9 Nov 2006

Classical vs. Quantum λ-calculus

For us, the "quantum λ-calculus" is a method of describing morphisms in a monoidal closed category (& eventually symmetric monoidal closed...). It's an alternative to the string diagram method. We're doing the typed λ-calculus, so instead of just saying something like

$$(f, a) \longmapsto f(a)$$

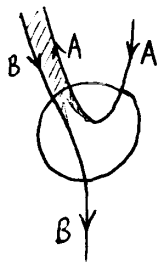
we'll write

$$(f,a) \in \hom(A,B) \otimes A \longmapsto f(a)$$

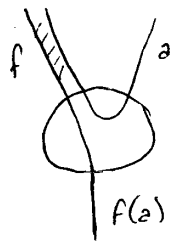
for the morphism

$$ev : \hom(A,B) \otimes A \longrightarrow B$$

or



in string diagram notation. Warning: a & f are dummy variables, not elements of any sets ($\hom(A,B)$, A, & B aren't necessarily sets!). In string diagram notation, the dummy variables are the strings themselves.



note: the labels in the previous diagram indicate the types of these variables.

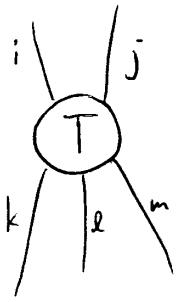
For example, in Vect, physicists write a morphism

$$T: V \otimes V \longrightarrow V \otimes V \otimes V$$

~~they denote~~ as

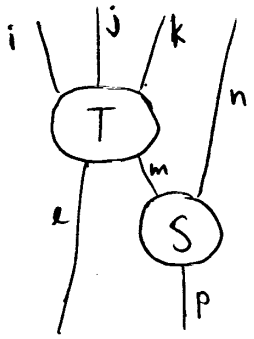
$$T^{ij}_{klm}$$

which corresponds to



& they write

$$T^{ijk}_{lm} S^{mn}_{p}$$ for



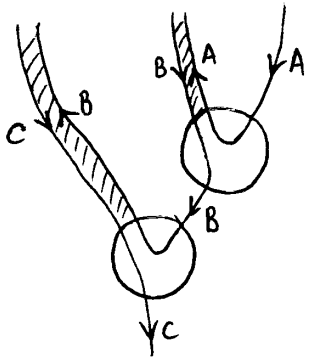
How about

$$(f, g, a) \in hom(B,C) \otimes hom(A,B) \otimes A \longmapsto f(g(a))$$

This is the λ-calculus for some morphism

$$hom(B,C) \otimes hom(A,B) \otimes A \longrightarrow C$$

which in string diagrams is:



$hom(B,C) \otimes hom(A,B) \otimes A$

$\downarrow 1 \otimes ev$

$hom(B,C) \otimes B$

$\downarrow ev$

C

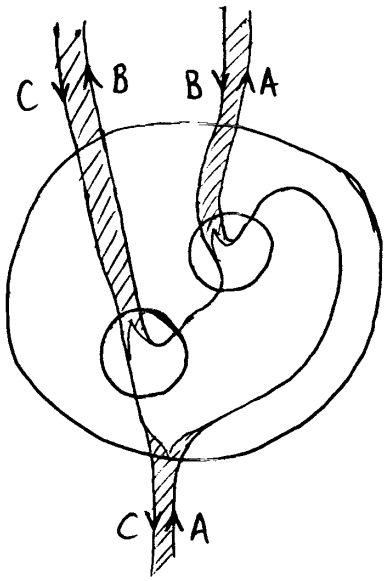
How about this

$$(f,g) \longmapsto f \circ g$$

Here I'm using new made-up notation. Translating it into official approved notation:

$$(f,g) \in \hom(B,C) \otimes \hom(A,B) \longmapsto (a \in A \longmapsto f(g(a)))$$

which has string diagram:



obtained by currying the previous example. So in any monoidal closed category we get morphisms

$$\circ : \hom(B,C) \otimes \hom(A,B) \longrightarrow \hom(A,C)$$

called <u>internal composition</u>, not to be confused with

$$\circ : \mathrm{Hom}(B,C) \times \mathrm{Hom}(A,B) \longrightarrow \mathrm{Hom}(A,C)$$

— ordinary "external" composition.

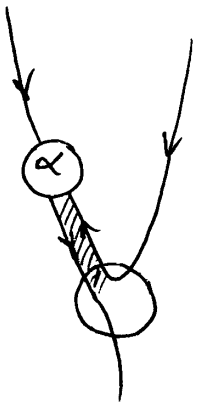
The So-Called "Untyped" λ-Calculus

A "type" is just an object in our monoidal category $\mathcal{C}$, but "untyped" just means we have an object $X \in \mathcal{C}$ with an isomorphism:

$$\alpha : X \xrightarrow{\sim} \hom(X,X)$$

This is a strange idea — it means we can turn "data" (of type X) into "programs" (that eat and spit out data of that type) — in a bijective way: all programs are data, all data are programs. So we get

$$X \otimes X \xrightarrow{\alpha \otimes 1} \hom(X,X) \otimes X \xrightarrow{ev} X$$



We'd like to use α to identify X with $\hom(X,X)$, and we can do this rigorously using:

Theorem: Suppose $\mathcal{C}$ is any category and $\alpha : A \xrightarrow{\sim} B$ is an isomorphism. There's an equivalence $F : \mathcal{C} \to \mathcal{C}'$ s.t. $F(A) = F(B)$ & $F(\alpha) = 1_{F(A)}$.

So, assume w.l.o.g. that

$$X = \hom(X,X) \quad \& \quad \alpha = 1_X .$$

If $\mathcal{C}$ is a cartesian closed category with this property, we can "build a computer" in $\mathcal{C}$.

First, we define natural numbers, called Church numerals, which are certain morphisms in $\mathcal{C}$. Here they are

$$\begin{aligned}
\bar{0} &:= f \in \hom(X,X) \longmapsto (a \longmapsto a) \\
\bar{1} &:= f \in \hom(X,X) \longmapsto (a \longmapsto f(a)) \\
\bar{2} &:= f \in \hom(X,X) \longmapsto (a \longmapsto f(f(a))) \\
&\vdots \qquad\qquad\qquad \vdots
\end{aligned}$$

& in general for any $n \in \mathbb{N}$ we get

$$\bar{n} := f \in \hom(X,X) \longmapsto (a \longmapsto \underbrace{f(\cdots f}_{n}(a)\cdots))$$

i.e.

$$\bar{n} : \hom(X,X) \longrightarrow \hom(X,X)$$

is "raising to the n". Note: since $X = \hom(X,X)$

$$\bar{n} : X \longrightarrow X$$

$$\text{"}\bar{n}\text{"} : 1 \longrightarrow X$$

so "$\bar{n}$" is an "element" of X.

So X has church numerals as "elements", but they may not be distinct. If $\mathcal{C}$ is the free CCC on one object with $X = \hom(X,X)$, they will be distinct.

We can also define "true" & "false" & logical operations in $\mathcal{C}$!

$$T := \quad x \longmapsto (y \longmapsto x)$$
$$F := \quad x \longmapsto (y \longmapsto y)$$

(curried versions of fns. that pick out the 1st & 2nd entries of an ordered pair)

where now all dummy variables are of type X.

$$\& := \quad (x,y) \longmapsto (x(y))(F)$$

Next time we'll prove

$$\&(T,T) = T$$
$$\&(T,F) = F$$
$$\&(F,T) = F$$
$$\&(F,F) = F$$