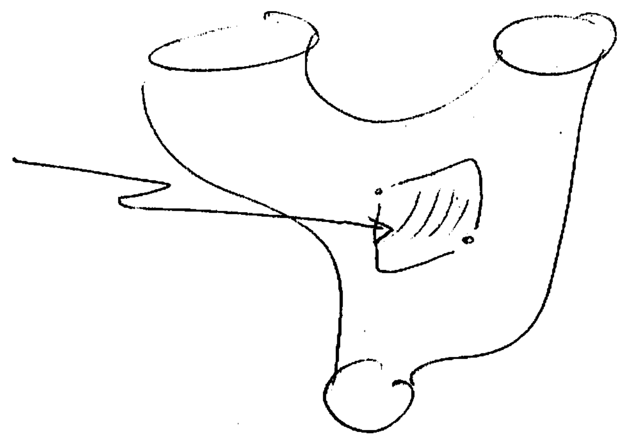
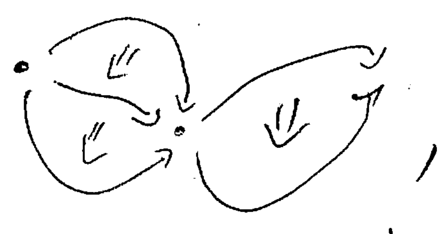
Second Part

o in which we pass from generators of 2d cobordisms

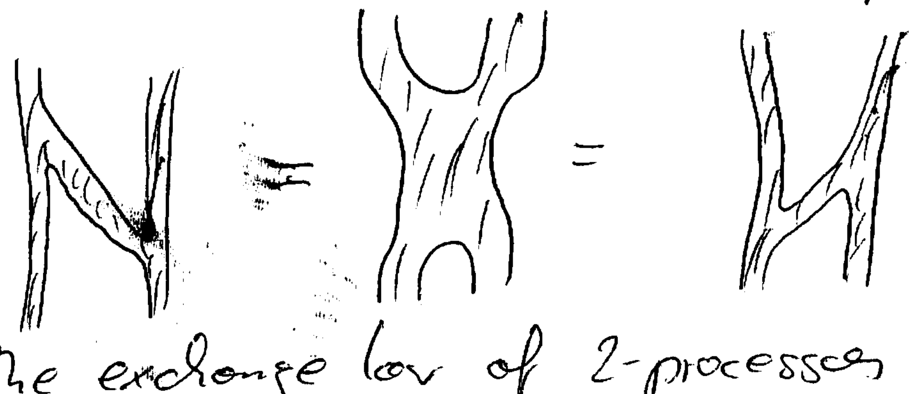
to "2-generators"



o and begin to think about "2-processes"



only to rediscover the Frobenius property



from the exchange law of 2-processes

Last time we saw

a Quantum Field Theory
is a representation of
a Cobordism category

$$QFT : nCob_S \longrightarrow Vect$$

possibly with extra Structure like conformal, Riemannian, etc.

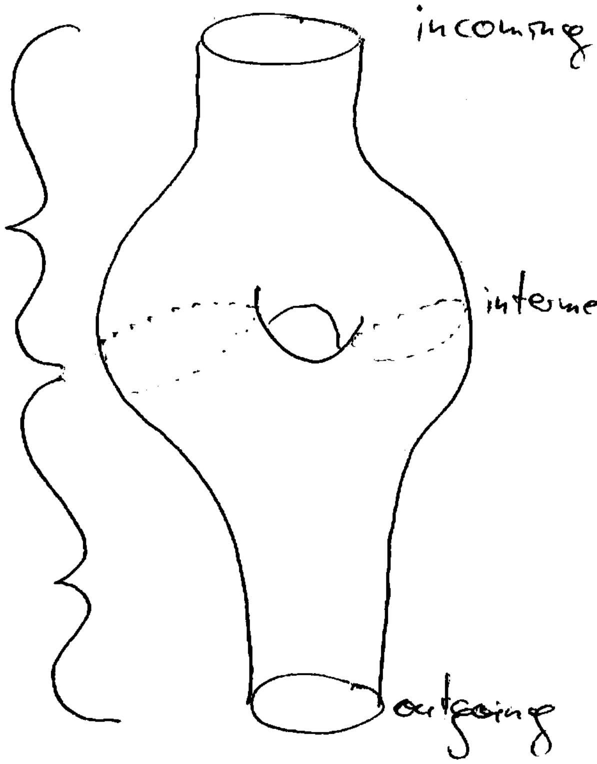
possibly with extra structure (topological, Hilbert, etc.)

For topological QFT we can understand such functors by finding generators for all cobordisms.

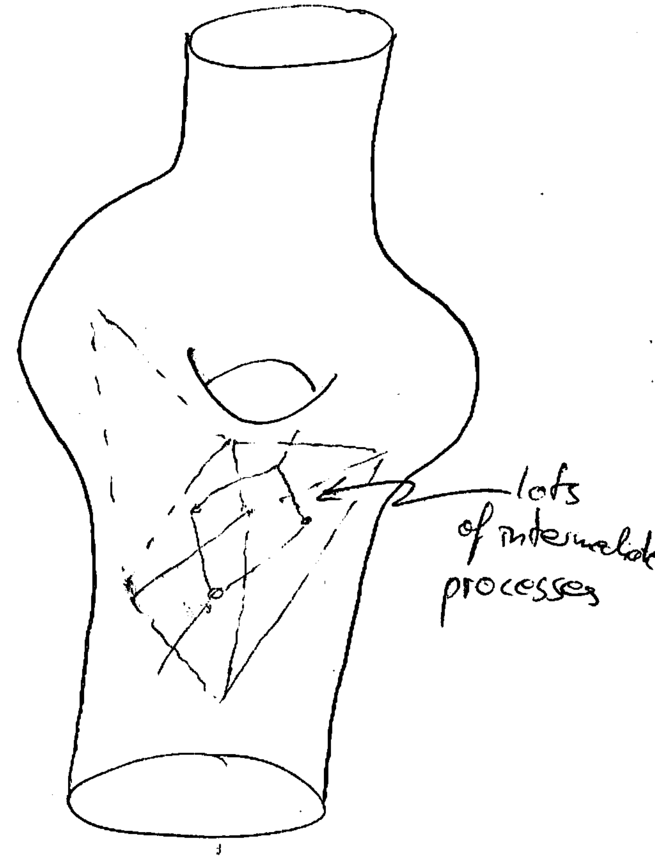
For conformal QFT this is still possible, but more subtle (FFRS 2006)

BUT what is easier is...

