Quantum Geometry of Interaction

Ichiro Hasuo  
University of Tokyo (JP)

Naohiko Hoshino  
RIMS, Kyoto University (JP)

Talk based on:  
I. Hasuo & N. Hoshino,  
Semantics of Higher-Order Quantum Computation via Geometry of Interaction,  
In Proc. Logic in Computer Science (LICS), June 2011.
What’s Done

* The Categorical GoI workflow
  * GoI = “Geometry of Interaction”
  * General, standard construction of denotational models

* Applied to quantum computation
  * Quantum $\lambda$-calculus = linear $\lambda$-cal. + quantum constructs
  * with insights from theory of coalgebra
  * Outcome: first adequate denotational semantics for a full quantum language (with ! and recursion)
Plan

* The categorical GoI workflow
  [Abramsky, Haghverdi, Scott, Jacobs, Longley, Lenisa, Hoshino, ...]

* GoI + realizability

* Generic — still concrete and dynamic

* Coalgebraic view ➔ let’s do something fancy

* Elements of quantum computation

  * Not much, really!

* The calculus $q\lambda_\ell$ Based on [Selinger-Valiron’09]

* The denotational model
## Quantum $\lambda$-calculus

<table>
<thead>
<tr>
<th>Classical</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Boolean) circuit</td>
<td>Quantum circuit</td>
</tr>
</tbody>
</table>

### Programming language

```c
int i, j;
int factorial(int k)
{
    j = 1;
    for (i = 1; i <= k; i++)
    {
        j = j * i;
    }
    return j;
}
```
# Quantum $\lambda$-calculus

## Classical vs. Quantum

<table>
<thead>
<tr>
<th>Classical</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Boolean) circuit</td>
<td>Quantum circuit</td>
</tr>
<tr>
<td>Programming language</td>
<td>Quantum programming language</td>
</tr>
</tbody>
</table>

### Classical Programming Language

```c
int i, j;
int factorial(int k) {
    j = 1;
    for (i = 1; i <= k; i++)
        j = j * i;
    return j;
}
```

### Quantum Programming Language

```plaintext
telep = let (x, y) = EPR * in
       let f = BellMeasure x in
       let g = U y
       in (f, g).
```

---

Hasuo (Tokyo)
## Quantum $\lambda$-calculus

### Classical vs. Quantum

<table>
<thead>
<tr>
<th>Classical</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Boolean) circuit</td>
<td>Quantum circuit</td>
</tr>
<tr>
<td>Programming language</td>
<td>Quantum programming language</td>
</tr>
</tbody>
</table>

### Classical Program Example
```c
int i, j;
int factorial(int k) {
    j = 1;
    for (i = 1; i <= k; i++)
        j *= i;
    return j;
}
```

### Quantum Program Example
```
telep = let (x, y) = EPR * in
        let f = BellMeasure x in
        let g = U y
        in (f, g).
```

* Quantum $\lambda$: prototype of quantum functional language

Hasuo (Tokyo)
Quantum $\lambda$-Calculus:
Prototype of Quantum Functional Languages

* Why (high-level) language?
  ➔ structured programming

* Discovery of new algorithms

* Program verification
Quantum $\lambda$-Calculus:
Prototype of Quantum Functional Languages

* **Why** (high-level) language?
  ➜ **structured programming**

* Discovery of new algorithms
* Program verification

* **Why** functional language?
  ➜ **Mathematically nice and clean**

* Aids (denotational) semantics
* Transfer from classical to quantum
Quantum $\lambda$-Calculus:
Prototype of Quantum Functional Languages

* Why denotational semantics?
  ➔ For quantum communication as well as for quantum computation
Quantum λ-Calculus:
Prototype of Quantum Functional Languages

* Why denotational semantics?
  ➔ For quantum communication as well as for quantum computation

* “Absolute security” via e.g. quantum key distr.
Quantum λ-Calculus: 
Prototype of Quantum Functional Languages

* Why denotational semantics?
  ➔ For quantum communication as well as for quantum computation

* “Absolute security” via e.g. quantum key distr.

* Being tested for real-world usage
Quantum $\lambda$-Calculus:
Prototype of Quantum Functional Languages

* **Why** denotational semantics?
  ➔ For quantum communication as well as for quantum computation

* “Absolute security” via e.g. quantum key distr.

