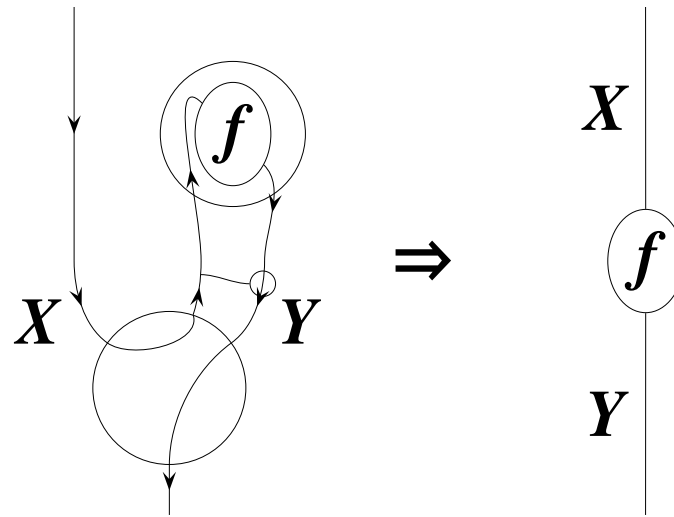


# Computation and the Periodic Table

John C. Baez

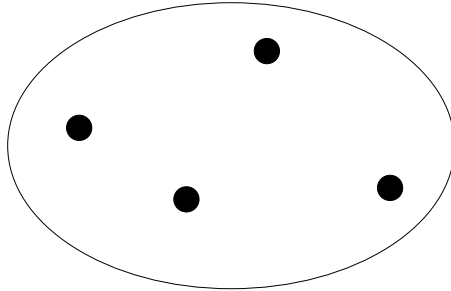
LICS 2009



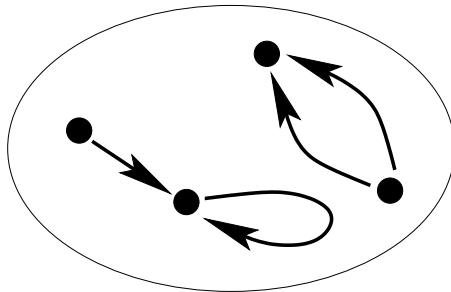
$$((\lambda x : X . f) a) \quad \Rightarrow \quad f[a/x]$$

for references and more, see  
<http://math.ucr.edu/home/baez/lics2009/>

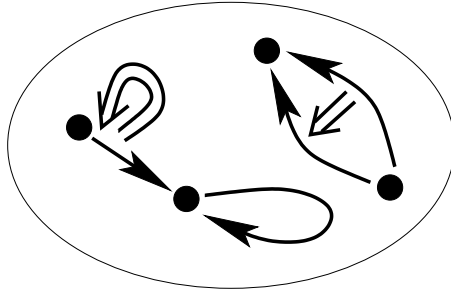
**Once upon a time, mathematics was all about *sets*:**



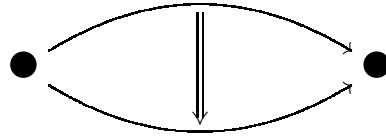
**In 1945, Eilenberg and Mac Lane introduced *categories*:**



**In 1967 Bénabou introduced fully general *2-categories*:**



**A 2-category also has ‘2-morphisms’ between morphisms:**



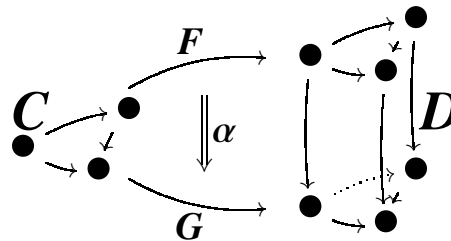
**These describe ‘processes between processes’.**

The ‘set of all sets’ is really a category: **Set**. This has:

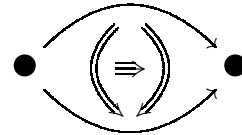
- sets as objects,
- functions as morphisms.

The ‘category of all categories’ is really a 2-category: **Cat**. This has:

- categories as objects,
- functors as morphisms,
- natural transformations as 2-morphisms.



**In 1995, Gordon, Power and Street introduced fully general *3-categories*:**



**For example, there is a 3-category  $2\text{Cat}$ !**

**Now people are studying *n-categories* and even  $\infty$ -categories. This is starting to have a big impact on topology and physics.**

**How about logic and computation?**

**I claim *yes*.**

For example, Lambek showed that any theory in the simply typed  $\lambda$ -calculus gives a cartesian closed category where:

- objects are *types*  $X, Y, Z, \dots$
- morphisms  $f: X \rightarrow Y$  are equivalence classes of *terms* of type  $Y$  with free variable of type  $X$ .

