Programming Languages in String Diagrams

( three )

Tensorial Logic

Paul-André Melliès

Oregon Summer School in Programming Languages

June 2011
Looking for the basic principles of logic

The discovery of the blood flow around
Claim: negation is the time of logic
The strategy (true $\otimes$ false)

The asynchronous game $B \otimes B$ has four leaves
The clique associated to (true $\otimes$ false)

\[ V \otimes F \]

\[ V \otimes \ast \quad \ast \otimes F \]

\[ \ast \otimes \ast \]
The positionality theorem of innocent strategies

Where are the hidden positions?
Tensorial logic

A primitive logic of tensor and negation
Five primitive components of logic

[1] the negation \( \neg \)

[2] the linear conjunction \( \otimes \)

[3] the repetition modality \( ! \)

[4] the existential quantification \( \exists \)

[5] the least fixpoint \( \mu \)

Logic = Data Structure + Duality
Tensorial logic

\[
\begin{align*}
A \vdash A & \quad \text{axiom} \\
\Gamma \vdash A & \quad \frac{}{\Gamma, \neg A \vdash} \text{left } \neg \\
\Gamma, A, B \vdash C & \quad \frac{}{\Gamma, A \otimes B \vdash C} \text{ left } \otimes \\
\Gamma \vdash A & \quad \frac{}{\Gamma, \text{true} \vdash A} \text{ left true} \\
\Gamma \vdash A, \Delta \vdash B & \quad \frac{}{\Gamma, \Delta \vdash B} \text{ cut} \\
\Gamma, A \vdash & \quad \frac{}{\Gamma, \Delta \vdash \neg A} \text{ right } \neg \\
\Gamma \vdash A & \quad \frac{}{\Gamma, \Delta \vdash \neg A} \text{ right } \otimes \\
\Gamma \vdash A & \quad \frac{}{\Gamma, \Delta \vdash A \otimes B} \text{ right true}
\end{align*}
\]
Tensorial logic

\[
\frac{\Gamma \vdash B}{\Gamma, !A \vdash B} \quad \text{weakening} \\
\frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B} \quad \text{dereliction} \\
\frac{\Gamma, A(x) \vdash B}{\Gamma, \exists x. A \vdash B} \quad \text{left } \exists \\
\frac{\Gamma \vdash A(t)}{\Gamma \vdash \exists x. A} \quad \text{right } \exists
\]

\[
\frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} \quad \text{contraction} \\
\frac{!\Gamma \vdash A}{!\Gamma \vdash !A} \quad \text{promotion}
\]
Tensorial logic

The boolean in linear logic:

\[ \mathcal{B} := (1 \oplus 1) \]

The boolean in tensorial logic:

\[ \mathcal{B} := \neg \neg (1 \oplus 1) \]

The classical boolean in tensorial logic:

\[ \mathcal{C} := \neg ! \neg (1 \oplus 1) \]
The left-to-right implementation of conjunction

\[ B \otimes B \rightarrow B \]

\[ q \]

\[ a_1 \]

\[ q \]

\[ a_2 \]

\[ a_1 \land a_2 \]
The right-to-left implementation of conjunction

\[ B \otimes B \rightarrow B \]

\[ q \]

\[ q \]

\[ a_2 \]

\[ q \]

\[ a_1 \]

\[ a_1 \land a_2 \]
The programming language PCF

The two implementations of the strict conjunction

\[ \mathbb{B} \otimes \mathbb{B} \vdash \mathbb{B} \]

have the same interpretation in the models of PCF.

In that sense, the language PCF is a linear language!
There is no taster of conjunction in PCF

It is not possible to program a strategy which behaves as follows:

\[(B \otimes B) \rightarrow B \rightarrow B\]

![Diagram](image)

The strategy is **innocent** but does not respect the **well-bracketing** condition which characterizes the programs implemented in the language PCF.
A taster of conjunction in PCF + control

But change \( \mathbb{B} \) into \( \mathcal{C} \) and it becomes possible to program a taster of conjunction:

\[
( \mathcal{C} \otimes \mathcal{C} ) \rightarrow \mathbb{B} \rightarrow \mathbb{B}
\]

- \( \mathcal{C} \otimes \mathcal{C} \rightarrow \mathbb{B} \rightarrow \mathbb{B} \)
  - \( q \) \( q \) \( q \)
  - \( q \) \( q \) \( q \) \( \text{true} \)

The \textbf{well-bracketing} condition is a \textbf{linearity} condition!
Conversely...