* Being tested for real-world usege

* Comm. protocols are notoriously error-prone; quantum primitives make it worse
Quantum $\lambda$-Calculus:
Prototype of Quantum Functional Languages

* Linear $\lambda$-calculus
  * “No cloning” by linearity:
  * Classical data (dupllicable) via $!$

* + Quantum primitives
  * State preparation
  * Unitary transformation
  * Measurement
"Quantum Data, Classical Control"

Quantum data

Illustration by N. Hoshino

Classical control

Hasuo (Tokyo)
"Quantum Data, Classical Control"

Illustration by N. Hoshino

Quantum data

\[ \frac{1}{\sqrt{2}} \]

Classical control

Hasuo (Tokyo)
“Quantum Data, Classical Control”

Quantum data

\[ \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} \]

Classical control

Illustration by N. Hoshino

Hasuo (Tokyo)
Denotational Semantics for Quantum $\lambda$

- In Hilb $\mathcal{H}$?
  - Not that easy. Classical data?

- [Selinger&Valiron’08] Den. sem. for the $!$-free fragment
- [Selinger&Valiron’09] Operational semantics (nice!)
- [Current Work]
  - The first model for the full fragment
    (with $!$ and recursion)
  - Categorical GoI:
    useful for “Quantum Data, Classical Control”
Critical Acclaim (?) for:

I. Hasuo & N. Hoshino, Semantics of Higher-Order Quantum Computation via Geometry of Interaction
Critical Acclaim (?) for:

I. Hasuo & N. Hoshino, Semantics of Higher-Order Quantum Computation via Geometry of Interaction

* “[T]he amount of material ... goes far beyond the 10-page limit ... Now, I understand that self-containedness is an impossible objective in cases like this, but ...” —Reviewer 3
Critical Acclaim (?) for:

I. Hasuo & N. Hoshino, Semantics of Higher-Order Quantum Computation via Geometry of Interaction

* “[T]he amount of material ... goes far beyond the 10-page limit ... Now, I understand that self-containedness is an impossible objective in cases like this, but ...” —Reviewer 3

* “This is clearly a 30-page paper (or more) than has been compressed into 10 pages.” —Reviewer 4
Critical Acclaim (?) for:
I. Hasuo & N. Hoshino, Semantics of Higher-Order Quantum Computation via Geometry of Interaction

* “[T]he amount of material ... goes far beyond the 10-page limit ... Now, I understand that self-containedness is an impossible objective in cases like this, but ...” —Reviewer 3

* “This is clearly a 30-page paper (or more) than has been compressed into 10 pages.” —Reviewer 4

* Now their pain is yours!!
Part 1

Categorical GoI

(Geometry of Interaction)
GoI: Geometry of Interaction

* J.-Y. Girard, at Logic Colloquium ’88
GoI: Geometry of Interaction

J.-Y. Girard, at Logic Colloquium ’88
GoI:
Geometry of Interaction

- J.-Y. Girard, at Logic Colloquium ’88

Disclaimer (and sincere apologies):
- I’m no linear logician!
GoI: Geometry of Interaction

- J.-Y. Girard, at Logic Colloquium ’88

Disclaimer (and sincere apologies):
- I’m no linear logician!

In this talk:
- Its categorical formulation
  [Abramsky,Haghverdi&Scott’02]
- “The GoI Animation”
The GoI Animation

\[ [M] = (\mathbb{N} \rightarrow \mathbb{N}, \text{a partial function}) \]

= “piping”

\[ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \ldots \quad \text{(countably many)} \]
The GoI Animation

\[ [M] = (\mathbb{N} \rightarrow \mathbb{N}, \text{a partial function}) \]

= “piping”

\( \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \)

1 2 3 4 ...

(countably many)

Hasuo (Tokyo)
The GoI Animation

\[ [M] = (\mathbb{N} \rightarrow \mathbb{N}, \text{a partial function}) \]

= “piping”

1 2 3 4 ...

(countably many)
The GoI Animation

\[
\llbracket M \rrbracket = (\mathbb{N} \rightarrow \mathbb{N}, \text{a partial function})
\]

= “piping”

... (countably many)
The GoI Animation

\[
[M] = ( \mathbb{N} \twoheadrightarrow \mathbb{N}, \text{ a partial function } )
\]

= “piping”

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\downarrow & \downarrow & \downarrow & \downarrow \\
\end{array}
\]

... (countably many)
The GoI Animation

\[[M]\] = (\mathbb{N} \rightarrow \mathbb{N}, \text{ a partial function})

= “piping”

\[1 \quad 2 \quad 3 \quad 4 \quad \ldots \quad \text{(countably many)}\]
\[ [M] = (\mathbb{N} \twoheadrightarrow \mathbb{N}, \text{a partial function}) \]

= "piping"

\[ \downarrow \downarrow \downarrow \downarrow \]

1 2 3 4 ...