Two terms give the same morphism if they differ by certain *rewrite rules*, which include  $\beta$ -reduction:

$$((\lambda x:X . f) a) \Rightarrow f[a/x]$$

and  $\eta$ -reduction:

$$(\lambda x:X . (fx)) \Rightarrow f$$

**Alas, identifying terms that differ by rewrite rules *ignores the actual process of computation!* All these terms give the same morphism:**

$$\begin{aligned}
& ((\lambda f.(\lambda x.(f(fx))))(\lambda g.(\lambda y.(g(gy))))(\lambda z.z + 1))0 \\
\Rightarrow & (\lambda x.((\lambda g.(\lambda y.(g(gy))))((\lambda g.(\lambda y.(g(gy))))x)(\lambda z.z + 1))0) \\
\Rightarrow & (\lambda x.((\lambda g.(\lambda y.(g(gy))))(\lambda y.(x(xy)))(\lambda z.z + 1))0) \\
\Rightarrow & (\lambda x.((\lambda g.(\lambda y.(g(gy))))(\lambda z.(x(xz)))(\lambda z.z + 1))0) \\
\Rightarrow & (\lambda x.(\lambda y.((\lambda z.(x(xz)))(\lambda z.(x(xz)))y)))(\lambda z.z + 1))0) \\
\Rightarrow & ((\lambda x.(\lambda y.((\lambda z.(x(xz)))(x(xy)))))(\lambda z.z + 1))0) \\
\Rightarrow & ((\lambda x.(\lambda y.(x(x(x(xy)))))(\lambda z.z + 1))0) \\
\Rightarrow & ((\lambda y.(\lambda z.z + 1)((\lambda z.z + 1)((\lambda z.z + 1)((\lambda z.z + 1))))0) \\
\Rightarrow & ((\lambda y.((\lambda z.z + 1)((\lambda z.z + 1)((\lambda z.z + 1)(y + 1))))0) \\
\Rightarrow & ((\lambda y.((\lambda z.z + 1)((\lambda z.z + 1)(y + 2))))0) \\
\Rightarrow & ((\lambda y.((\lambda z.z + 1)(y + 3)))0) \\
\Rightarrow & ((\lambda y.(y + 4))0) \\
\Rightarrow & (0 + 4) \\
\Rightarrow & 4
\end{aligned}$$

To keep track of the process of computation, we can use a 2-category where:

- objects are *types*  $X, Y, Z, \dots$
- morphisms  $f: X \rightarrow Y$  are *terms* of type  $Y$  with free variable of type  $X$ .
- 2-morphisms  $\alpha: f \Rightarrow g$  are equivalence classes of *sequences of rewrites* going from  $f$  to  $g$ .

Any theory in the simply-typed  $\lambda$ -calculus gives a ‘cartesian closed 2-category’ this way. This has already been studied by R. A. G. Seely and Barnaby P. Hilken. But their work is just beginning to be absorbed.

**I would like to explain how this passage from *cartesian closed categories* to *cartesian closed 2-categories* is mathematically analogous to the passage from *particle theory* to *string theory*!**

**To understand this, we should start with the ‘Periodic Table’ of *n*-categories.**

The idea is that a  $(n + k)$ -category with only one  $j$ -morphism for  $j < k$  can be reinterpreted as an  $n$ -category:

$$\begin{array}{ccc}
 (n + k)\text{-morphisms} & \rightsquigarrow & n\text{-morphisms} \\
 (n + k - 1)\text{-morphisms} & \rightsquigarrow & (n - 1)\text{-morphisms} \\
 \vdots & & \vdots \\
 \vdots & & \vdots \\
 k\text{-morphisms} & \rightsquigarrow & \text{objects} \\
 \text{one } (k - 1)\text{-morphism} & \rightsquigarrow & \bullet \\
 \vdots & & \vdots \\
 \vdots & & \vdots \\
 \text{one morphism} & \rightsquigarrow & \bullet \\
 \text{one object} & \rightsquigarrow & \bullet
 \end{array}$$

We get a special sort of  $n$ -category: a  $k$ -tuply monoidal  $n$ -category.

### *k*-tuply monoidal *n*-categories

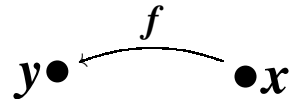
	<i>n</i> = 0	<i>n</i> = 1	<i>n</i> = 2
<i>k</i> = 0	sets	categories	2-categories
<i>k</i> = 1	monoids	monoidal categories	monoidal 2-categories
<i>k</i> = 2	commutative monoids	braided monoidal categories	braided monoidal 2-categories
<i>k</i> = 3	“	symmetric monoidal categories	syllaptic monoidal 2-categories
<i>k</i> = 4	“	“	symmetric monoidal 2-categories
<i>k</i> = 5	“	“	“

Let's try to understand the Periodic Table.

