Physics, Topology, Logic, and Computation: A Rosetta Stone

John C. Baez
UC Riverside

Mike Stay
Google, U. of Auckland
The Rosetta Stone (pocket version)

| Category Theory | Physics  | Topology | Logic     | Computation |
|-----------------|----------|----------|-----------|-------------|
| object          | system   | manifold | proposition | data type   |
| morphism        | process  | cobordism| proof     | program     |
Objects

- String diagrams have ‘strings’ or ‘wires’:

- Quantum mechanics has Hilbert spaces: $X \cong \mathbb{C}^n$
- Topology has manifolds: $X \subset$
- Linear logic has propositions:
  
  $X = \text{“I have an item of type } X \text{.”}$

- Computation has datatypes: interface $X$
- SET has sets: $X$
Morphisms

• String diagrams have vertices:

\[
\begin{array}{c}
X \\
\circlearrowleft \\
Y
\end{array}
\]

• Quantum mechanics has linear transformations:

\[
f : X \rightarrow Y \cong f : \mathbb{C}^n \rightarrow \mathbb{C}^m
\]

(An \( m \times n \) matrix with complex entries)
Morphisms

- Topology has cobordisms: $\begin{array}{c} X \\ & \downarrow \\ Y \end{array}$
- Linear logic has constructive proofs: $X \vdash Y$
- Computation has programs: $Y \ f(X) ;$
- SET has functions: $f : X \rightarrow Y$
Morphisms compose associatively

• **String diagrams:**

![String Diagram](image)

• **Quantum mechanics: matrix multiplication**

• **Topology:**

![Topology Diagram](image)
Morphisms compose associatively

- **Linear logic:** \[ \frac{Y \vdash Z \quad X \vdash Y}{X \vdash Z} \] (\(\circ\))

- **Computation:**
  
  \[
  Y \ f(X \ x); \\
  Z \ g(Y \ y); \\
  \ldots \\
  z = g(f(x));
  \]

- **SET:** \((g \circ f) : X \rightarrow Z\)
Identity morphisms

• String diagrams:

• Quantum mechanics: identity matrix \(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\)

• Topology:

• Linear logic: \(\Gamma \vdash X \quad (i)\)

• Computation: \(X \; \text{id}(X \; x) \{ \text{return} \; x; \} \)

• SET: \(1_X : X \to X\)
Monoidal categories

• **String diagrams:**

\[
\begin{aligned}
X & \xrightarrow{f} Y \\
X' & \xrightarrow{f'} Y'
\end{aligned}
= \begin{aligned}
X & \xrightarrow{f \otimes f'} Y' \\
X' & \xrightarrow{f \otimes f'} Y'
\end{aligned}
= \begin{aligned}
X & \otimes X' Y \\
Y' & \otimes Y'
\end{aligned}
\]

• **Quantum mechanics: tensor product**

\[
\begin{pmatrix}
ab \\
cd
\end{pmatrix} \otimes 
\begin{pmatrix}
e f g \\
h j k
\end{pmatrix} = 
\begin{pmatrix}
ab & be & bf & bg \\
ah & aj & ak & bh & bj & bk \\
ce & cf & cg & de & df & dg \\
ch & cj & ck & dh & dj & dk
\end{pmatrix}
\]
Monoidal categories

• Topology:

• Linear logic: AND \( \frac{X \vdash Y \quad X' \vdash Y'}{X \otimes X' \vdash Y \otimes Y'} \) (\( \otimes \))

• Computation: parallel programming
  
  \[ \text{Pair}\langle X, X'\rangle \text{ pair}; \]

• SET: \( f \times f' : X \times X' \rightarrow Y \times Y' \)
Monoidal unit

- String diagram:
- Quantum mechanics: $I = \mathbb{C}$, the phase of a photon
- Topology:
- Linear logic: $I$, trivial proposition
- Computation: $I = \text{void}$ or $I = \text{unit type}$
- SET: one-element set $I$
Braided monoidal categories

• String diagrams:

\[ X \leftrightarrow Y \]

• Quantum mechanics: swap the particles. Bosons commute, fermions anticommute; quantized magnetic flux tubes in thin films, or “anyons”, can have arbitrary phase multiplier.
Braided monoidal categories

• Topology:

• Linear logic: \( W \vdash X \otimes Y \) (b)

\[ \frac{W \vdash Y \otimes X}{W \vdash X \otimes Y} \] (b)

• Computation: `pair.swap();`

• SET: \( b(\langle x, y \rangle) = \langle y, x \rangle \)
Braided monoidal closed categories

- String diagrams:

- Quantum mechanics: antiparticles

\[ 1_X : X \rightarrow X \cong \text{pair} : I \rightarrow X^* \otimes X \]
Braided monoidal closed categories

• Topology:

• Linear logic: IMPLIES \( \frac{X \otimes Y \vdash Z}{Y \vdash X \multimap Z} \) (c)

• Computation: Currying

\[
z = f(x, y); \\
or \\
z = f(y)(x);
\]

• SET: \( f : X \times Y \rightarrow Z \cong f : Y \rightarrow Z^X \)
Model Theory

\[ X \xrightarrow{f} Y \xleftarrow{g} X \]
Model Theory

\[ \text{Th(Graph)} \]
## Model Theory

| Th(Graph)        | SET                              |
|------------------|----------------------------------|
| object $V$       | set of vertices                  |
| object $E$       | set of edges                     |
| morphism $s : E \to V$ | function that picks out the source of each edge |
| morphism $t : E \to V$ | function that picks out the target of each edge |
A functor is a structure-preserving map. In Java terms, a functor picks out a class that implements the interface.
## Model Theory (Classical)

| Syntax [Programming language] | Semantics [CE-SET]                      |
|-------------------------------|----------------------------------------|
| data type                     | computably enumerable                  |
|                               | set of values                          |
| method                        | partially recursive function           |
## Model Theory (Quantum)

| Syntax [Programming language] | Semantics [QM]                  |
|------------------------------|---------------------------------|
| data type                    | Hilbert space of values         |
| method                       | linear transformation           |
Model Theory (Quantum)

| Syntax [Topology] | Semantics [QM]            |
|-------------------|----------------------------|
| manifold          | Hilbert space of states    |
| cobordism         | linear transformation      |

Topological Quantum Field Theory