(countably many)

Hasuo (Tokyo)

Sunday, September 11, 2011
The GoI Animation

* Function application $[MN]$

* by “parallel composition + hiding”
\[ \begin{bmatrix} M & N \end{bmatrix} = \begin{bmatrix} M \end{bmatrix} \quad \begin{bmatrix} N \end{bmatrix} \]
\[ [M N] \]

\[ = \]

\[ M \]

\[ N \]
\[ MN \] = \[ M \]

\[ N \]
\[MN\] =
\[ \begin{bmatrix} MN \end{bmatrix} = \begin{bmatrix} M \\ N \end{bmatrix} \]
\[ MN \] = ...
\[ MN \] =

\[ M \]

\[ N \]

“parallel composition + hiding” (cf. games)
\[ MN \]

\[ = \]

\[ M = \lambda x. x + 1 \quad N = 2 \]

\[ M = \lambda x. 1 \quad N = 2 \]

\[ M = \lambda f. f 1 \quad N = \lambda x. (x + 1) \]
\[ M = \lambda x. x + 1 \]
\[ N = \lambda x. (x + 1) \]
\[ I + 1 = I \]
\[ N = 2 \]
\[ M = \lambda x. 1 \]
\[ N = \lambda f. f 1 \]
\[ [MN] = \]

\[ \rightarrow M = \lambda x. x + 1 \quad N = 2 \]
\[ M = \lambda x. 1 \quad N = 2 \]
\[ M = \lambda f. f1 \quad N = \lambda x. (x + 1) \]
\[ MN \]

\[ = \]

\[ M = \lambda x. x + 1 \quad N = 2 \]
\[ M = \lambda x. 1 \quad N = 2 \]
\[ M = \lambda f. f1 \quad N = \lambda x. (x + 1) \]
\[ MN \]

\[ = \]

\[ M = \lambda x. x + 1 \quad N = 2 \]
\[ M = \lambda x. 1 \quad N = 2 \]
\[ M = \lambda f. f1 \quad N = \lambda x. (x + 1) \]
\[ MN \]

\[
\begin{align*}
M &= \lambda x. x + 1 \\
N &= 2 \\
M &= \lambda x. 1 \\
N &= 2 \\
M &= \lambda f. f1 \\
N &= \lambda x. (x + 1)
\end{align*}
\]
\[
M = \lambda x. x + 1 \\
N = 2 \\
M = \lambda f. f I \\
N = \lambda x. (x + 1)
\]

\[
M = [N] \\
N = [M]
\]
\[ MN \]

\[ MN = \lambda x. x + 1 \quad N = 2 \]

\[ M = \lambda x. 1 \quad N = 2 \]

\[ M = \lambda f. f1 \quad N = \lambda x. (x + 1) \]
\[ MN \]

= 

\[ M = \lambda x. x + 1 \quad N = 2 \]

\[ M = \lambda x. 1 \quad N = 2 \]

\[ M = \lambda f. f1 \quad N = \lambda x. (x + 1) \]
\[ MN \]

\[ = \]

\[ M = \lambda x. x + 1 \quad N = 2 \]
\[ M = \lambda x. 1 \quad N = 2 \]
\[ \rightarrow M = \lambda f. f1 \quad N = \lambda x. (x + 1) \]
\[ [MN] = \]

\[
\begin{align*}
M &= \lambda x. x + 1 & N &= 2 \\
M &= \lambda x. 1 & N &= 2 \\
\rightarrow M &= \lambda f. f1 & N &= \lambda x. (x + 1)
\end{align*}
\]
$MN$ = 

$M = \lambda x. x + 1$  $N = 2$

$M = \lambda x. 1$  $N = 2$

$M = \lambda f. f1$  $N = \lambda x. (x + 1)$
\[MN\] =

\[\begin{align*}
M &= \lambda x. x + 1 \\
N &= 2 \\
M &= \lambda x. 1 \\
N &= 2 \\
\Rightarrow M &= \lambda f. f1 \\
N &= \lambda x. (x + 1)
\end{align*}\]
\[ MN \] = ...