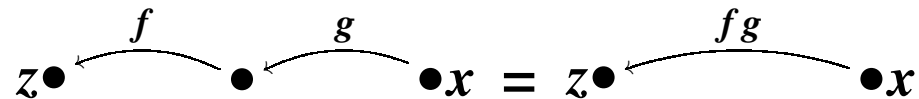
A *category* has objects:

$x \bullet$

and morphisms between objects:

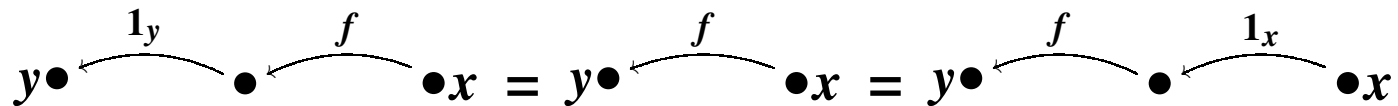


which we can compose:



in an associative way.

It also has an identity morphism for every object:



A 2-category also has 2-morphisms, which we can compose vertically:

$$\begin{array}{c}
 \begin{array}{ccc}
 y \bullet & & \bullet x \\
 \begin{array}{c} \curvearrowright \\ f' \\ \Downarrow \alpha \\ \Downarrow \alpha' \\ \curvearrowleft \\ f'' \end{array} & & \\
 \end{array} & = & \begin{array}{ccc}
 y \bullet & & \bullet x \\
 \begin{array}{c} \curvearrowright \\ f \\ \Downarrow \alpha\alpha' \\ \curvearrowleft \\ f'' \end{array} & & \\
 \end{array}
 \end{array}$$

or horizontally:

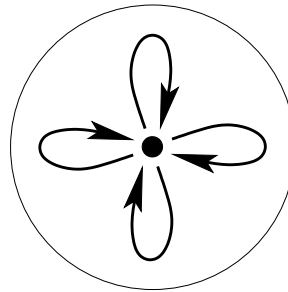
$$\begin{array}{c}
 \begin{array}{ccccc}
 z \bullet & & \bullet & & \bullet x \\
 \begin{array}{c} \curvearrowright \\ f \\ \Downarrow \alpha \\ \curvearrowleft \\ f' \end{array} & & & & \begin{array}{c} \curvearrowright \\ g \\ \Downarrow \beta \\ \curvearrowleft \\ g' \end{array} \\
 \end{array} & = & \begin{array}{ccc}
 z \bullet & & \bullet x \\
 \begin{array}{c} \curvearrowright \\ fg \\ \Downarrow \alpha\otimes\beta \\ \curvearrowleft \\ f'g' \end{array} & & \\
 \end{array}
 \end{array}$$

Various laws hold, notably the ‘interchange’ law:

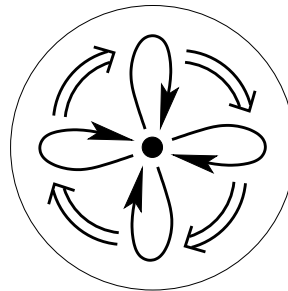
$$\begin{array}{ccc}
 \bullet & & \bullet \\
 \begin{array}{c} \curvearrowright \\ f'' \\ \Downarrow \alpha \\ \Downarrow \alpha' \\ \curvearrowleft \\ f'' \end{array} & & \begin{array}{c} \curvearrowright \\ g'' \\ \Downarrow \beta \\ \Downarrow \beta' \\ \curvearrowleft \\ g'' \end{array} \\
 \bullet & & \bullet
 \end{array}$$

$$(\alpha\alpha') \otimes (\beta\beta') = (\alpha \otimes \beta)(\alpha' \otimes \beta')$$

**A category with one object is a *monoid* — a set with associative multiplication and a unit element:**



**A 2-category with one object is a *monoidal category* — a category with an associative ‘tensor product’ and a unit object:**



**To regard a 2-category with one object as a monoidal category:**

- **we ignore the one object,**
- **we rename the morphisms ‘objects’.**
- **we rename the 2-morphisms ‘morphisms’.**

**Composition of 1-morphisms becomes ‘tensoring objects’.**

**Vertical composition of 2-morphisms becomes ‘composing morphisms’.**

**Horizontal composition of 2-morphisms becomes ‘tensoring morphisms’.**

More generally, an  $n$ -category with just one object is a *monoidal  $(n - 1)$ -category*.