Linearity is implemented by **multi-bracketing**:

\[ B \otimes B \rightarrow B \]

This may be achieved with a semantics of **queries** and **responses**.
Conversely...

Linearity is implemented by **multi-bracketing**:

\[ C \otimes B \rightarrow B \]

This may be achieved with a semantics of **queries** and **responses**
Conversely...

Linearity is implemented by multi-bracketing:

![B ⊗ B → B]

This may be achieved with a semantics of **queries** and **responses**
A calculus of linear continuations & control

This means that the distinction between

<table>
<thead>
<tr>
<th>intuitionistic logic</th>
<th>classical logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$-calculus</td>
<td>$\lambda\mu$-calculus</td>
</tr>
<tr>
<td>well-bracketed</td>
<td>non well-bracketed</td>
</tr>
<tr>
<td>strategies</td>
<td>strategies</td>
</tr>
<tr>
<td>purely functional</td>
<td>control operators</td>
</tr>
</tbody>
</table>

is entirely regulated by the position of the resource modalities in the formula of tensorial logic – understood as a static type.

This enables to intertwine the two policies in the same language.
Back to the drinker formula

\[
\begin{align*}
A(x_0) \vdash A(x_0) & \quad \text{Axiom} \\
A(x_0) \vdash A(x_0), \forall x.A(x) \\
\vdash A(x_0), A(x_0) \Rightarrow \forall x.A(x) \\
\vdash A(x_0), \exists y.\{A(y) \Rightarrow \forall x.A(x)\} \\
\vdash \forall x.A(x), \exists y.\{A(y) \Rightarrow \forall x.A(x)\} \\
A(y_0) \vdash \forall x.A(x), \exists y.\{A(y) \Rightarrow \forall x.A(x)\} \\
\vdash A(y_0) \Rightarrow \forall x.A(x), \exists y.\{A(y) \Rightarrow \forall x.A(x)\} \\
\vdash \exists y.\{A(y) \Rightarrow \forall x.A(x)\}, \exists y.\{A(y) \Rightarrow \forall x.A(x)\} \\
\vdash \exists y.\{A(y) \Rightarrow \forall x.A(x)\} \\
\vdash \exists y.\{A(y) \Rightarrow \forall x.A(x)\}
\end{align*}
\]
The drinker formula in tensorial logic
The drinker formula in tensorial logic

\[ \exists y \Rightarrow \forall x \Rightarrow A(y) \Rightarrow A(x) \]

\[ \exists y \Rightarrow \forall x \Rightarrow \exists y \Rightarrow R \Rightarrow \forall x \Rightarrow L \Rightarrow \exists y \Rightarrow R \Rightarrow ?w \Rightarrow *A(y) \Rightarrow ?w \Rightarrow \forall x \Rightarrow \forall x \Rightarrow L \Rightarrow A(x) \]
The drinker formula in tensorial logic

\[
\exists y \implies \forall x \implies A(y) \mapsto R \mapsto \forall x \implies A(x)
\]

\[
\Rightarrow \text{ } \exists y \mapsto R \mapsto \forall x \mapsto A(x)
\]
A proof of the drinker formula

\[
A(x_0) \vdash A(x_0) \\
\neg A(x_0), A(x_0) \vdash \\
\neg A(x_0), !w A(x_0) \vdash \\
\neg A(x_0), !w A(x_0), !w \exists x \neg A(x) \vdash \\
\neg A(x_0) \vdash \neg \{!w A(x_0) \otimes !w \exists x \neg A(x)\} \\
\neg A(x_0) \vdash \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\}, \neg A(x_0) \vdash \\
\neg A(x_0) \vdash \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\}, \exists x \neg A(x) \vdash \\
\neg A(x_0) \vdash \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\}, , !w \exists x \neg A(x) \vdash \\
\neg A(x_0) \vdash \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\}, , !w A(y_0), , !w \exists x \neg A(x) \vdash \\
\neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\}, \neg A(x_0) \vdash \\
\neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \neg \{!w A(y_0) \otimes !w \exists x \neg A(x)\} \\
\neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \\
\neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \\
! \neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \\
\neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \\
\neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \\
\neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \\
\neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \neg \exists y \neg \{!w A(y) \otimes !w \exists x \neg A(x)\} \vdash \\
\neg A(x_0), !w A(x_0), !w \exists x \neg A(x) \vdash
\]