... understanding cobordisms from their "2-generators"



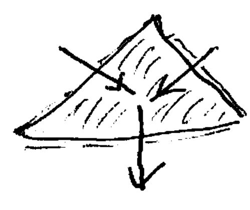
decomposition into generators



decomposition into "2-generators" ⇝ triangulation

Fact: every surface may be triangulated

⇝ sufficient to understand "globular" slices of cobordisms



so we need to think about

how to think about

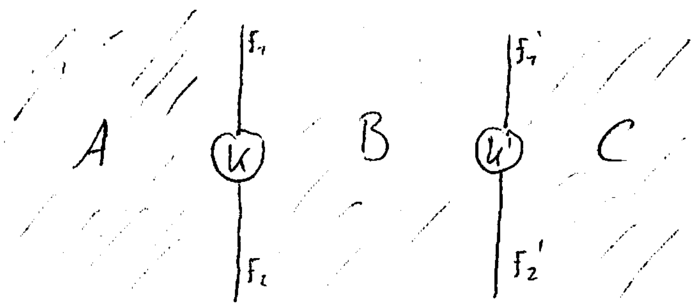
2-dimensional processes

the Frobenius algebras that we have seen last time will reappear – but now in the form of certain 2-processes called

special ambidextrous adjunctions

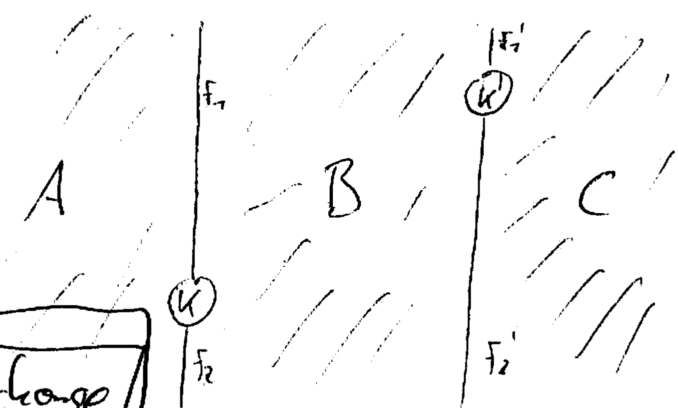
globular notation $\simeq$ string diagram notation

$$\begin{tikzcd} A \arrow[r, bend left=50, "f_1"] \arrow[r, bend right=50, "f_2"'] \arrow[r, phantom, "\Downarrow k"] & B \arrow[r, bend left=50, "f_1'"] \arrow[r, bend right=50, "f_2'"'] \arrow[r, phantom, "\Downarrow k'"] & C \end{tikzcd}$$



$$= \begin{tikzcd} A \arrow[r, bend left=50, "f_1"] & B \arrow[r, bend left=50, "f_1'"] \arrow[r, bend right=50, "f_2'"'] \arrow[r, phantom, "\Downarrow k'"] & C \\ A \arrow[r, bend left=50, "f_1"] \arrow[r, bend right=50, "f_2"'] \arrow[r, phantom, "\Downarrow k"] & B \arrow[r, bend right=50, "f_2'"'] & C \end{tikzcd}$$

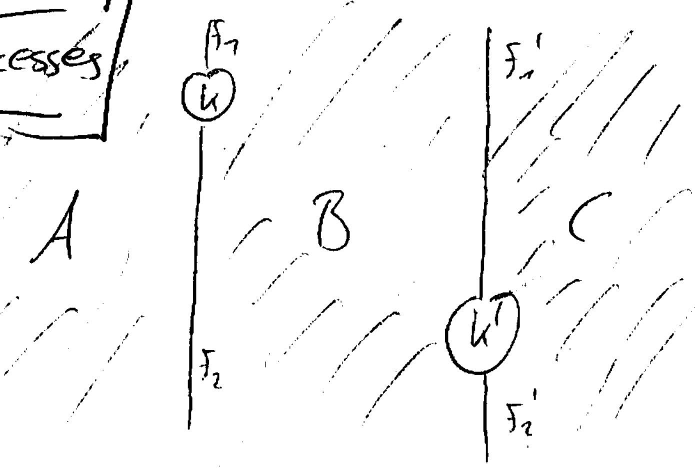
=



the exchange law of 2-processes

$$= \begin{tikzcd} A \arrow[r, bend left=50, "f_1"] \arrow[r, bend right=50, "f_2"'] \arrow[r, phantom, "\Downarrow k"] & B \arrow[r, bend left=50, "f_1'"] & C \\ A \arrow[r, bend right=50, "f_2"'] & B \arrow[r, bend left=50, "f_1'"] \arrow[r, bend right=50, "f_2'"'] \arrow[r, phantom, "\Downarrow k'"] & C \end{tikzcd}$$

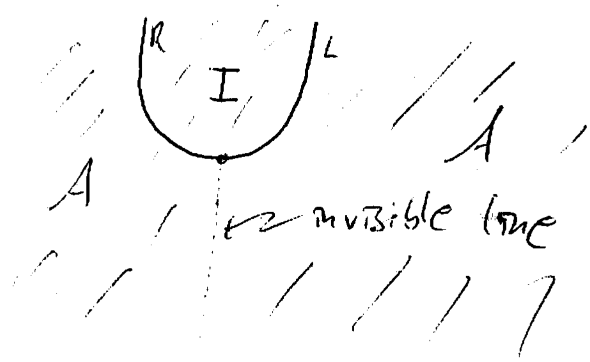
=



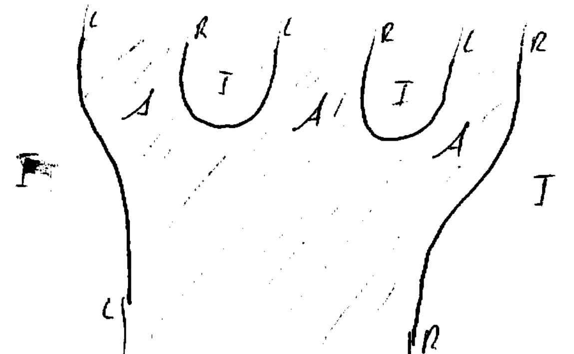
two ways to talk about "planar processes" a contraction ("on I")

defines an associative monoid ("on I")

$$\begin{tikzcd} A \arrow[r, "R"] \arrow[rr, bend right, "\mathrm{Id}"'] & I \arrow[r, "L"] \arrow[d, phantom, "\Downarrow"] & A \\ & {} & \end{tikzcd}$$