For example:

- **FinVect** is a monoidal category with the usual tensor product  $V \otimes W$  of finite-dimensional vector spaces.
- **Set** is a monoidal category, using the cartesian product  $S \times T$  of sets.
- **Cat** is a monoidal 2-category, using the cartesian product  $C \times D$  of categories.
- We expect that  $n\text{Cat}$  is a monoidal  $(n + 1)$ -category!

**QUESTION:** what's a *monoidal category* with just one object?  
It must be some sort of monoid...

It has one object, namely the unit  $I$ , and a set of morphisms  $\alpha: I \rightarrow I$ . We can compose morphisms:

$$\alpha\beta$$

and also tensor them:

$$\alpha \otimes \beta$$

Composition and tensoring are related by the interchange law:

$$(\alpha\alpha') \otimes (\beta\beta') = (\alpha \otimes \beta)(\alpha' \otimes \beta')$$

So, we can carry out the ‘Eckmann–Hilton argument’:

$$\begin{array}{|c|c|} \hline \alpha & \beta \\ \hline \end{array}
 \quad
 \begin{array}{|c|c|} \hline \alpha & 1 \\ \hline 1 & \beta \\ \hline \end{array}
 \quad
 \begin{array}{|c|} \hline \alpha \\ \hline \beta \\ \hline \end{array}
 \quad
 \begin{array}{|c|c|} \hline 1 & \alpha \\ \hline \beta & 1 \\ \hline \end{array}
 \quad
 \begin{array}{|c|c|} \hline \beta & \alpha \\ \hline \end{array}$$

$$\begin{aligned}
 \alpha \otimes \beta &= (\alpha \otimes 1)(1 \otimes \beta) & (1\beta) \otimes (\alpha 1) &= \beta \otimes \alpha \\
 &\parallel & &\parallel \\
 &(\alpha 1) \otimes (1\beta) &= \alpha\beta &= (1 \otimes \alpha)(\beta \otimes 1)
 \end{aligned}$$

**ANSWER:** a monoidal category with one object is a *commutative monoid!*

**In other words:** a 2-category with one object and one morphism is a commutative monoid.

**What's the pattern?**

**An  $(n + k)$ -category with only one  $j$ -morphism for  $j < k$  can be reinterpreted as an  $n$ -category.**

**But, it will be an  $n$ -category with  $k$  ways to 'multiply': a  *$k$ -tuply monoidal  $n$ -category*.**

**When there are several ways to multiply, the Eckmann–Hilton argument gives a kind of 'commutativity'.**

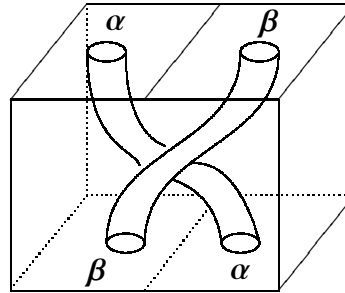
**Our guesses are shown in the Periodic Table...**

### *k*-tuply monoidal *n*-categories

	<i>n</i> = 0	<i>n</i> = 1	<i>n</i> = 2
<i>k</i> = 0	sets	categories	2-categories
<i>k</i> = 1	monoids	monoidal categories	monoidal 2-categories
<i>k</i> = 2	commutative monoids	braided monoidal categories	braided monoidal 2-categories
<i>k</i> = 3	“	symmetric monoidal categories	syllaptic monoidal 2-categories
<i>k</i> = 4	“	“	symmetric monoidal 2-categories
<i>k</i> = 5	“	“	“

Take  $n = 1, k = 2$ . A doubly monoidal 1-category is a *braided monoidal category*:

$$\begin{array}{|c|c|} \hline \alpha & \beta \\ \hline \end{array} \cong \begin{array}{|c|c|} \hline \alpha & 1 \\ \hline 1 & \beta \\ \hline \end{array} \cong \begin{array}{|c|} \hline \alpha \\ \hline \beta \\ \hline \end{array} \cong \begin{array}{|c|c|} \hline 1 & \alpha \\ \hline \beta & 1 \\ \hline \end{array} \cong \begin{array}{|c|c|} \hline \beta & \alpha \\ \hline \end{array}$$

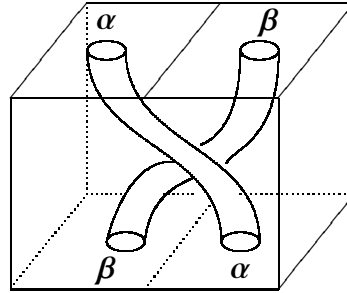


$$B_{\alpha,\beta}: \alpha \otimes \beta \xrightarrow{\sim} \beta \otimes \alpha$$

The *process of proving an equation* has become an *isomorphism!*

This happens when we move a step right in the Periodic Table.