Axiom
Negation
Dereliction
Weakening
Negation
Right \exists
Negation
Left \exists
Dereliction
Weakening
Negation
Right \exists
Negation
Dereliction
Contraction
Negation
A proof of the drinker formula

\[
\begin{align*}
A(x_0) \vdash A(x_0) & \quad \text{Axiom} \\
\neg A(x_0), A(x_0) \vdash & \quad \text{Negation} \\
\neg A(x_0), !w A(x_0) \vdash & \quad \text{Negation}_2 \\
\neg A(x_0), !w A(x_0), !w \exists x \neg A(x) \vdash & \quad \text{Right}_\exists \\
\neg A(x_0) \vdash \neg \{ !w A(x_0) \otimes !w \exists x \neg A(x) \} & \quad \text{Negation}_1 \\
\neg A(x_0) \vdash \exists y \neg \{ !w A(y) \otimes !w \exists x \neg A(x) \} & \quad \text{Left}_\exists \\
\neg A (x_0) \vdash \neg \{ !w A(y) \otimes !w \exists x \neg A(x) \}, !w \exists x \neg A(x) \vdash & \quad \text{Negation}_2 \\
\neg \exists y \neg \{ !w A(y) \otimes !w \exists x \neg A(x) \}, !w \exists x \neg A(x) \vdash & \quad \text{Negation}_1 \\
\neg \exists y \neg \{ !w A(y) \otimes !w \exists x \neg A(x) \}, !w \exists x \neg A(x) \vdash & \quad \text{Contraction} \\
\neg \exists y \neg \{ !w A(y) \otimes !w \exists x \neg A(x) \}, !w \exists x \neg A(x) \vdash & \quad \text{Negation}_0 \\
!\neg \exists y \neg \{ !w A(y) \otimes !w \exists x \neg A(x) \} \vdash & \\
\vdash !\neg \exists y \neg \{ !w A(y) \otimes !w \exists x \neg A(x) \} 
\end{align*}
\]
A non valid formula
Tensorial logic

(axiom)

\[ \vdash P^*, P \]

↑-intro

\[ \vdash \Gamma, P \]

\[ \vdash \Gamma, \uparrow P \]

\[ \vdash \Gamma, M, N \]

\[ \vdash \Gamma, M \otimes N \]

false-intro

\[ \vdash \Gamma, P \]

\[ \vdash \Gamma, P^*, P \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, P^*, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

cut

↓-intro

\[ \vdash \Gamma, N \]

\[ \vdash \Gamma, \downarrow N \]

\[ \vdash \Gamma, P \]

\[ \vdash \Delta, Q \]

\[ \vdash \Gamma, \Delta, P \otimes Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, P \]

\[ \vdash \Delta, Q \]

\[ \vdash \Gamma, \Delta, P \otimes Q \]

\[ \vdash \Gamma, \Delta, P \otimes Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]

\[ \vdash \Gamma, \Delta, Q \]
Tensorial logic (1-sided formulation)

\[ \vdash \Gamma, P \quad \vdash \Gamma, ?N, P \quad \vdash \Gamma, ?N, ?N, P \quad \vdash \Gamma, ?N, P \]

weakening
contraction

dereliction
promotion

\[ \vdash \Gamma, N, P \quad \vdash \Gamma, N, P \]

\[ \vdash \Gamma, ?\Gamma, P \quad \vdash \Gamma, ?\Gamma, !P \]
dereliction
promotion

\[ \vdash \Gamma, N(x), P \quad \vdash \Gamma, \forall x. N(x), P \quad \vdash \Gamma, P(t) \quad \vdash \Gamma, \exists x. P(x) \]

\[ \vdash \Gamma, \exists x. N(x), P \]

∀-intro
∃-intro
Categories and functors

Functorial boxes in string diagrams
Categories

A category $\mathcal{C}$ is given by the following data:

— a class of objects,

— a set $\text{Hom}(A, B)$ of morphisms for every pair of objects $(A, B)$,

— a composition law $\circ : \text{Hom}(B, C) \times \text{Hom}(A, B) \to \text{Hom}(A, C)$

— an identity morphism $id_A \in \text{Hom}(A, A)$ for every object $A$. 
Categories

such that

1— the composition law \( \circ \) is associative

\[
\forall (f, g, h) \in \text{Hom}(A, B) \times \text{Hom}(B, C) \times \text{Hom}(C, D) \\
\quad f \circ (g \circ h) = (f \circ g) \circ h
\]

2— the morphisms \( \text{id} \) are neutral elements of \( \circ \)

\[
\forall f \in \text{Hom}(A, B) \\
\quad f \circ \text{id}_A = f = \text{id}_B \circ f
\]

Notation: one writes

\[
f : A \rightarrow B
\]

when

\[
f \in \text{Hom}(A, B)
\]
Functors

A functor

\[ F : \mathcal{C} \rightarrow \mathcal{D} \]

is given by

- an object \( FA \) of \( \mathcal{D} \) for every object \( A \) of \( \mathcal{C} \),

- a function \( F : \text{Hom}_\mathcal{C}(A, B) \rightarrow \text{Hom}_\mathcal{D}(FA, FB) \) for all objects \( A, B \) of \( \mathcal{C} \).

