gm$$

- These must satisfy a condition with respect to the arrows of **X**, namely for each arrow C: $m \rightarrow n$

$$\begin{array}{ccc} Fm & \xrightarrow{\alpha_m} & Gm \\ {\scriptstyle FC}\downarrow & & \downarrow{\scriptstyle GC} \\ Fn & \xrightarrow{\alpha_n} & Gn \end{array}$$

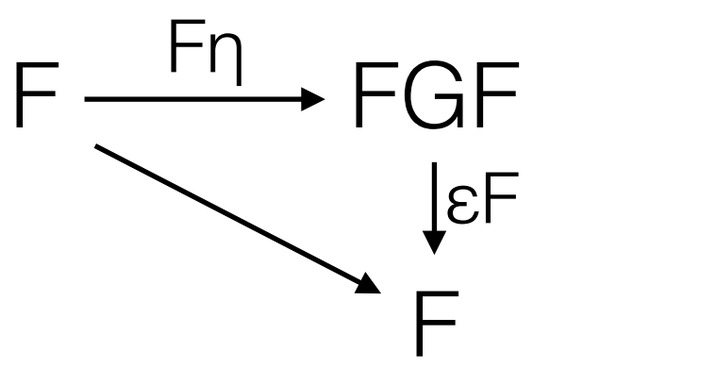
commutes.

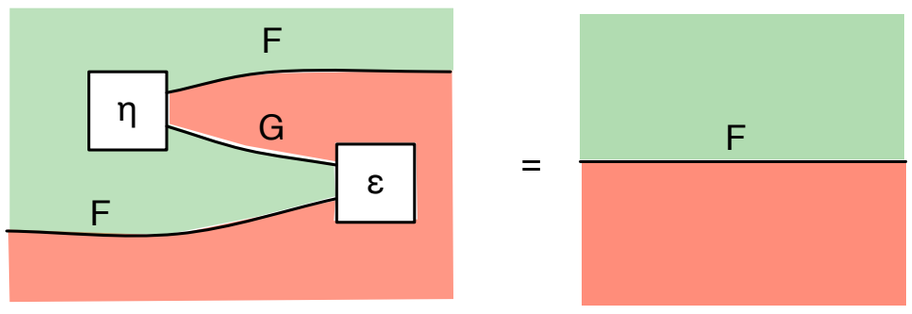
These are sometimes called naturality squares

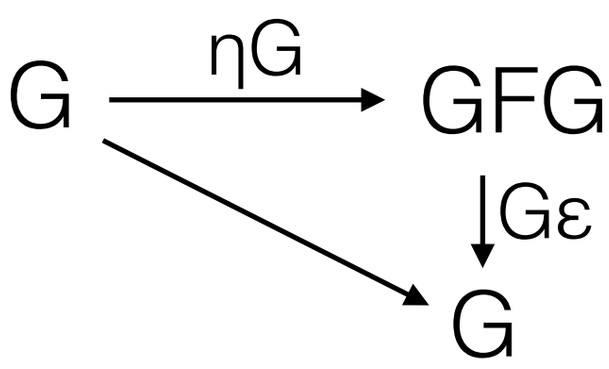
Aside — string diagrams for **Cat**

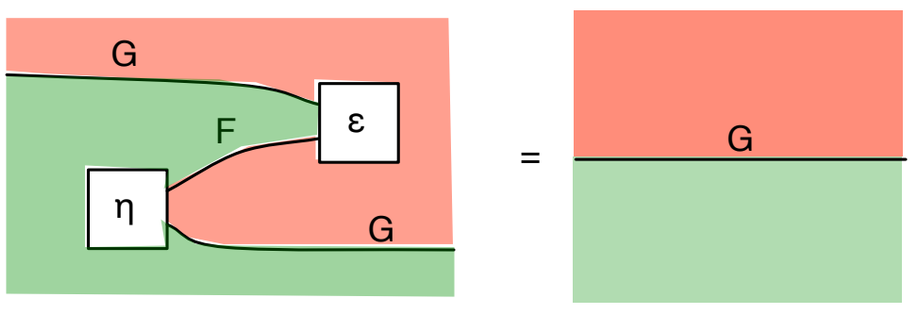
- **Cat** is a 2-category
  - *Objects (0-morphisms)*: categories
  - *Arrows (1-morphisms)*: functors
  - *2-cells (2-morphisms)*: natural transformations

- Suppose that F: **C**→**D** has right adjoint G: **D**→**C**, then the triangle equations can be drawn as follows, using string diagrams:









Symmetry as natural transformation

- For a monoidal category **C**, there are actually two functors $\mathbf{C}\times\mathbf{C} \to \mathbf{C}$ given by $\oplus$