\[
\begin{align*}
M &= \lambda x. x + 1 \\
N &= 2 \\
M &= \lambda x. 1 \\
N &= 2 \\
M &= \lambda f. f1 \\
N &= \lambda x. (x + 1)
\end{align*}
\]
Categorical GoI

- Axiomatics of GoI in the categorical language
- Abstraction & genericity, which we exploit

- Our main reference (recommended!):
  - Especially its technical report version (Oxford CL), since it’s more detailed
The Categorical GoI Workflow

- Traced monoidal category $\mathbb{C}$
  + other constructs $\rightarrow$ "GoI situation" [AHS02]

- Categorical GoI [AHS02]

- Linear combinatory algebra

- Realizability

- Linear category
The Categorical GoI Workflow

Traced monoidal category $\mathcal{C}$
+ other constructs $\rightarrow$ “GoI situation” [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category
The Categorical GoI Workflow

Traced monoidal category $\mathcal{C}$
+ other constructs $\Rightarrow$ “GoI situation” [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

- Applicative str. + combinators
- Model of untyped calculus
The Categorical GoI Workflow

Traced monoidal category $C$ + other constructs $\rightarrow$ “GoI situation” [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

Model of typed calculus

Applicative str. + combinators

Model of untyped calculus

Hasuo (Tokyo)
The Categorical GoI Workflow

Traced monoidal category $\mathcal{C}$ + other constructs $\Rightarrow$ "GoI situation" [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

Model of typed calculus

Weak linear category $\text{Int}(\mathcal{C})$

Int-constr. [Joyal, Street & Verity96]

\[
\begin{array}{c}
\begin{tikzpicture}
  \node (A) at (0,0) {$A$};
  \node (B) at (1,0) {$B$};
  \node (C) at (1,1) {$C$};
  \node (D) at (2,1) {$A$};
  \node (E) at (2,0) {$B$};
  \draw (A) -- (B);
  \draw (B) -- (C);
  \draw (C) -- (D);
  \draw (D) -- (E);
  \draw (A) to (D);
\end{tikzpicture}
\end{array}
\]

Applicative str. + combinators

Model of untyped calculus

Hasuo (Tokyo)

Sunday, September 11, 2011
The Categorical GoI Workflow

- Traced monoidal category $\mathbb{C}$
  + other constructs $\rightarrow$ “GoI situation” [AHS02]

- Categorical GoI [AHS02]

- Linear combinatory algebra

- Realizability

- Linear category

- Model of typed calculus
  * Applicative str. + combinators
  * Model of untyped calculus

Hasuo (Tokyo)
The Categorical GoI Workflow

- **Traced monoidal category** $\mathcal{C}$
  + other constructs $\to$ “GoI situation” [AHS02]

- **Categorical GoI** [AHS02]

- **Linear combinatory algebra**

**Realizability**

- Applicative str. + combinators
- Model of *untyped* calculus

**Linear category**

Model of *typed* calculus
Definition (LCA)
A linear combinatory algebra (LCA) is a set $A$ equipped with

- a binary operator (called an applicative structure)
  \[ \cdot : A^2 \rightarrow A \]

- a unary operator
  \[ ! : A \rightarrow A \]

- (combinators) distinguished elements $B, C, I, K, W, D, \delta, F$ satisfying

  - $Bxyz = x(yz)$ Composition, Cut
  - $Cxyz = (xz)y$ Exchange
  - $Ix = x$ Identity
  - $Kx!y = x$ Weakening
  - $Wx!y = x!y!y$ Contraction
  - $D!x = x$ Dereliction
  - $\delta!x = !!!x$ Comultiplication
  - $F!x!y = !(xy)$ Monoidal functoriality

Here: $\cdot$ associates to the left; $\cdot$ is suppressed; and $!$ binds stronger than $\cdot$ does.
Defn. (LCA)
A linear combinatory algebra (LCA) is a set $A$ equipped with

- a binary operator (called an applicative structure)
  \[ \cdot : A^2 \rightarrow A \]

- a unary operator
  \[ ! : A \rightarrow A \]

