From typed λ-calculi to Cartesian closed categories

Last time we gave Lambek & Scott's definition of typed λ-calculus (minus their "natural numbers object"), involving

- types
- terms of a given type
- relations between terms

They show that for any typed λ-calculus, P, one can construct a CCC C_P where:

- objects of C_P are types of P
- morphisms f: A → B are (equivalence classes of) pairs (x∈A, ϕ(x)) where ϕ(x) is a term of type B with x as its only free variable.
- equations between morphisms (x∈A, ϕ(x)) and (y∈A, ψ(y)) where ϕ(x), ψ(y) are of the same type and
  \[
  \left[\begin{array}{c}
  \phi(x) = \\
  \psi(x)
  \end{array}\right].
  \]
Given morphisms

\[(x \in A, \varphi(x)) \quad \text{(say } \varphi(x) \in B)\]

and

\[(y \in B, \psi(y)) \quad \text{(say } \psi(y) \in C)\]

we define their composite to be

\[(x \in A, \psi(\varphi(x)))\]

Also, every object \( A \) has an identity morphism:

\[(x \in A, x)\]

Lambek & Scott show that \( Cp \) is a cartesian closed category, so we can think of a typed \( \lambda \)-calculus as a “presentation” of a CCC. In fact, they construct a category \( \lambda \text{Calc} \) where the objects are typed \( \lambda \)-calculi & the morphisms are “translations”, sending

- types to types
- terms to terms
- relations to relations

There’s a category \( \text{Cart} \) where the objects are CCCs and the morphisms are cartesian closed functors

\[F : C \rightarrow C'\]
F: C \rightarrow C' being cartesian means
\[ F(1_c) \cong 1_{c'}, \]

(where \(1_c\) denotes the terminal object of \(C\))

and
\[ F(A \times B) \cong F(A) \times F(B) \]

where we don't need to specify an isomorphism in either case since terminal objects and products are defined by universal properties.

F: C \rightarrow C' being closed means
\[ F(\text{hom}(A,B)) \cong \text{hom}(FA,FB) \]

where again we needn't specify an isomorphism since \(\text{hom}(A,B)\) is defined by a universal property:
\[
\begin{array}{c}
X \times A \rightarrow B \\
\downarrow \\
X \rightarrow \text{hom}(A,B)
\end{array}
\]

Lambek and Scott show there's a functor:
\[ C: \lambda\text{Calc} \rightarrow \text{Cart} \]

which we've just described on objects. They show this is an equivalence of categories, by constructing an
adjoint

\[ L : \text{Cart} \rightarrow \lambda \text{Calc} \]

where \( L C \) is the internal language of \( C \), having all objects of \( C \) as types, terms built from variables of each type & morphisms in \( C \), relations built from equations between morphisms.

Examples:

1) The typed \( \lambda \)-calculus for commutative rings. This has a type \( R \), for our ring, and thus other types including \( R \times R \), \( \text{hom}(R \times R, R) \), etc. We also have terms including

\[ + \in \text{hom}(R \times R, R) \]

\[ \cdot \in \text{hom}(R \times R, R) \]

\[ 0 \in R \]

\[ 1 \in R \]

and relations including the usual comm. ring axioms. For example, to state the distributive law we include the relation

\[ \cdot (x, + (y, z)) \equiv \bigoplus_{x, y, z} + (\cdot (x, y), \cdot (x, z)) \]

Let's call this typed \( \lambda \)-calculus \( \lambda \text{Th}(\text{CommRing}) \).
What's a cartesian closed functor

\[ F : C_{\lambda \text{Th}(\text{commRing})} \rightarrow \text{Set} \]

A typical object in \( C_{\lambda \text{Th}(\text{commRing})} \) is

\[ \text{hom}(\text{hom}(R \times R, 1), \text{hom}(R, R)) \]

i.e. any type in \( \lambda \text{Th}(\text{commRing}) \). Some typical morphisms in \( C_{\lambda \text{Th}(\text{commRing})} \) are

\[ (x \in \text{hom}(R, R), \ y \in R \mapsto x(x(y))) \]

(a morphism from \( \text{hom}(R, R) \) to \( \text{hom}(R, R) \) that "composes \( x \) with itself")

or

\[ (x \in R, \ +(x, \cdot (x, \cdot (x, x)))) \]

To be continued...