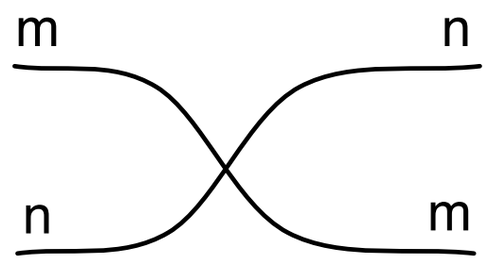
  $\oplus = -_1 \oplus -_2 : \mathbf{C}\times\mathbf{C} \to \mathbf{C}$ – stack the first on the second

  $\oplus' = -_2 \oplus -_1 : \mathbf{C}\times\mathbf{C} \to \mathbf{C}$ – stack the second on the first

- *tw* is a natural transformation from the first to the second, with components $tw_{m,n} : m \oplus n \to n \oplus m$

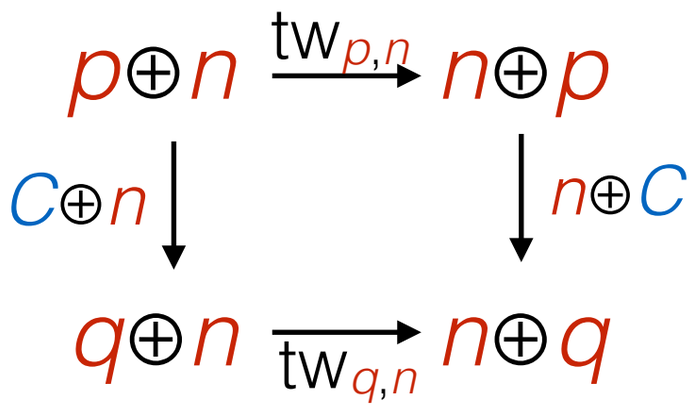
Drawing twists

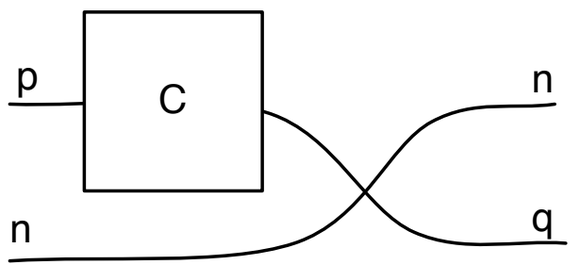
$\mathrm{tw}_{m,n}$: $m \oplus n \to n \oplus m$