- (combinators) distinguished elements $B, C, I, K, W, D, \delta, F$ satisfying

\[
\begin{align*}
B & : x(yz) = x(yz) & \text{(Composition, Cut)} \\
C & : (x)(yz) = (xy)z & \text{(Exchange)} \\
I & : x = x & \text{(Identity)} \\
K & : x ! y = x & \text{(Weakening)} \\
W & : x ! y = x ! y ! y & \text{(Contraction)} \\
D & : x ! x = x & \text{(Dereliction)} \\
\delta & : x = ! ! x & \text{(Comultiplication)} \\
F & : x ! y = ! (xy) & \text{(Monoidal functoriality)}
\end{align*}
\]

Here: $\cdot$ associates to the left; $\cdot$ is suppressed; and $!$ binds stronger than $\cdot$ does.
Defn. (LCA)

A linear combinatory algebra (LCA) is a set $A$ equipped with

- a binary operator (called an applicative structure)
  $\cdot : A^2 \rightarrow A$

- a unary operator
  $! : A \rightarrow A$

- (combinators) distinguished elements $B, C, I, K, W, D, \delta, F$
  satisfying

  \[
  \begin{align*}
  Bxyz &= x(yz) & \text{Composition, Cut} \\
  Cxyz &= (xz)y & \text{Exchange} \\
  Ix &= x & \text{Identity} \\
  Kx!y &= x & \text{Weakening} \\
  Wx!y &= x!y!y & \text{Contraction} \\
  Dx &= x & \text{Dereliction} \\
  \delta x &= !!x & \text{Comultiplication} \\
  Fx!y &= !(xy) & \text{Monoidal functoriality}
  \end{align*}
  \]

Here: $\cdot$ associates to the left; $\cdot$ is suppressed; and $!$ binds stronger than $\cdot$ does.
**Linear Combinatory Algebra (LCA)**

**Defn. (LCA)**

A *linear combinatory algebra (LCA)* is a set $A$ equipped with

- a binary operator (called an *applicative structure*)
  
  
  \[ \cdot : A^2 \rightarrow A \]

- a unary operator
  
  \[ ! : A \rightarrow A \]

- (combinators) distinguished elements $B, C, I, K, W, D, \delta, F$

  satisfying

  \[
  \begin{align*}
  Bxyz & = x(yz) \quad \text{Composition, Cut} \\
  Cxyz & = (xz)y \quad \text{Exchange} \\
  ix & = x \quad \text{Identity} \\
  Kxy & = x \quad \text{Weakening} \\
  Wxy & = x ! y \quad \text{Contraction} \\
  Dxy & = x \quad \text{Dereliction} \\
  \delta x & = ! ! x \quad \text{Comultiplication} \\
  Fx & = ! (xy) \quad \text{Monoidal functoriality}
  \end{align*}
  \]

Here: $\cdot$ associates to the left; $\cdot$ is suppressed; and $!$ binds stronger than $\cdot$ does.

---

**What we want (outcome)**

- Model of untyped linear $\lambda$

- $a \in A \approx$ closed linear $\lambda$-term
Defn. (LCA)
A linear combinatory algebra (LCA) is a set $A$ equipped with

- a binary operator (called an applicative structure)
  $· : A^2 \rightarrow A$

- a unary operator
  $! : A \rightarrow A$

- (combinators) distinguished elements $B, C, I, K, W, D, δ, F$
  satisfying

  - $Bxyz = x(yz)$       Composition, Cut
  - $Cxyz = (xz)y$       Exchange
  - $lx = x$             Identity
  - $Kxy = x$            Weakening
  - $Wxy = x! y! y$      Contraction
  - $D!x = x$            Dereliction
  - $δ!x = !!x$          Comultiplication
  - $F!x!y = !(xy)$      Monoidal functoriality

Here: $·$ associates to the left; $·$ is suppressed; and $!$ binds stronger than $·$ does.

What we want (outcome)

- Model of untyped linear $λ$
- $a \in A \approx$ closed linear $λ$-term
- No $S$ or $K$ (linear!)
**Linear Combinatory Algebra (LCA)**

**Defn. (LCA)**
A linear combinatory algebra (LCA) is a set $A$ equipped with

- a binary operator (called an applicative structure)
  
  $\cdot : A^2 \rightarrow A$

- a unary operator
  
  $!: A \rightarrow A$

- (combinators) distinguished elements $B, C, I, K, W, D, \delta, F$

  satisfying

  - $Bxyz = x(yz)$ (Composition, Cut)
  - $Cxyz = (xz)y$ (Exchange)
  - $Ix = x$ (Identity)
  - $Kxy = x$ (Weakening)
  - $Wxy = x(yy)$ (Contraction)
  - $Dx = x$ (Dereliction)
  - $\delta x = !!x$ (Comultiplication)
  - $F!x = !(xy)$ (Monoidal functoriality)

Here: $\cdot$ associates to the left; $\cdot$ is suppressed; and $!$ binds stronger than $\cdot$ does.