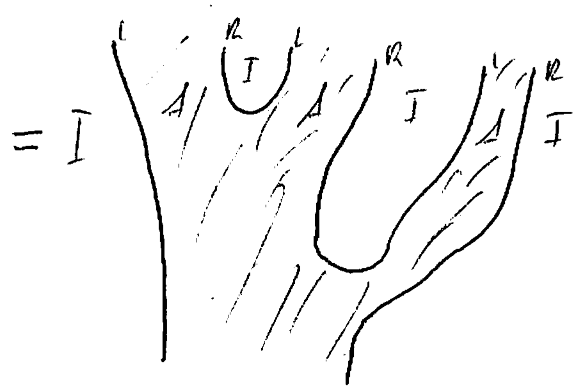


$$\begin{tikzcd} I \arrow[r, "L"] & A \arrow[r, "R"] \arrow[rr, bend right, "\mathrm{Id}"'] & I \arrow[r, "L"] & A \arrow[r, "R"] \arrow[rr, bend right, "\mathrm{Id}"'] & I \arrow[r, "L"] & A \arrow[r, "R"] & I \end{tikzcd}$$



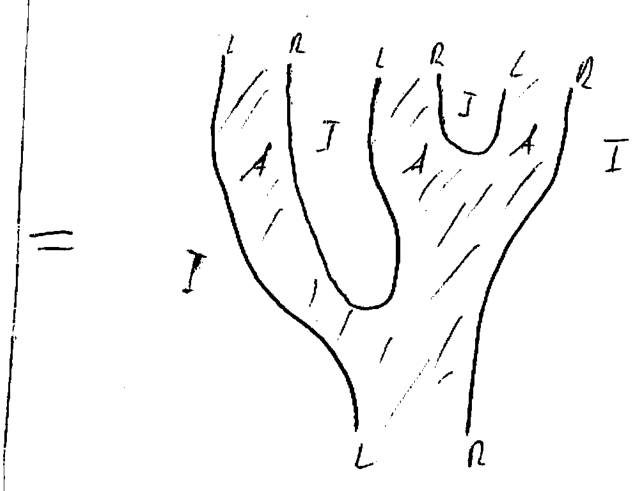
$$= \begin{tikzcd} I \arrow[r, "L"] & A \arrow[r, "R"] \arrow[rr, bend right, "\mathrm{Id}"'] & I \arrow[r, "L"] & A \arrow[r, "R"] & I \arrow[r, "L"] & A \arrow[r, "R"] & I \end{tikzcd}$$

$$\begin{tikzcd} I \arrow[r, "L"] & A \arrow[r, "\mathrm{Id}"] & A \arrow[r, "R"] \arrow[rr, bend right, "\mathrm{Id}"'] & I \arrow[r, "L"] & A \arrow[r, "R"] & I \end{tikzcd}$$



$$= \begin{tikzcd} I \arrow[r, "L"] & A \arrow[r, "R"] & I \arrow[r, "L"] & A \arrow[r, "R"] \arrow[rr, bend right, "\mathrm{Id}"'] & I \arrow[r, "L"] & A \arrow[r, "R"] & I \end{tikzcd}$$

$$\begin{tikzcd} I \arrow[r, "L"] & A \arrow[r, "R"] \arrow[rr, bend right, "\mathrm{Id}"'] & I \arrow[r, "L"] & A \arrow[r, "\mathrm{Id}"] & A \arrow[r, "R"] & I \end{tikzcd}$$



a <u>contraction</u>

$$\begin{tikzcd} A \arrow[r, "R"] \arrow[rr, bend right=60, "\mathrm{Id}"'] & I \arrow[d, phantom, "\Downarrow"] \arrow[r, "L"] & A \\ & {} & \end{tikzcd}$$

<u>with</u> <u>cocontraction</u>

$$\begin{tikzcd} & {} \arrow[d, phantom, "\Downarrow"] & \\ I \arrow[r, "L"'] \arrow[rr, bend left=60, "\mathrm{Id}"] & A \arrow[r, "R"'] & I \end{tikzcd}$$

satisfying the

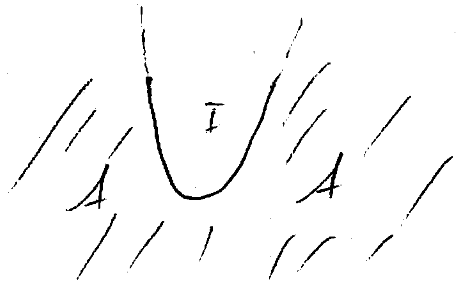
$\boxed{\text{zig-zag law}}$

$$\begin{tikzcd} A \arrow[r, "R"] \arrow[rr, bend right=60, "\mathrm{Id}"'] & I \arrow[r, "L"] \arrow[rr, bend left=60, "\mathrm{Id}"] & A \arrow[r, "R"] & I \end{tikzcd}$$

$$= A \xrightarrow{R} I$$

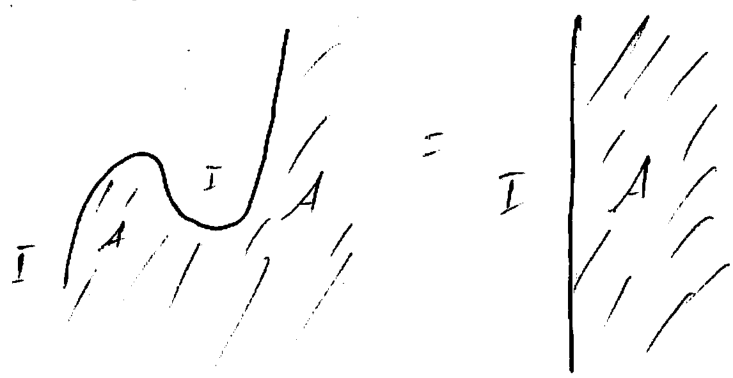
&

$$\begin{tikzcd} I \arrow[r, "L"] \arrow[rr, bend left=60, "\mathrm{Id}"] & A \arrow[r, "R"] \arrow[rr, bend right=60, "\mathrm{Id}"'] & I \arrow[r, "L"] & A \end{tikzcd}$$