One requires moreover that \( F \) respects the identities:

\[ FA \xrightarrow{Fid_A} FA = FA \xrightarrow{id_{FA}} FA \]

and the composition:

\[ FA \xrightarrow{Ff} FB \xrightarrow{Fg} FC = FA \xrightarrow{F(g \circ f)} FC \]
First illustration – partial orders

Every partially ordered set \((X, \leq)\) defines a category whose objects are the elements of \(X\), and in which:

\[
\text{Hom}_C(x, y) = \begin{cases} 
\{\ast\} & \text{si } x \leq y \\
\emptyset & \text{otherwise}
\end{cases}
\]

In particular, there exists a morphism at most between two objects.
Second illustration – monoids

A monoid \((M, \cdot, e)\) is a set \(M\) equipped with a binary product and a neutral element, such that:

- **Associativity** \(\forall x, y, z \in M, (x \cdot y) \cdot z = x \cdot (y \cdot z)\)
- **Unit** \(\forall x \in M, \quad x \cdot e = x = e \cdot x\).

A homomorphism from \((M, \cdot, e)\) to \((N, \bullet, u)\) is a function

\[
f : M \longrightarrow N
\]

which respects the unit

\[
f(e) = u
\]

and the product

\[
\forall x, y \in M, \quad f(x \cdot y) = f(x) \bullet f(y)
\]
Second illustration – monoids

Every monoid

\(( M, \cdot, e)\)

induces a category

\(\Sigma M\)

called the **suspension** of \(M\) with a single object \(*\) and

\(\text{Hom}_{\Sigma M}(\ast, \ast) = M\)

with identity and composition defined as

\[ id_\ast := e \quad \quad x \circ y := x \cdot y \]
Second illustration – monoids

Every category $\mathcal{C}$ with a single object is equal to the suspension $\Sigma M$ of a monoid $M$. Moreover, a functor

$$F : \Sigma M \rightarrow \Sigma N$$

is the same thing as a monoid homomorphism

$$f : M \rightarrow N$$

Hence, a category is often called a monoid with several objects.
Third illustration – deterministic automata

Every alphabet $A$ generates a free monoid

$$A^* = \bigcup_{n \in \mathbb{N}} A^n.$$ 

A functor

$$\Sigma A^* \rightarrow \text{Set}$$

is the same thing as a **deterministic** automaton on the alphabet $A$.

An automaton with no initial position, no terminal position.
Transformations

A transformation

\[ \theta : F \rightarrow G \]

between two functors

\[ F, G : A \rightarrow B \]

is a family of morphisms of the category \( B \)

\[ ( \theta_A : FA \rightarrow GA )_{A \in \text{Obj}(A)} \]

indexed by the objects of the category \( A \).
Natural transformations

A transformation

\[ \theta : F \rightarrow G : A \rightarrow B \]

is natural when the diagram

\[ \begin{array}{ccc}
FA & \xrightarrow{\theta_A} & GA \\
\downarrow^{Ff} & & \downarrow^{Gf} \\
FB & \xrightarrow{\theta_B} & GB 
\end{array} \]

commutes for every morphism

\[ \begin{array}{ccc}
A & \xrightarrow{f} & B 
\end{array} \]
Functorial boxes

Functors in string diagrams
Functorial boxes

\[
Ff : FB \rightarrow FB
\]
\[
FA \rightarrow FA
\]

\[
F : \mathcal{C} \rightarrow \mathcal{D}
\]

A window on the blue \(\mathcal{C}\) inside the ambiant \(\mathcal{D}\).
Functorial equalities
Lax monoidal functor

A **lax monoidal functor** is a functor $F : \mathcal{C} \to \mathcal{D}$ equipped with morphisms

$$m_{[A,B]} : FA \otimes FB \to F(A \otimes B)$$

$$m[-] : I \to FI$$

satisfying a series of coherence relations.