---

- Model of untyped linear $\lambda$
- $a \in A \approx$ closed linear $\lambda$-term
- No $S$ or $K$ (linear!)
- Combinatory completeness: e.g.

  $\lambda xyz. zxy$

  designates elem. of $A$
**Defn.** (GoI situation [AHS02])

A *GoI situation* is a triple $(\mathbb{C}, F, U)$ where

- $\mathbb{C} = (\mathbb{C}, \otimes, I)$ is a traced symmetric monoidal category (TSMC);

- $F : \mathbb{C} \to \mathbb{C}$ is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).

  - $e : FF \triangleleft F : e'$, Comultiplication
  - $d : \text{id} \triangleleft F : d'$, Dereliction
  - $c : F \otimes F \triangleleft F : c'$, Contraction
  - $w : K_I \triangleleft F : w'$, Weakening

  Here $K_I$ is the constant functor into the monoidal unit $I$;

- $U \in \mathbb{C}$ is an object (called *reflexive object*), equipped with the following retractions.

  - $j : U \otimes U \triangleleft U : k$
  - $I \triangleleft U$
  - $u : FU \triangleleft U : v$
**GoI situation**

* Monoidal category \((C, \otimes, I)\)

* String diagrams

**Defn.** (GoI situation [AHS02])

A GoI situation is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \to C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  
  
  \[
  \begin{align*}
  e &: FF \otimes F : e' & \text{Comultiplication} \\
  d &: \text{id} \otimes F : d' & \text{Dereliction} \\
  c &: F \otimes F \otimes F : c' & \text{Contraction} \\
  w &: K_I \otimes F : w' & \text{Weakening}
  \end{align*}
  \]

  Here \(K_I\) is the constant functor into the monoidal unit \(I\);

- \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
  
  \[
  \begin{align*}
  j &: U \otimes U \otimes U : k \\
  I &: \text{id} \otimes U \\
  u &: FU \otimes U : v
  \end{align*}
  \]
GoI situation

\textbf{Defn. (GoI situation \cite{AHS02})}

A \textit{GoI situation} is a triple \((\mathcal{C}, F, U)\) where

- \(\mathcal{C} = (\mathcal{C}, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : \mathcal{C} \rightarrow \mathcal{C}\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  
  - \(e : FF \triangleleft F : e'\) \hspace{1cm} \text{Comultiplication}
  - \(d : \text{id} \triangleleft F : d'\) \hspace{1cm} \text{Dereliction}
  - \(c : F \otimes F \triangleleft F : c'\) \hspace{1cm} \text{Contraction}
  - \(w : K_I \triangleleft F : w'\) \hspace{1cm} \text{Weakening}

  Here \(K_I\) is the constant functor into the monoidal unit \(I\);

- \(U \in \mathcal{C}\) is an object (called \textit{reflexive object}), equipped with the following retractions.
  
  - \(j : U \otimes U \triangleleft U : k\)
  - \(I \triangleleft U\)
  - \(u : FU \triangleleft U : v\)
A GoI situation is a triple $(C, F, U)$ where

- $C = (C, \otimes, I)$ is a traced symmetric monoidal category (TSMC);
- $F : C \to C$ is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  
  - $e : FF \triangleleft F : e'$: Comultiplication
  - $d : id \triangleleft F : d'$: Dereliction
  - $c : F \otimes F \triangleleft F : c'$: Contraction
  - $w : K_I \triangleleft F : w'$: Weakening

  Here $K_I$ is the constant functor into the monoidal unit $I$;
- $U \in C$ is an object (called reflective object), equipped with the following retractions.
  