Indeed, a *different proof* of commutativity becomes a *different isomorphism*:



$$B_{\beta,\alpha}^{-1} : \alpha \otimes \beta \xrightarrow{\sim} \beta \otimes \alpha$$

In physics, we visualize these tubes as 1-dimensional paths of particles in (1 + 2)-dimensional spacetime.

In general, we can use  $k$ -tuply monoidal  $n$ -categories to describe  $n$ -dimensional surfaces in  $(n + k)$ -dimensional spacetime. Here  $n = 1, k = 2$ .

**A triply monoidal 1-category is a *symmetric monoidal category*.  
Now the two ways of moving  $\alpha$  past  $\beta$  become equal:**

The diagram consists of two identical diagrams separated by an equals sign. Each diagram shows two curved lines representing paths. The left diagram has a line labeled  $\alpha$  on the left and a line labeled  $\beta$  on the right. The  $\alpha$  line starts at the top left and curves down and right, crossing over the  $\beta$  line. The  $\beta$  line starts at the top right and curves down and left, crossing under the  $\alpha$  line. The right diagram is identical, but the  $\alpha$  line crosses under the  $\beta$  line.

**Now we can visualize these curves as 1-dimensional paths of particles in (1 + 3)-dimensional spacetime.**

**Symmetric monoidal categories are ‘more commutative’ than braided ones. This happens when we move one step down in the Periodic Table.**

**However,  $k$ -tuply monoidal  $n$ -categories seem to become ‘maximally commutative’ when  $k$  reaches  $n + 2$ .**

**For example, you can untie all  $n$ -dimensional knots in a  $(2n + 2)$ -dimensional cube. Extra dimensions don’t help!**

***The Stabilization Hypothesis:  $k$ -tuply monoidal  $n$ -categories are equivalent to  $(k + 1)$ -tuply monoidal  $n$ -categories when  $k \geq n + 2$ .***

**I want to lead you down the  $n = 1$  column:**

- **categories**
- **monoidal categories**
- **braided monoidal categories**
- **symmetric monoidal categories**

**and show how these arise in particle physics and — especially in the symmetric case — also computation and logic.**

**Then I want to show how going to  $n = 2$  takes us from *particles* to *strings* — and what *this* corresponds to in computation and logic.**

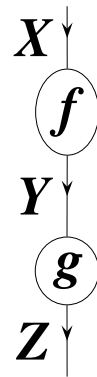
In a 'Feynman diagram', physicists draw a morphism

$$f: X \rightarrow Y$$

like this:



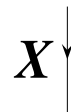
Composition looks like this:



**The associative law is then implicit:**



**If we draw  $1_X: X \rightarrow X$  like this:**



**the unit laws are implicit too.**

For theories with at least 1 dimension of space, physicists use *monoidal* categories.

Here any pair of morphisms  $f: X \rightarrow Y$ ,  $f': X' \rightarrow Y'$  has a tensor product

$$f \otimes f': X \otimes X' \rightarrow Y \otimes Y'$$

We use this to describe parallel processes:

$$\begin{array}{ccc}
 \begin{array}{c} X \downarrow \\ \circlearrowleft f \\ Y \downarrow \end{array} & \begin{array}{c} X' \downarrow \\ \circlearrowleft f' \\ Y' \downarrow \end{array} & = & \begin{array}{c} X \otimes X' \downarrow \\ \circlearrowleft f \otimes f' \\ Y \otimes Y' \downarrow \end{array}
 \end{array}$$

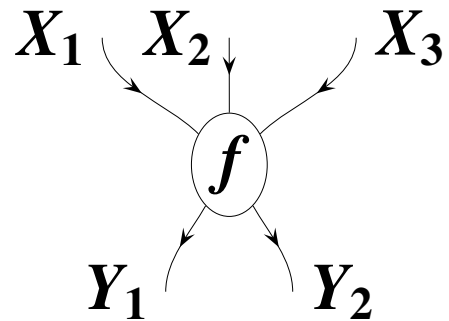
**Examples:**

- The category **FinVect**, with its usual tensor product  $\otimes$ .
- The category **Set**, with the cartesian product  $\times$ .
- In categories of propositions, where  $X \vdash Y$  is a morphism from  $X$  to  $Y$ ,  $X \otimes Y$  means ‘ $X$  and  $Y$ ’.