A **strong monoidal functor** is lax monoidal with **invertible** coercions.
The purpose of coercions

\[ F(A_1 \otimes A_2 \otimes A_3) \]

\[ m_{[A_1, A_2, A_3]} \]

\[ m_{[-]} \]
A **lax monoidal functor** is a box with **many inputs - one output**.

\[
F(f) \circ m_{[A_1, \ldots, A_k]} : FA_1 \otimes \cdots \otimes FA_k \rightarrow FB
\]
Functorial equalities (on lax functors)
Strong monoidal functors

A strong monoidal functor is a box with many inputs - many outputs
Functorial equalities (on strong functors)

\[
\begin{align*}
F &\colon \mathcal{A} 
\Rightarrow \mathcal{B} \\
\mathcal{C} &\colon \mathcal{A} 
\Rightarrow \mathcal{B}
\end{align*}
\]
Functorial equalities (on strong functors)
Natural transformations

A natural transformation

\[ \theta : F \rightarrow G : \mathcal{C} \rightarrow \mathcal{D} \]

satisfies the pictorial equality:
Monoidal natural transformations

A **monoidal** natural transformation

\[ \theta : F \to G : \mathcal{C} \to \mathcal{D} \]

satisfies the pictorial equality:

![Diagram showing the equality of monoidal natural transformations](image)
Adjunctions and monads
Adjonction

An **adjunction** is a triple \((L, R, \phi)\) where \(L\) and \(R\) are two functors

\[
L : A \longrightarrow B \\
R : B \longrightarrow A
\]

and \(\phi\) is a family of bijections

\[
\phi_{A,B} : B(LA, B) \cong A(A, RB)
\]

natural in \(A\) et \(B\). One also writes

\[
\begin{array}{ccc}
LA & \longrightarrow & B \\
\downarrow & & \downarrow \phi_{A,B} \\
A & \longrightarrow & RB
\end{array}
\]

One says that \(L\) is left adjoint to \(R\), noted \(L \dashv R\).
The naturality of the bijection $\phi$

Natural in $A$ and $B$ means that the family of bijections $\phi_{A,B}$ transforms every commutative diagram

\[
\begin{array}{c}
LA & \xrightarrow{g} & B \\
\downarrow{Lh_A} & & \downarrow{h_B} \\
LA' & \xrightarrow{f} & B'
\end{array}
\]

into a commutative diagram

\[
\begin{array}{c}
A & \xrightarrow{\phi_{A,B}(g)} & RB \\
\downarrow{h_A} & & \downarrow{Rh_B} \\
A' & \xrightarrow{\phi_{A',B'}(f)} & RB'
\end{array}
\]
Example: the free monoid

where

\[ A = \text{Sets} : \text{the category of sets and functions} \]
\[ B = \text{Monoids} : \text{the category of monoids and homomorphisms} \]
\[ R : \text{the « forgetful » functor} \quad (M, \cdot, u) \mapsto M \]
\[ L : \text{the « free monoid » functor} \quad A \mapsto A^* \]

\[ A^* := \bigcup_{n \in \mathbb{N}} A^n \]
Illustration: the tensor algebra

\[ \begin{array}{c}
\text{Vect} \\ \perp \\
\text{Alg} \\
\end{array} \xrightarrow{R} \xleftarrow{L} \]

where

\[ A = \text{Vect} : \text{the category of vector spaces} \]
\[ B = \text{Alg} : \text{the category of algebras and homomorphisms}, \]

\[ R : \text{the « forgetful » functor } A \mapsto U(A). \]
\[ L : \text{the « free algebra » functor } V \mapsto TV. \]

\[ TV := \bigoplus_{n \in \mathbb{N}} V \otimes n \]
Illustration: the free category

where

\[ A = \text{Graph} : \quad \text{the category of graphs,} \]
\[ B = \text{Cat} : \quad \text{the category of categories and functors,} \]
\[ R : \quad \text{the « forgetful » functor} \]
\[ L : \quad \text{the « free category » functor} \]
Illustration: the terminal object

where

\[ A = C \quad : \quad \text{any category equipped with a terminal object } 1, \]
\[ B = 1 \quad : \quad \text{the singleton category}, \]
\[ R \quad : \quad \text{the functor whose image is the terminal object } 1, \]
\[ L \quad : \quad \text{the canonical (and unique) functor} \]
An algebraic presentation of the adjunction

An adjonction is a quadruple \((L, R, \eta, \varepsilon)\) where \(L\) and \(R\) are functors
\[
L : \mathcal{A} \to \mathcal{B} \quad R : \mathcal{B} \to \mathcal{A}
\]
and \(\eta\) and \(\varepsilon\) are natural transformations:
\[
\eta : \text{Id}_\mathcal{A} \to RL \quad \varepsilon : LR \to \text{Id}_\mathcal{B}
\]
such that the composite are the identities: (of \(L\) and \(R\) respectively).