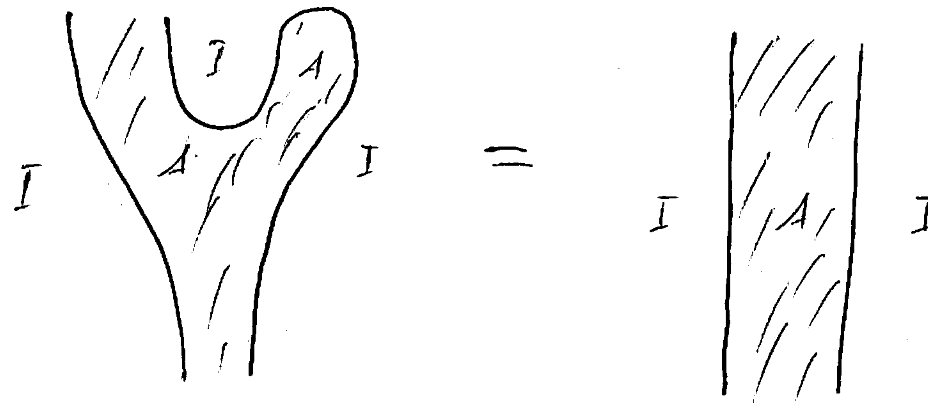
this is called a (left) <u>adjunction</u>

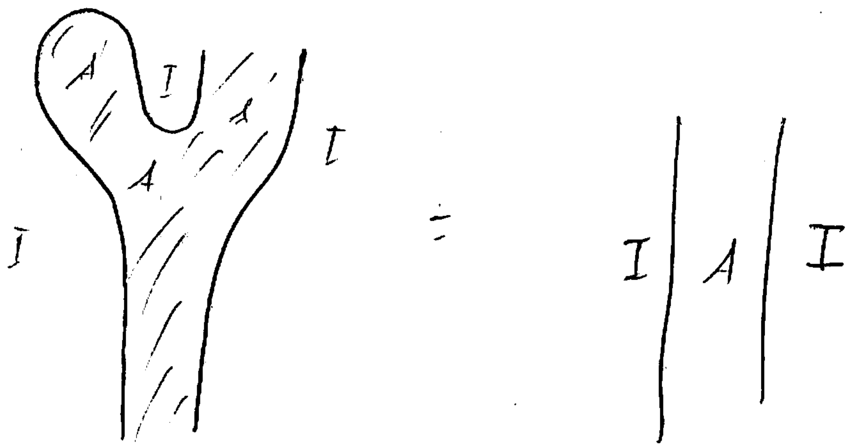
&



defines an...

... associative monoid with counit :



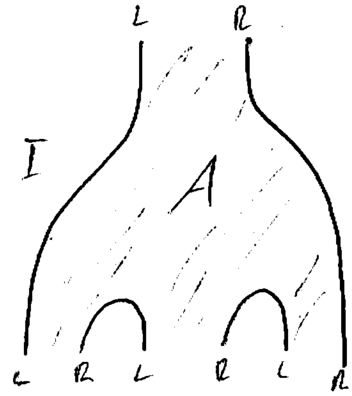


we can play the same game with A and I interchanged ...

a cocontraction on A defines a **co-associative co-monad** on I

$$\begin{tikzcd} A \arrow[r, "R"'] \arrow[rr, bend left, "\mathrm{Id}"] & I \arrow[u, phantom, "\Downarrow"] \arrow[r, "L"'] & A \end{tikzcd}$$

$$\begin{tikzcd} I \arrow[r, "L"] & A \arrow[r, "R"] \arrow[rr, bend left, "\mathrm{Id}"] & I \arrow[u, phantom, "\Downarrow"] \arrow[r, "L"] & A \arrow[r, "R"] \arrow[rr, bend left, "\mathrm{Id}"] & I \arrow[u, phantom, "\Downarrow"] \arrow[r, "L"] & A \arrow[r, "R"] & I \end{tikzcd}$$

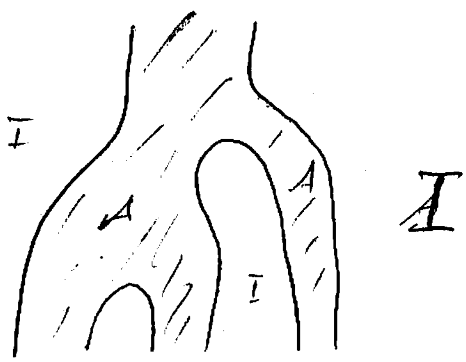


$$\begin{tikzcd} I \arrow[r, "L"] & A \arrow[rr, "\mathrm{Id}"] \arrow[rr, phantom, "\Downarrow I"'] & & A \arrow[r, "R"] \arrow[rr, bend left, "\mathrm{Id}"] & I \arrow[u, phantom, "\Downarrow"] \arrow[r, "L"] & A \arrow[r, "R"] & I \end{tikzcd}$$

$=$

$$\begin{tikzcd} I \arrow[r, "L"] & A \arrow[r, "R"] \arrow[rr, bend left, "\mathrm{Id}"] & I \arrow[u, phantom, "\Downarrow"] \arrow[r, "L"] & A \arrow[r, "R"] & I \arrow[r, "L"] & A \arrow[r, "R"] & I \end{tikzcd}$$