  - $j : U \otimes U \triangleleft U : k$
  - $I \triangleleft U$
  - $u : FU \triangleleft U : v$
**GoI situation**

**Defn.** (GoI situation [AHS02])

A GoI situation is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \to C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF < F : e'\) Comultiplication
  - \(d : \text{id} < F : d'\) Dereliction
  - \(c : F \otimes F < F : c'\) Contraction
  - \(w : K_I < F : w'\) Weakening

Here \(K_I\) is the constant functor into the monoidal unit \(I\);

- \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
  - \(j : U \otimes U < U : k\)
  - \(i < U\)
  - \(u : FU < U : v\)

**Monoidal category** \((C, \otimes, I)\)

**String diagrams**

\[
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
B & \xrightarrow{g} & C
\end{array}
\]

\[
A \xrightarrow{g \circ f} C
\]

\[
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
C & \xrightarrow{g} & D
\end{array}
\]

\[
A \otimes C \xrightarrow{f \otimes g} B \otimes D
\]

Hasuo (Tokyo)
**GoI situation**

**Defn.** (GoI situation \([AHS02]\))

A *GoI situation* is a triple \((\mathcal{C}, F, U)\) where

- \(\mathcal{C} = (\mathcal{C}, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : \mathcal{C} \to \mathcal{C}\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF \triangleleft F \to e'\) \quad \text{Comultiplication}
  - \(d : \text{id} \triangleleft F \to d'\) \quad \text{Dereliction}
  - \(c : F \otimes F \triangleleft F \to c'\) \quad \text{Contraction}
  - \(w : K_I \triangleleft F \to w'\) \quad \text{Weakening}

Here \(K_I\) is the constant functor into the monoidal unit \(I\);

- \(U \in \mathcal{C}\) is an object (called reflexive object), equipped with the following retractions.
  - \(j : U \otimes U \triangleleft U \to k\)
  - \(I \triangleleft U\)
  - \(u : FU \triangleleft U \to v\)

**Monoidal category** \((\mathcal{C}, \otimes, I)\)

**String diagrams**

\[
\begin{align*}
A \xrightarrow{f} B & \quad B \xrightarrow{g} C \\
A \xrightarrow{g \circ f} C
\end{align*}
\]

\[
\begin{align*}
A \xrightarrow{f} B & \quad C \xrightarrow{g} D \\
A \otimes C \xrightarrow{f \otimes g} B \otimes D
\end{align*}
\]

\[
h \circ (f \otimes g)
\]
**GoI situation**

**Defn.** (GoI situation [AHS02])

A GoI situation is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \to C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF \triangleleft F : e'\) Comultiplication
  - \(d : \text{id} \triangleleft F : d'\) Dereliction
  - \(c : F \otimes F \triangleleft F : c'\) Contraction
  - \(w : K_I \triangleleft F : w'\) Weakening

Here \(K_I\) is the constant functor into the monoidal unit \(I\);

- \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
  - \(j : U \otimes U \triangleleft U : k\)
  - \(I \triangleleft U\)
  - \(u : FU \triangleleft U : v\)

\[ A \otimes C \xrightarrow{f} B \otimes C \]

\[ A \xrightarrow{\text{tr}(f)} B \]

that is

\[ \begin{array}{ccc}
A & C & f \\
B & C & \xrightarrow{\text{tr}(f)} B
\end{array} \]
In this talk, I use two ways of depicting partial functions $\mathbb{N} \rightarrow \mathbb{N}$.
In this talk, I use two ways of depicting partial functions $\mathbb{N} \rightarrow \mathbb{N}$.

In the monoidal category $(\text{Pfn}, +, 0)$.
Traced Sym. Monoidal Category
(Pfn, +, 0)

* Category Pfn of partial functions
  - Obj. A set $X$
  - Arr. A partial function

\[
\begin{array}{c}
X \rightarrow Y \text{ in Pfn} \\
\hline
X \leftarrow Y, \text{ partial function}
\end{array}
\]
Traced Sym. Monoidal Category
(Pfn, +, 0)

* Category Pfn of partial functions

* Obj. A set $X$

* Arr. A partial function $X \rightarrow Y$ in Pfn

* is traced symmetric monoidal
Traced Sym. Monoidal Category 
(Pfn, +, 0)

* Given $X + Z \xrightarrow{f} Y + Z$ in Pfn
Traced Sym. Monoidal Category 
\((Pfn, +, 0)\)

* Given \(X + Z \xrightarrow{f} Y + Z\) in \(Pfn\)

\[
\begin{array}{c}
X \\
\hline
Y \\
\hline
\end{array}
\begin{array}{c}
Z \\
\hline
f \\
\hline
f \\
\hline
\end{array}
\begin{array}{c}
Y \\
\hline
Z \\
\hline
\end{array}
\]
Traced Sym. Monoidal Category