\[
\begin{array}{c}
R \xrightarrow{\eta R} RLR \xrightarrow{R\varepsilon} R \\
L \xrightarrow{L\eta} LRL \xrightarrow{\varepsilon F} L
\end{array}
\]

The situation is depicted as follows:
Dialogue categories

At the interface between algebra, games and logic
 Cartesian closed categories

A **cartesian** category $\mathcal{C}$ is **closed** when there exists a functor

$$\Rightarrow : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{C}$$

and a natural bijection

$$\varphi_{A,B,C} : \mathcal{C}(A \times B, C) \cong \mathcal{C}(A, B \Rightarrow C)$$
The free cartesian closed category

The objects of the category $\text{free-ccc}(\mathcal{C})$ are the formulas

$$A, B ::= X \mid A \times B \mid A \Rightarrow B \mid 1$$

where $X$ is an object of the category $\mathcal{C}$.

The morphisms are the simply-typed $\lambda$-terms, modulo $\beta\eta$-conversion.
The simply-typed $\lambda$-calculus

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>$x : X \vdash x : X$</td>
</tr>
<tr>
<td>Abstraction</td>
<td>$\Gamma, x : A \vdash P : B$ $\vdash \lambda x. P : A \Rightarrow B$</td>
</tr>
<tr>
<td>Application</td>
<td>$\Gamma \vdash P : A \Rightarrow B$ $\Delta \vdash Q : A$ $\vdash \Gamma, \Delta \vdash PQ : B$</td>
</tr>
<tr>
<td>Weakening</td>
<td>$\vdash P : B$ $\vdash \Gamma, x : A \vdash P : B$</td>
</tr>
<tr>
<td>Contraction</td>
<td>$\Gamma, x : A, y : A \vdash P : B$ $\vdash \Gamma, z : A \vdash P[x, y \leftarrow z] : B$</td>
</tr>
<tr>
<td>Permutation</td>
<td>$\Gamma, x : A, y : B, \Delta \vdash P : C$ $\vdash \Gamma, y : B, x : A, \Delta \vdash P : C$</td>
</tr>
</tbody>
</table>
Proof invariants

Every ccc $\mathcal{D}$ induces a proof invariant $[\cdot]$ modulo execution.

The functor $[\cdot]$ interprets every proof as a map in $\mathcal{D}$
Ribbon category

A braided category in which every object $A$ has a dual $A^*$ satisfying:
Dualities

Recall that a duality $A \dashv B$ is a pair of morphisms

$I \xrightarrow{\eta} A \otimes B$

$B \otimes A \xrightarrow{\epsilon} I$
Dualities

satisfying the two “zig-zag” equalities:

In that case, $A$ is called a **left dual** of $B$. 
Knot invariants

Every ribbon category $\mathcal{D}$ induces a knot invariant

$\text{free-ribbon}(\mathcal{C}) \xrightarrow{[\cdot]} \mathcal{D}$

The functor $[\cdot]$ interprets every knot as a map in $\mathcal{D}$
Jones polynomial invariant

\[ \frac{2}{x^2} + \frac{1}{x^4} + \frac{y^2}{x^2} \]

\[ 2x^2 - x^4 + x^2y^2 \]

A compositional semantics of knots
Dialogue categories

A monoidal category with duality

Just as many morphisms

\[ A \rightarrow \neg B \]

as there are morphisms

\[ A \otimes B \rightarrow \bot \]

A familiar situation in algebra
Dialogue categories

A **symmetric monoidal** category $\mathcal{C}$ equipped with a functor

$$\neg : \mathcal{C}^{\text{op}} \rightarrow \mathcal{C}$$

and a natural bijection

$$\varphi_{A,B,C} : \mathcal{C}(A \otimes B, \neg C) \cong \mathcal{C}(A, \neg (B \otimes C))$$
The free dialogue category

The objects of the category \( \text{free-dialogue}(\mathcal{C}) \) are dialogue games constructed by the grammar

\[
A, B ::= X \mid A \otimes B \mid \neg A \mid 1
\]

where \( X \) is an object of the category \( \mathcal{C} \).

The morphisms are total and innocent strategies on dialogue games.