$=$

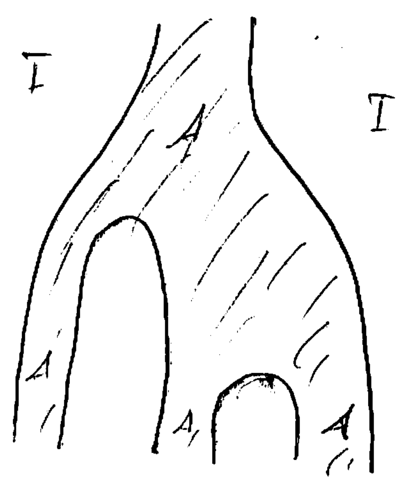


$$\begin{tikzcd} I \arrow[r, "L"] & A \arrow[r, "L"] \arrow[rr, bend left, "\mathrm{Id}"] & I \arrow[u, phantom, "\Downarrow"] \arrow[r, "R"] & A \arrow[r, "\mathrm{Id}"] & A \arrow[r, "R"] & I \end{tikzcd}$$

$=$

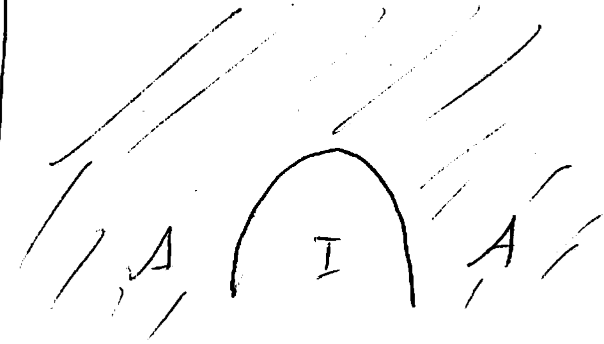
$$\begin{tikzcd} I \arrow[r, "L"] & A \arrow[r, "R"] & I \arrow[r, "L"] & A \arrow[r, "R"] \arrow[rr, bend left, "\mathrm{Id}"] & I \arrow[u, phantom, "\Downarrow"] \arrow[r, "L"] & A \arrow[r, "R"] & I \end{tikzcd}$$

$=$



a co-contraction on A

$$\begin{tikzcd} A \arrow[r, "R"'] \arrow[rr, bend left, "Id"] & I \arrow[r, "L"'] \arrow[u, phantom, "\Downarrow"] & A \end{tikzcd}$$



with contraction

$$\begin{tikzcd} I \arrow[r, "L"] \arrow[rr, bend right, "Id"'] & A \arrow[r, "R"] \arrow[d, phantom, "\Downarrow"] & I \\ & {} & \end{tikzcd}$$

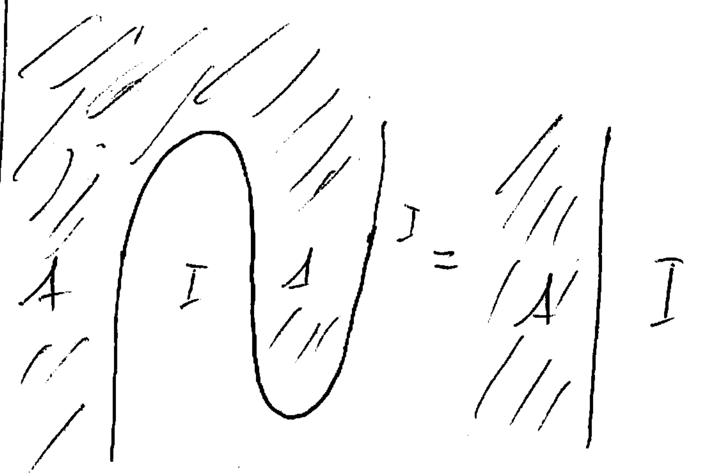
this is called a (right) <u>adjunction</u>

satisfying the

$\boxed{\text{Zig-zag-law}}$

$$\begin{tikzcd} A \arrow[r, "R"] \arrow[rr, bend left, "Id"] & I \arrow[r, "L"] \arrow[rr, bend right, "Id"'] & A \arrow[r, "R"] & I \end{tikzcd}$$

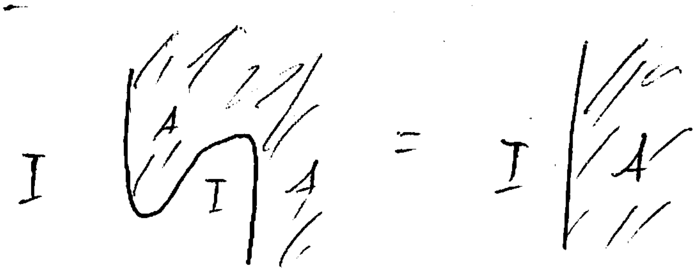
$= A \xrightarrow{R} I$



&

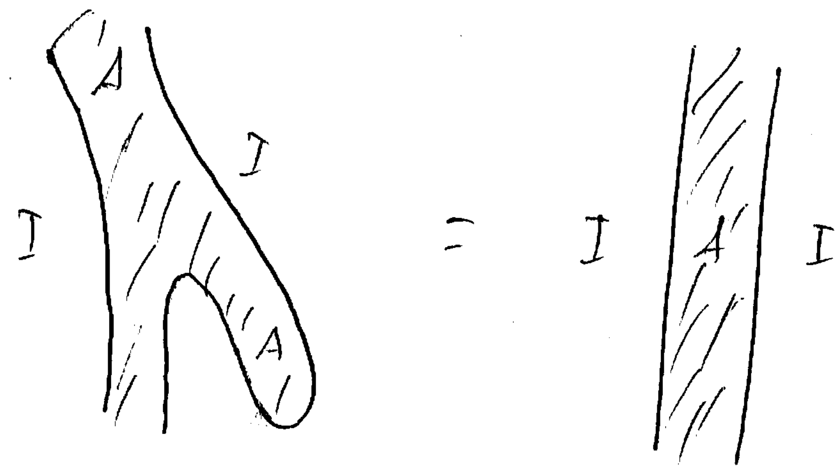
$$\begin{tikzcd} I \arrow[r] \arrow[rr, bend right, "Id"'] & A \arrow[r] \arrow[rr, bend left, "Id"] & I \arrow[r] & A \end{tikzcd}$$