**More generally, we can draw any morphism**

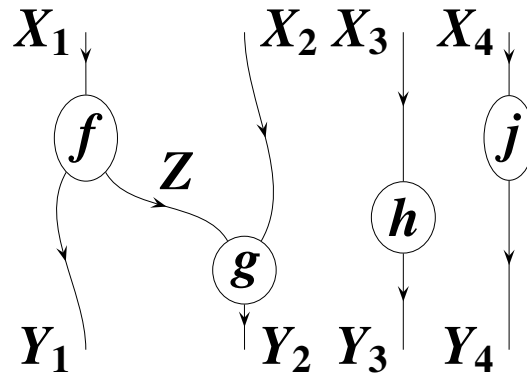
$$f: X_1 \otimes \cdots \otimes X_n \rightarrow Y_1 \otimes \cdots \otimes Y_m$$

**like this:**

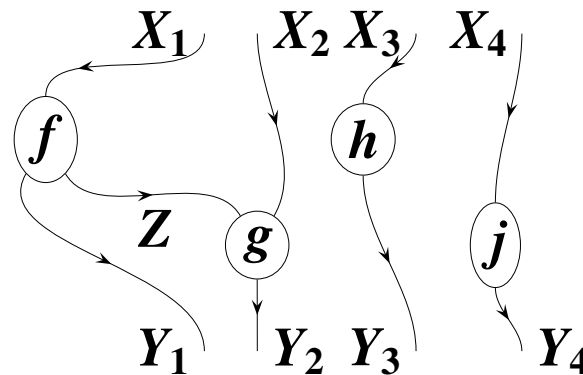


**In physics we use this to depict an interaction of particles.**

By composing and tensoring morphisms, we can build up more interesting Feynman diagrams:



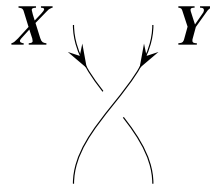
Thanks to the laws governing a monoidal category, we can deform the picture without changing the morphism it describes:



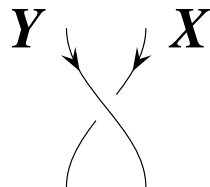
In theories with least 2 dimensions of space, we use *braided* monoidal categories. In Feynman diagrams, we draw the braiding

$$B_{X,Y}: X \otimes Y \rightarrow Y \otimes X$$

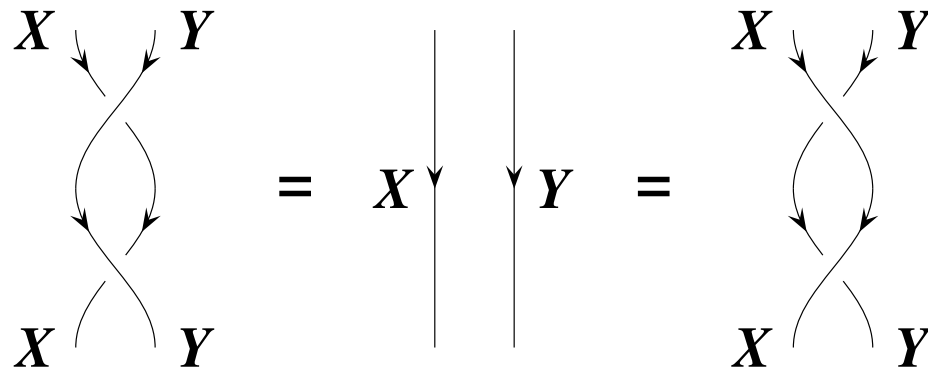
like this:



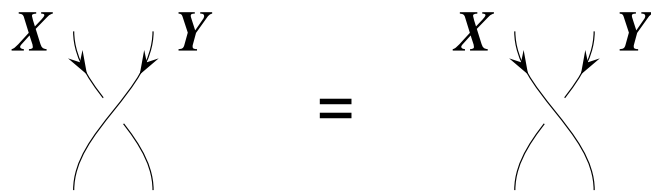
We draw its inverse like this:



**Then we have:**



**In theories with at least 3 dimensions of space, we use *symmetric* monoidal categories, where:**



**This lets us ‘untie knots’.**

**In physics, logic and computation, the most interesting monoidal categories are ‘closed’. Here for any objects  $Y, Z$  we have an ‘function type’ object  $Y \multimap Z$ , and a natural bijection**

$$\text{hom}(X \otimes Y, Z) \cong \text{hom}(X, Y \multimap Z)$$

**called *currying*.**

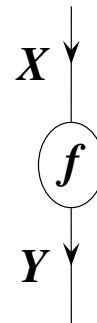
- In  $\text{Set}$ ,  $X \multimap Y$  is the set of functions from  $X$  to  $Y$ .**
- In  $\text{FinVect}$ ,  $X \multimap Y$  is the vector space of linear maps from  $X$  to  $Y$ .**
- In a category of propositions,  $X \multimap Y$  is the proposition ‘ $X$  implies  $Y$ ’, and currying says**