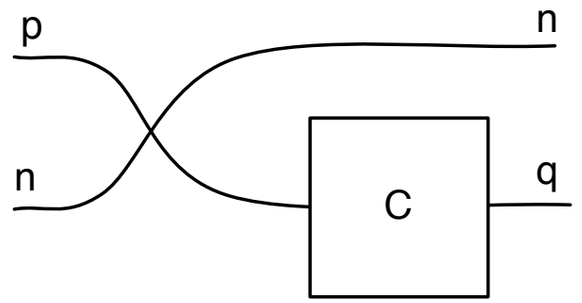
Diagrammatic reasoning III

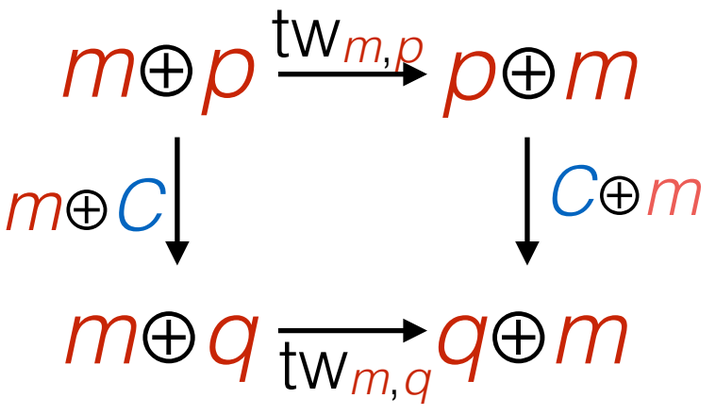
naturality = sliding across twists

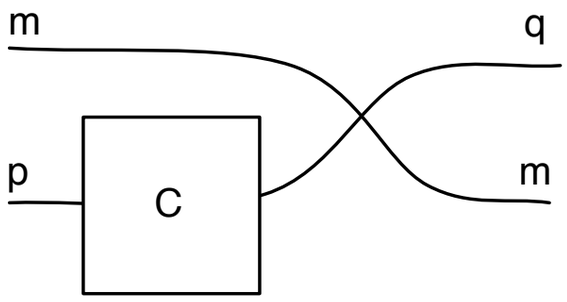




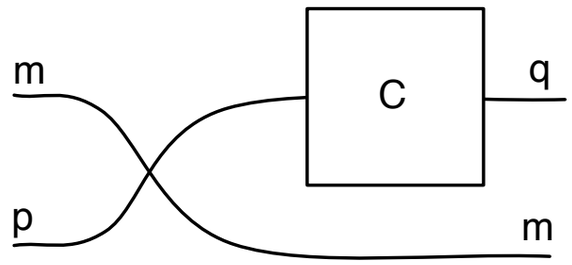
=







=

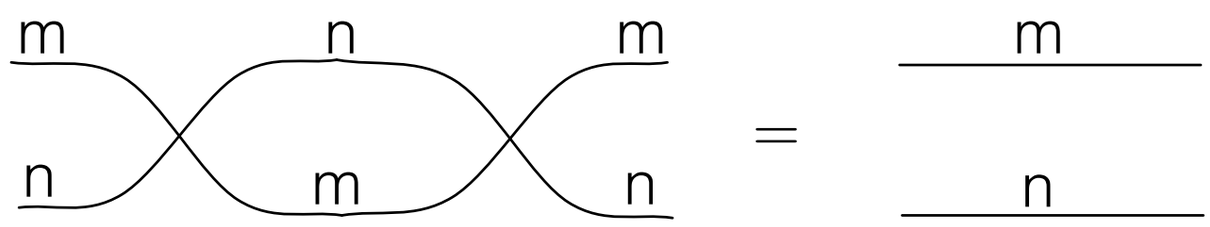


Diagrammatic reasoning IV

tightening (without wires tangling)

- In any symmetric monoidal category, the twist is **invertible**, and has itself as inverse, in the following sense

$$tw_{m,n} ; tw_{n,m} = I_{m \oplus n} : m \oplus n \to m \oplus n$$



In a braided monoidal category, the twist is invertible, but it is not, in general, it's own inverse

Symmetric monoidal categories

- A *strict symmetric monoidal* category **C**, is a strict monoidal category with a natural family of arrows $tw_{m,n} : m \oplus n \rightarrow n \oplus m$, indexed by pairs of objects of **C**, such that

$$tw_{m,n} ; tw_{n,m} = I_{m \oplus n} : m \oplus n \rightarrow m \oplus n$$

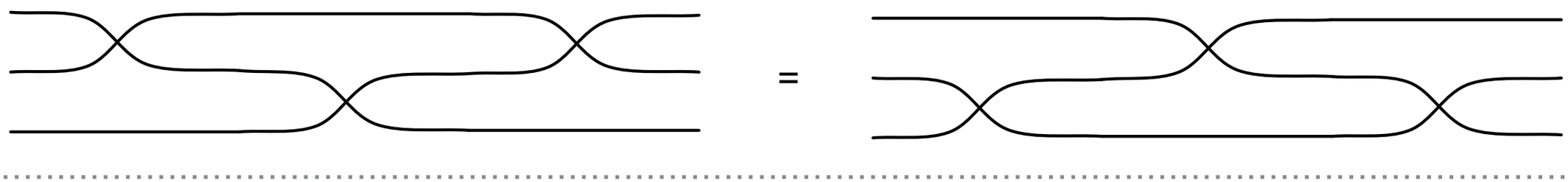
$$tw_{p,\, m \oplus n} = (tw_{p,m} \oplus I_n); (I_m \oplus tw_{p,n}): p \oplus m \oplus n \rightarrow m \oplus n \oplus p$$

$$tw_{m \oplus n,\, p} = (I_m \oplus tw_{n,p}) ; (tw_{p,m} \oplus I_n): m \oplus n \oplus p \rightarrow p \oplus m \oplus n$$

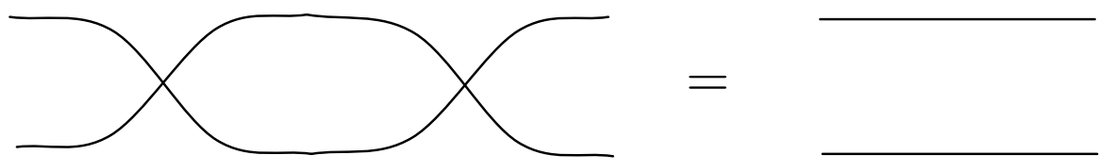
$$tw_{I,m} = I_m = tw_{m,I}$$

Yang-Baxter

- *Exercise*: The famous Yang-Baxter equation is an instance of naturality of the twist



- together with



we get that “pure” wiring diagrams $t^{\oplus m} \to t^{\oplus m}$ are in 1-1 correspondence with permutations of the m-element set

The category of permutations

- *Objects*: finite ordinals $\underline{m} = \{1,\ldots,m\}$
- *Arrows*: no arrows from m to n if $m \neq n$, otherwise the permutations

- Strict symmetric monoidal, with $\underline{m} \oplus \underline{n} := \underline{m+n}$