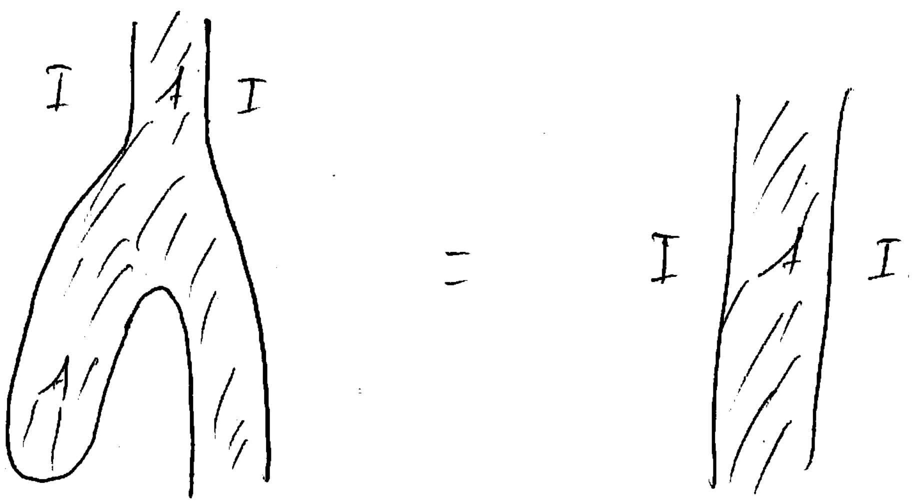
$= I \xrightarrow{L} A$



defines a ...

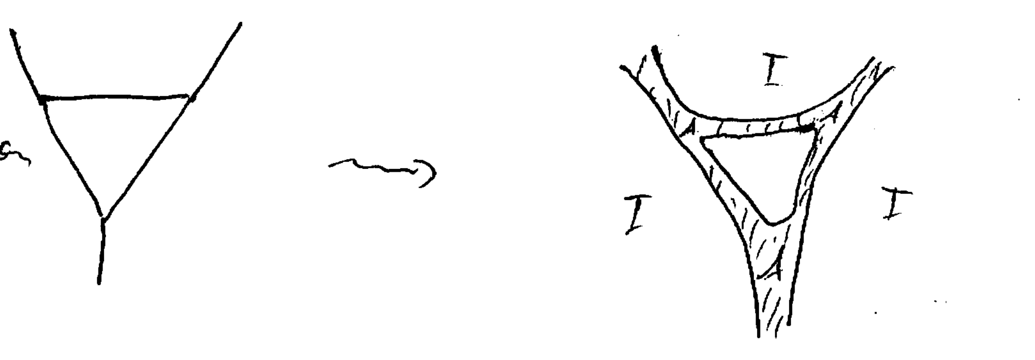
... co-associative comonoid
<u>with</u> <u>counit</u>



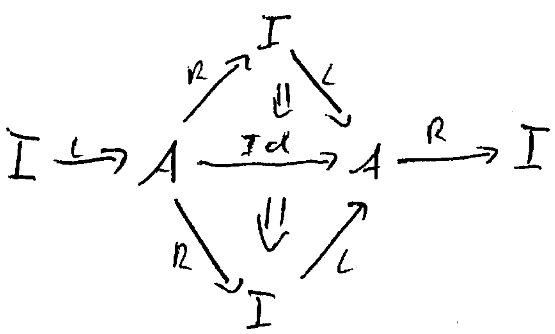


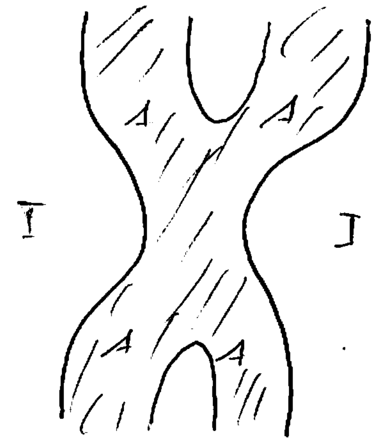
in our application we
want **both** unital monoid
+ counital comonoid

triangulation

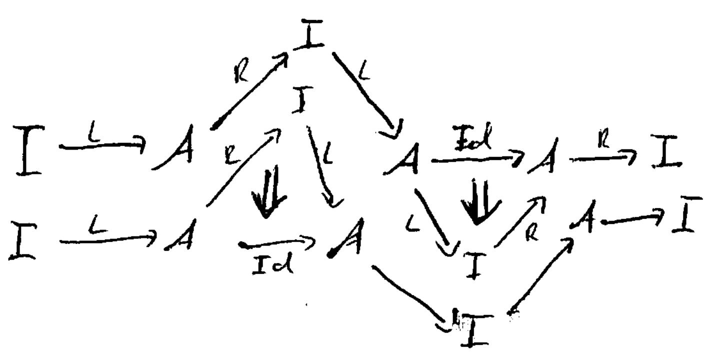


Fact: The monad structure coming from a left-and-right adjunction ("ambidextrous") is automatically Frobenius