$$\frac{X \otimes Y \vdash Z}{Y \vdash X \multimap Z}$$

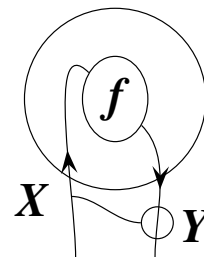
We can draw the identity morphism on  $X \multimap Y$  like this:

$$\begin{array}{c}
 \begin{array}{c}
 \text{---} \circ \\
 | \\
 X \uparrow \quad Y \downarrow
 \end{array}
 \quad = \quad
 \begin{array}{c}
 \downarrow \\
 X \multimap Y
 \end{array}
 \end{array}$$

Any morphism  $f: X \rightarrow Y$ , drawn as:



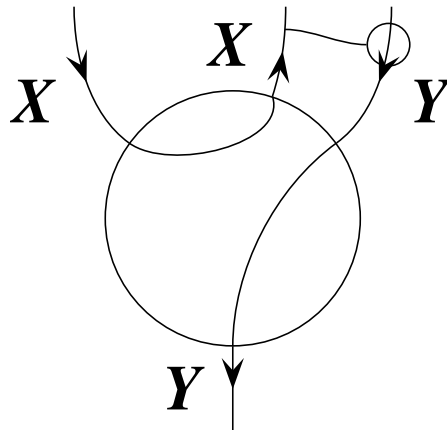
has a 'name'  $\ulcorner f \urcorner: I \rightarrow (X \multimap Y)$ , drawn as:



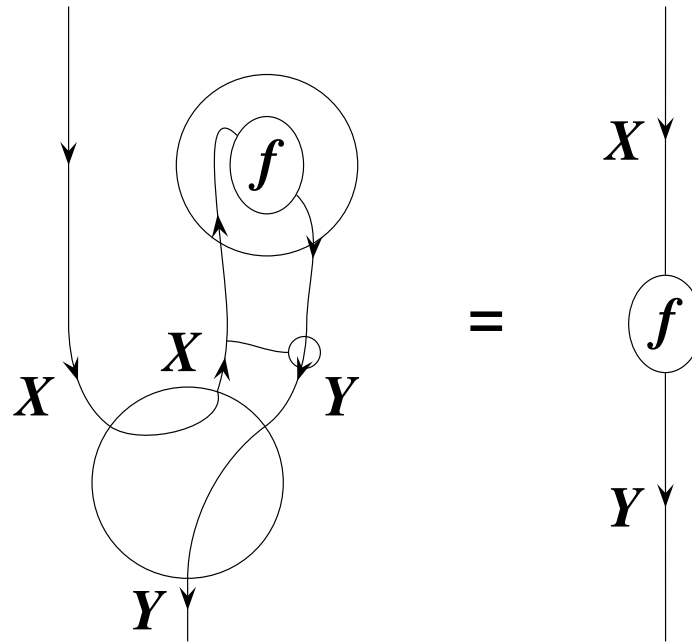
We also have an ‘evaluation’ morphism:

$$\text{ev}_{X,Y}: X \otimes (X \multimap Y) \rightarrow Y$$

drawn as:



Evaluating the name of  $f$  gives  $f$ :



In the  $\lambda$ -calculus this corresponds to  $\beta$ -reduction, treated as an *equation between morphisms*:

$$((\lambda x : X . f) a) = f[a/x]$$

In physics we often use a ‘compact’ closed category, where

$$X \multimap Y = X^* \otimes Y$$

The dual  $X^*$  corresponds to an *antiparticle* of type  $X$ , which Feynman drew as a particle going ‘backwards in time’:

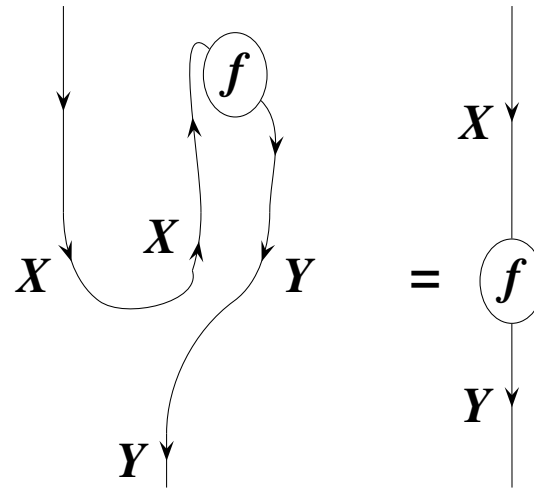
$$X \uparrow = X^* \downarrow$$

In a compact closed category, we have morphisms:

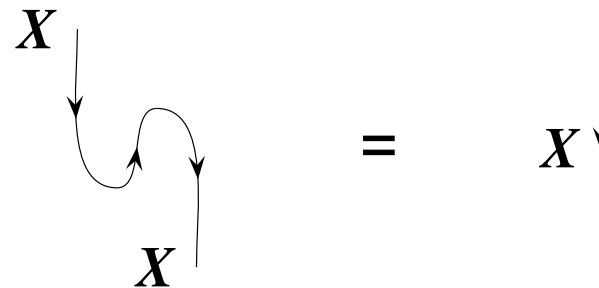
$$X \uparrow \curvearrowright X \qquad X \downarrow \cup X$$

In Feynman diagrams, these describe the creation and annihilation of virtual particle-antiparticle pairs!

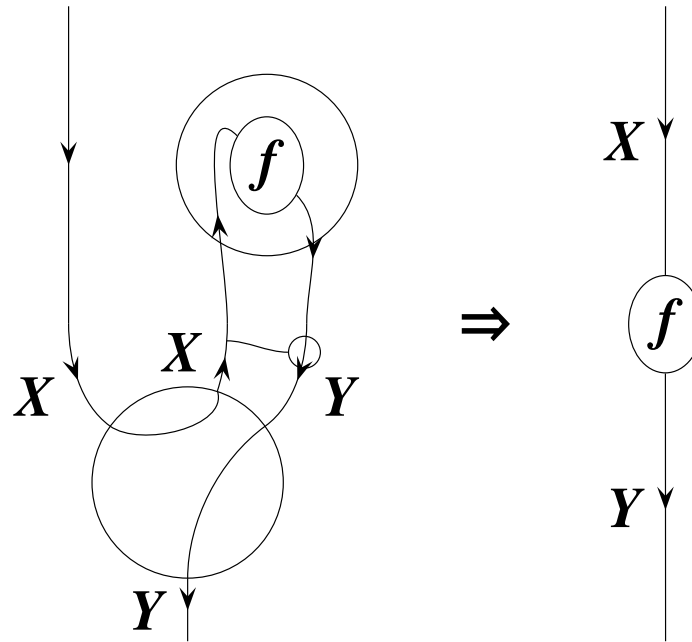
In a compact closed category, the equation that corresponds to  $\beta$ -reduction in the  $\lambda$ -calculus just says:



It follows from a simpler fact, the ‘zig-zag equation’:



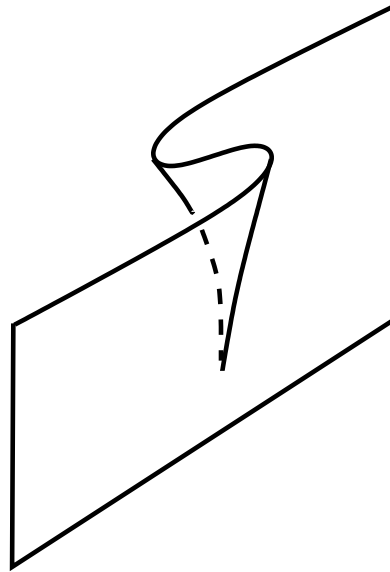
**In a monoidal closed 2-category, evaluating the name of  $f$  does not give  $f$ . It gives a morphism related to  $f$  by some 2-morphism:**



**In the  $\lambda$ -calculus this again corresponds to  $\beta$ -reduction, but now treated as a 2-morphism:**

$$((\lambda x : X . f) a) \Rightarrow f[a/x]$$

**In string theory, this 2-morphism corresponds to a surface like this:**



***It's the process of straightening out a zig-zag in a piece of string!***

See my webpage for online links to references, e.g.:

- **R. A. G. Seely, Modeling computations in a 2-categorical framework, *LICS* 1987.**
- **Barnaby P. Hilken, Towards a proof theory of rewriting: the simply-typed  $2\lambda$ -calculus, *Theor. Comp. Sci.* 170 (1996), 407.**
- **Albert Burroni, Higher-dimensional word problems with applications to equational logic, *Theor. Comp. Sci.* 115 (1993), 43.**
- **Yves Guiraud, The three dimensions of proofs, *Ann. Pure Appl. Logic* 141 (2006), 266.**
- **John Baez & Mike Stay, Physics, topology, logic and computation: a Rosetta Stone, to appear in *New Structures in Physics*, ed. Bob Coecke.**
- **John Baez & Aaron Lauda, A prehistory of  $n$ -categorical physics, to appear in *Deep Beauty*, ed. Hans Halvorson.**