As we will see: proofs are 3-dimensional variants of knots...
A presentation of logic by generators and relations

**Negation** defines a pair of **adjoint functors**

\[ \begin{align*}
  \mathcal{C}(A, \neg B) & \cong \mathcal{C}(B, \neg A) & \cong \mathcal{C}^{\text{op}}(\neg A, B)
\end{align*} \]
The 2-dimensional topology of adjunctions

The **unit** and **counit** of the adjunction \( L \dashv R \) are depicted as

\[
\eta : \text{Id} \to R \circ L \\
\varepsilon : L \circ R \to \text{Id}
\]

Opponent move = functor \( R \)  \\
Proponent move = functor \( L \)
A typical proof

Reveals the algebraic nature of game semantics
A purely diagrammatic cut elimination
The 2-dimensional dynamics of adjunctions

\[ L \xrightarrow{\eta} L = L \xrightarrow{\epsilon} L \]

\[ R \xrightarrow{\eta} R = R \xrightarrow{\epsilon} R \]

Recovers the usual way to compose strategies in game semantics
Interesting fact

There are just as many canonical proofs

\[
\vdash 2p \supset \vdash 2q \\
\vdash \neg \cdots \neg \ A \qquad \vdash \neg \cdots \neg \ A
\]

as there are maps

\[
[p] \quad \rightarrow \quad [q]
\]

between the ordinals \([p] = \{0 < 1 < \cdots < p - 1\}\) and \([q]\).

This fragment of logic has the same combinatorics as simplices.
The two generators of a monad

Every increasing function is composite of **faces** and **degeneracies**:

\[
\eta : [0] \vdash [1]
\]
\[
\mu : [2] \vdash [1]
\]

Similarly, every proof is composite of the two generators:

\[
\eta : A \vdash \neg\neg A
\]
\[
\mu : \neg\neg\neg\neg A \vdash \neg\neg A
\]

The unit and multiplication of the double negation monad
The two generators in sequent calculus

\[
\frac{A \vdash A}{A \vdash \neg \neg A} \quad 2
\]

\[
\frac{A \vdash A}{A, \neg A \vdash} \quad 1
\]

\[
\frac{\neg A \vdash \neg A}{A \vdash \neg \neg A} \quad 3
\]

\[
\frac{\neg A, \neg \neg A \vdash}{\neg \neg \neg \neg A, \neg A \vdash} \quad 4
\]

\[
\frac{\neg \neg A, \neg \neg \neg \neg A \vdash}{\neg \neg \neg \neg \neg \neg \neg \neg \neg A \vdash} \quad 5
\]

\[
\frac{\neg \neg A \vdash \neg \neg A}{\neg \neg \neg \neg \neg \neg \neg \neg \neg \neg \neg \neg A \vdash} \quad 6
\]
The two generators in string diagrams

The unit and multiplication of the monad $R \circ L$ are depicted as

$$\eta : \text{Id} \rightarrow R \circ L$$

$$\mu : R \circ L \circ R \circ L \rightarrow R \circ L$$
Tensor and negation

Looking for the primitive components of logic
Guiding idea

A proof $\pi : A \vdash B$ is a linguistic phenomenon where

Proponent tries to convince Opponent

Opponent tries to refute Proponent

which we would like to decompose in elementary particles of logic
The linear decomposition of the intuitionistic arrow

\[ A \Rightarrow B = (\forall A) \rightarrow B \]

[1] a proof of \( A \rightarrow B \) uses its hypothesis \( A \) exactly once,

[2] a proof of \( \forall A \) is a bag containing an infinite number of proofs of \( A \).

Andreas Blass already knew this decomposition in 1972...
Five primitive components of logic

[1] the negation \( \neg \)
[2] the linear conjunction \( \otimes \)
[3] the repetition modality \( ! \)
[4] the existential quantification \( \exists \)
[5] the least fixpoint \( \mu \)

Logic = Data Structure + Duality
Tensor vs. negation

The continuation monad is strong

\[(\neg\neg A) \otimes B \rightarrow \neg\neg (A \otimes B)\]

The starting point of the algebraic theory of side effects
Tensor vs. negation

Proofs are generated by a **parametric strength**

\[ \kappa_X : \neg (X \otimes \neg A) \otimes B \longrightarrow \neg (X \otimes \neg (A \otimes B)) \]

which generalizes the usual notion of **strong monad**:

\[ \kappa : \neg \neg A \otimes B \longrightarrow \neg \neg (A \otimes B) \]
The proof in the sequent calculus

\[
\begin{align*}
A & \vdash A & B & \vdash B \\
A, B & \vdash A \otimes B \\
\neg (A \otimes B), A, B & \vdash \\
X & \vdash X & \neg (A \otimes B), A & \vdash \neg B \\
X, \neg (A \otimes B), A & \vdash X \otimes \neg B \\
X, \neg (A \otimes B), A, \neg (X \otimes \neg B) & \vdash \\
X \otimes \neg (A \otimes B), A \otimes \neg (X \otimes \neg B) & \vdash \\
A \otimes \neg (X \otimes \neg B) & \vdash \neg (X \otimes \neg (A \otimes B))
\end{align*}
\]
Proofs as 3-dimensional string diagrams