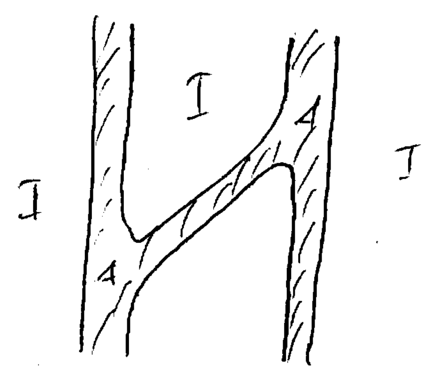




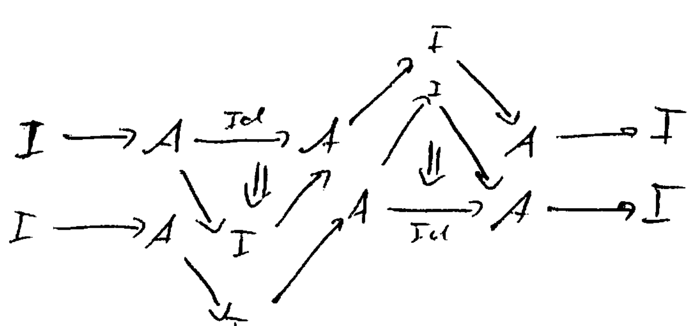
=



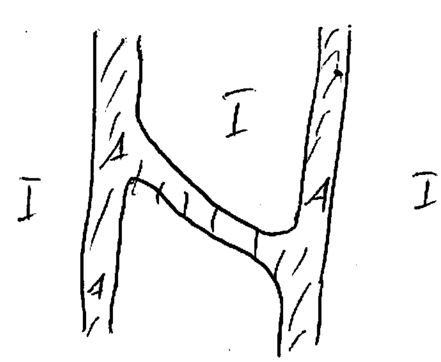
=



=



=



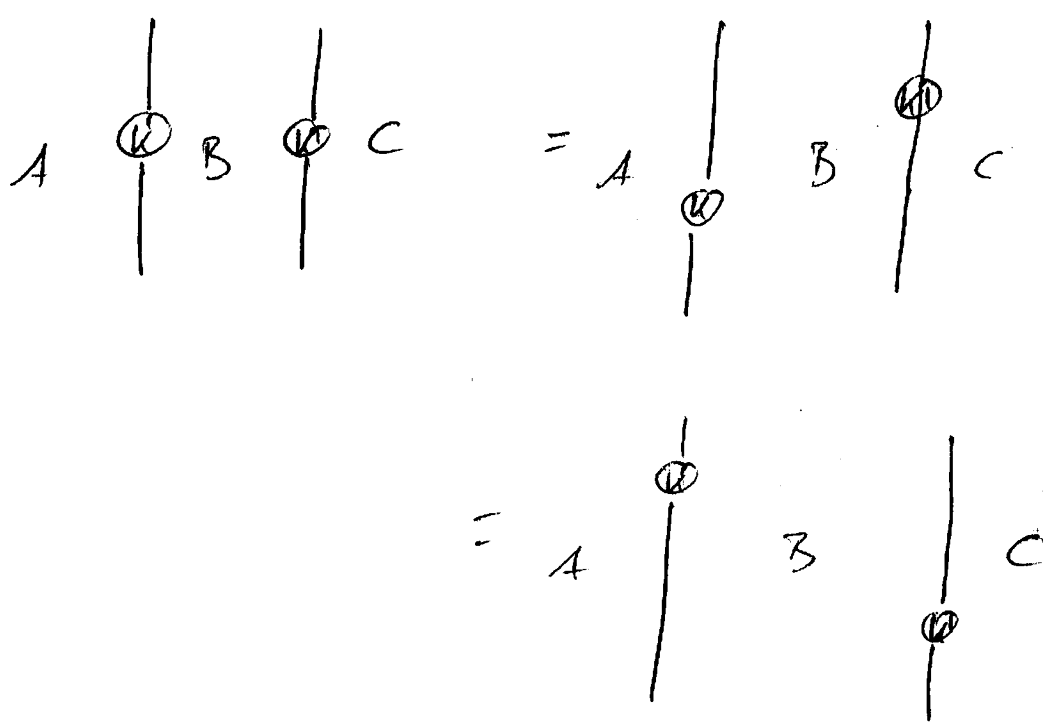
Moral: Frobenius property

is an incarnation of

the exchange law

for 2-processes!

recall: exchange law was



for our purpose, we need <u>one extra condition</u>

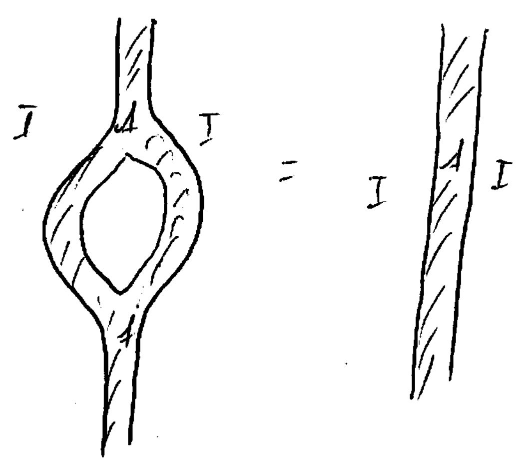
"pretending to start a 2-process but then not doing anything is the same as doing nothing"

$$\boxed{\begin{tikzcd} A \arrow[r, "R"] \arrow[rr, bend left=60, "Id"] \arrow[rr, bend right=60, "Id"'] & I \arrow[r, "L"] \arrow[u, phantom, "\Downarrow"] \arrow[d, phantom, "\Downarrow"] & A \end{tikzcd} \quad = \quad A \xrightarrow{Id} A}$$

this implies the <u>bubble move</u>

$$\begin{tikzcd} I \arrow[r, "L"] & A \arrow[r, "R"] \arrow[rr, bend left=60, "Id"] \arrow[rr, bend right=60, "Id"'] & I \arrow[r, "L"] \arrow[u, phantom, "\Downarrow"] \arrow[d, phantom, "\Downarrow"] & A \arrow[r, "R"] & I \end{tikzcd}$$

$$= \quad I \xrightarrow{L} A \xrightarrow{R} I$$



an ambidextrous adjunction with

this "bubble move property" is

a "special ambidextrous adjunction"

since the monad defined by it is

b) a "special Frobenius monad"

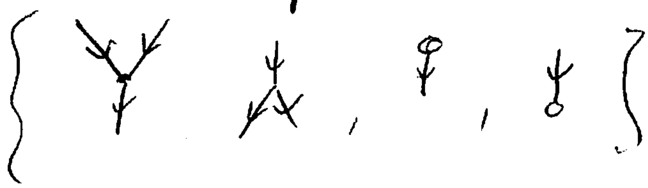
a) (= Frob. monad with bubble move property)

bad terminology!

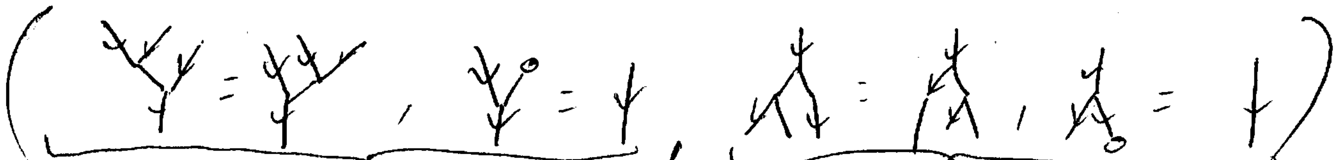
but a) is established

bad terminology...

recall: special Frob. monad is this:

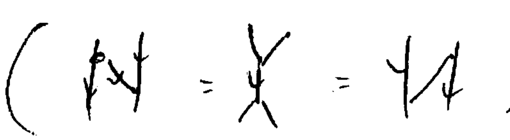


such that:

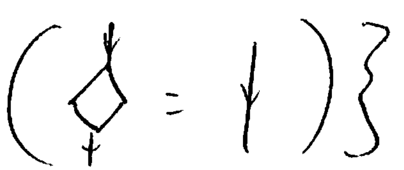


unital assoc.

counital coass.



Frobenius



"special"

aside (<u>don't read this</u>):

we started with

"QFT = 1-functor = 1-rep of cobord. 1-catg."

QFT() $= A \xrightarrow{\cdots} A$

we are heading towards something like

"extended QFT = 2-functor = 2-rep on ext. cob."

$$eQFT\left(\begin{tikzcd} x \arrow[r, bend left, "\gamma"] \arrow[r, bend right, "\gamma'"'] \arrow[r, phantom, "\Downarrow\Sigma"] & y \end{tikzcd}\right) = \begin{tikzcd} A \arrow[r, bend left, "\cdots"] \arrow[r, bend right, "\cdots"'] \arrow[r, phantom, "\Downarrow\cdots"] & B \end{tikzcd}$$

piece of cobordism

<u>Fact:</u> <u>special ambijunction is what allows one 2-functor to be expressed in terms of the other 2-functor</u>

$$\mathrm{tree}_1 \xrightarrow{L} \mathrm{tree}_2 \xrightarrow{R} \mathrm{tree}_1$$ special ambijunction

implies that the equation for L

$$\begin{tikzcd}
\mathrm{tree}_1(x) \arrow[r, bend left=50, "\mathrm{tree}_1(\gamma)"] \arrow[r, "\mathrm{tree}_1(\gamma')"'] \arrow[r, phantom, bend left=25, "\Downarrow \mathrm{tree}_1(\Sigma)"] \arrow[d, "L(x)"'] \arrow[dr, phantom, "\swarrow L(\gamma')"] & \mathrm{tree}_1(y) \arrow[d, "L(y)"] \\
\mathrm{tree}_2(x) \arrow[r, "\mathrm{tree}_2(\gamma')"'] & \mathrm{tree}_2(y)
\end{tikzcd}
=
\begin{tikzcd}
\mathrm{tree}_1(x) \arrow[r, "\mathrm{tree}_1(\gamma)"] \arrow[d, "L(x)"'] \arrow[dr, phantom, "\swarrow L(\gamma)"] & \mathrm{tree}_1(y) \arrow[d, "L(y)"] \\
\mathrm{tree}_2(x) \arrow[r, "\mathrm{tree}_2(\gamma)"] \arrow[r, bend right=50, "\mathrm{tree}_2(\gamma')"'] \arrow[r, phantom, bend right=25, "\Downarrow \mathrm{tree}_2(\Sigma)"] & \mathrm{tree}_2(y)
\end{tikzcd}$$

may be solved for $\mathrm{tree}_1(\Sigma)$:

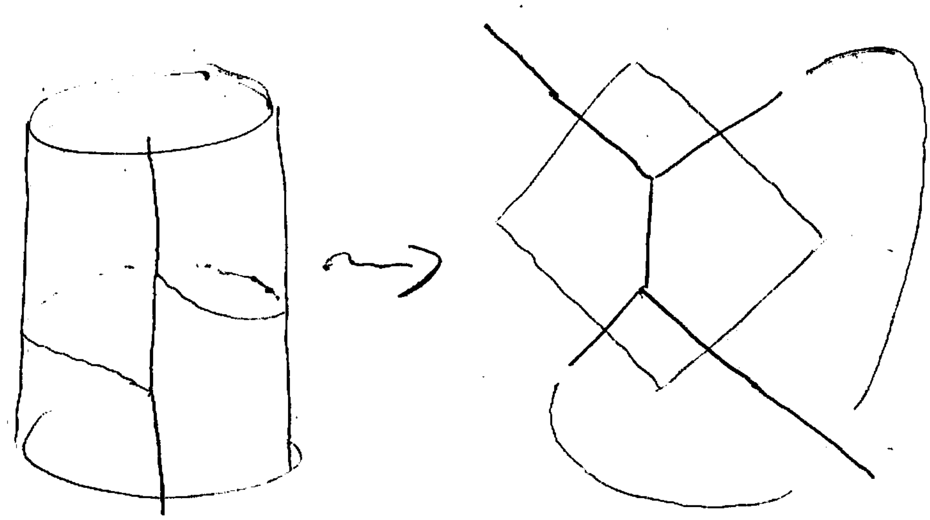
$\Rightarrow$

$$\begin{tikzcd}
\mathrm{tree}_1(x) \arrow[r, bend left=50, "\mathrm{tree}_1(\gamma)"] \arrow[r, bend right=50, "\mathrm{tree}_1(\gamma')"'] \arrow[r, phantom, "\Downarrow \mathrm{tree}_1(\Sigma)"] & \mathrm{tree}_1(y)
\end{tikzcd}
=
\begin{tikzcd}
\mathrm{tree}_1(x) \arrow[r, "\mathrm{tree}_1(\gamma)"] \arrow[d, "L(x)"'] \arrow[dr, phantom, "\swarrow L(\gamma)"] \arrow[dd, bend right=60, "\mathrm{Id}"'] & \mathrm{tree}_1(y) \arrow[d, "L(y)"] \arrow[dd, bend left=60, "\mathrm{Id}"] \\
\mathrm{tree}_2(x) \arrow[r, bend left=50, "\mathrm{tree}_2(\gamma)"] \arrow[r, bend right=50, "\mathrm{tree}_2(\gamma')"'] \arrow[r, phantom, "\Downarrow \mathrm{tree}_2(\Sigma)"] \arrow[d, "R(x)"'] \arrow[dr, phantom, "\swarrow R(\gamma')"] & \mathrm{tree}_2(y) \arrow[d, "R(y)"] \\
\mathrm{tree}_1(x) \arrow[r, "\mathrm{tree}_1(\gamma')"'] & \mathrm{tree}_1(y)
\end{tikzcd}$$

$\underline{\underline{\mathrm{tree}_1}}$ expressed in terms of $\underline{\underline{\mathrm{tree}_2}}$ + "transition data"

Third part

- on which we glue what we had sliced



- and finally see the need for

Frobenius algebras internal to ribbon categories