The left-to-right proof of the sequent

\[ \neg\neg A \otimes \neg\neg B \vdash \neg\neg (A \otimes B) \]

is depicted as
Tensor vs. negation: conjunctive strength

Linear distributivity in a continuation framework
Tensor vs. negation: disjunctive strength

Linear distributivity in a continuation framework
A factorization theorem

The four proofs \( \eta, \epsilon, \kappa^{\oplus} \) and \( \kappa^{\ominus} \) generate every proof of the logic. Moreover, every such proof generates a factorization

\[
X \rightarrow \epsilon \rightarrow \kappa^{\oplus} \rightarrow \epsilon \rightarrow \epsilon \rightarrow \eta \rightarrow \eta \rightarrow \kappa^{\ominus} \rightarrow \epsilon \rightarrow \eta \rightarrow \eta \rightarrow \kappa^{\oplus} \rightarrow \eta \rightarrow \eta \rightarrow Z
\]

factors uniquely as

\[
X \rightarrow \kappa^{\oplus} \rightarrow \epsilon \rightarrow \eta \rightarrow \kappa^{\ominus} \rightarrow Z
\]

Corollary: two proofs are equal iff they are equal as strategies.
Evaluation of a strategy against a counter-strategy

the formula $X$ is $\ominus$-free and the formula $Z$ is $\ominus$-free

hence, the formula $Y$ is both $\ominus$-free and $\ominus$-free

hence, the formula $Y$ is a play!
Basic $\eta - \varepsilon$ permutations
Permutation $\kappa \uplus - \varepsilon$
Permutation $\kappa^\bigcirc - \kappa^\bigotimes$ (1)
Permutation $\kappa \oplus - \kappa \ominus$ (2)
Revisiting the negative translation

A rational reconstruction of linear logic
The algebraic point of view (in the style of Boole)

The negated elements of a Heyting algebra form a Boolean algebra.
The algebraic point of view (in the style of Frege)

The double negation monad is idempotent when

In that case, the negated elements form a $\ast$-autonomous category.
The continuation monad is strong

\[(\lnot\lnot A) \otimes B \xrightarrow{lst} \lnot(A \otimes B)\]

\[A \otimes \lnot\lnot B \xrightarrow{rst} \lnot(A \otimes B)\]
The continuation monad is not commutative

There are two canonical morphisms

\[ \neg\neg A \otimes \neg\neg B \Rightarrow \neg\neg (A \otimes B) \]

Left strict and

Right strict and
Asynchronous games

Game semantics extended to linear logic by identifying the two strategies — hence mystifying the innocent audience.
Asynchronous games

Game semantics extended to linear logic by identifying the two strategies — hence mystifying the innocent audience.
Asynchronous games

Game semantics extended to linear logic by identifying the two strategies — hence mystifying the innocent audience.
Hence, the schism between games and linear logic

The isomorphism

\[ A \cong \neg
\neg A \]

means that linear logic is **de-temporalized** and **a posteriori**.

Imagine that a discussion is defined by the set of its final states
Tensorial logic

tensorial logic = a logic of tensor and negation
= linear logic without \( A \cong \neg \neg A \)
= the syntax of dialogue games

A synthesis between linear logic, games and continuations

Research program: integrate low level languages in the picture
Tensorial proof-nets

Axiom and cut links in game semantics
Axiom and cut links

When the formula $G = F^*$ is the negation of the formula $F$:

- **Axiom link**

- **Cut link**
Equalities between axiom and cut links
Equalities between axiom and cut links
Tensorial proof-nets

First observation.

One recovers a proof-net of linear logic by forgetting the flow of negations in the tensorial proof-net.

Second observation.

The traditional problems with the tensorial unit $\bot$ of linear logic come from the fact that the flow of negations is removed.

Third observation.

A notion of correctness criterion for tensorial proof-nets.
Multi-threaded strategies

The tensorial formulation of the mix rule
Multi-threaded strategies

\[ \neg A \otimes \neg B \rightarrow \neg (A \otimes B) \]

The tensorial formulation of the mix rule
Multi-threaded strategies

Additional hypothesis that negation defines a monoidal functor
Conclusion

Logic = Data Structure + Duality

This point of view is revealed by game semantics