\((\text{Pfn}, +, 0)\)

* Given

\[ X + Z \xrightarrow{f} Y + Z \text{ in } \text{Pfn} \]
Traced Sym. Monoidal Category

\((Pfn, +, 0)\)

* Given

\[ X + Z \xrightarrow{f} Y + Z \text{ in } Pfn \]

\[ f_{XY}(x) := \begin{cases} 
    f(x) & \text{if } f(x) \in Y \\
    \bot & \text{o.w.}
\end{cases} \]

Similar for \(f_{XZ}, f_{ZY}, f_{ZZ}\)
Traced Sym. Monoidal Category
(Pfn, +, 0)

* Given

\[ X + Z \xrightarrow{f} Y + Z \text{ in } \text{Pfn} \]

* Trace operator:

\[ f_{XY}(x) := \begin{cases} 
  f(x) & \text{if } f(x) \in Y \\
  \perp & \text{o.w.} 
\end{cases} \]

Similar for \( f_{XZ}, f_{ZY}, f_{ZZ} \)
Traced Sym. Monoidal Category
(Pfn, +, 0)

* Given
\[ X + Z \xrightarrow{f} Y + Z \text{ in Pfn} \]

* Trace operator:
\[ \text{tr}(f) = \bigoplus_{n \in \mathbb{N}} f_{XY} \circ (f_{ZZ})^n \circ f_{XZ} \]

\[ f_{XY}(x) := \begin{cases} f(x) & \text{if } f(x) \in Y \\ \perp & \text{o.w.} \end{cases} \]

Similar for \( f_{XZ}, f_{ZY}, f_{ZZ} \)
Traced Sym. Monoidal Category
\((Pfn, +, 0)\)

* Given \(X + Z \xrightarrow{f} Y + Z\) in \(Pfn\)

\[
f_{XY}(x) := \begin{cases} f(x) & \text{if } f(x) \in Y \\ \bot & \text{o.w.} \end{cases}
\]
Similar for \(f_{XZ}, f_{ZY}, f_{ZZ}\)

* Trace operator:

\[
\text{tr}(f) = f_{XY} \sqcup \left( \bigcup_{n \in \mathbb{N}} f_{ZY} \circ (f_{ZZ})^n \circ f_{XZ} \right)
\]

* Execution formula

Partiality is essential (infinite loop)
**GoI situation**

* **Traced sym. monoidal cat.**

* **Where one can “feedback”**

* **Why for GoI?**

---

**Defn. (GoI situation [AHS02])**

A GoI situation is a triple \((C, F, U)\) where

1. \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
2. \(F : C \to C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
   - \(e : FF \triangleleft F \to F'\) : Comultiplication
   - \(d : \text{id} \triangleleft F \to d'\) : Dereliction
   - \(c : F \otimes F \triangleleft F \to c'\) : Contraction
   - \(w : K_I \triangleleft F \to w'\) : Weakening

Here \(K_I\) is the constant functor into the monoidal unit \(I\);

3. \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
   - \(j : U \otimes U \triangleleft U \to k\)
   - \(I \triangleleft U\)
   - \(u : FU \triangleleft U \to v\)
\[ [M \times N] = \text{in string diagram} \]
**GoI situation**

* Traced sym. monoidal cat.
* Where one can “feedback”

**Defn.** (GoI situation [AHS02])

A GoI situation is a triple \((\mathcal{C}, F, U)\) where

- \(\mathcal{C} = (\mathcal{C}, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : \mathcal{C} \to \mathcal{C}\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF \otimes F \to F : e'\) Comultiplication
  - \(d : \text{id} \otimes F \to F : d'\) Dereliction
  - \(c : F \otimes F \otimes F \to F : c'\) Contraction
  - \(w : K_I \otimes F \to F : w'\) Weakening

Here \(K_I\) is the constant functor into the monoidal unit \(I\);
- \(U \in \mathcal{C}\) is an object (called reflexive object), equipped with the following retractions.
  - \(j : U \otimes U \otimes U \to U : k\)
  - \(I \otimes U\)
  - \(u : FU \otimes U \to U : v\)

**Why for GoI?**

\[
\begin{array}{c}
\begin{array}{c}
M \\
\circlearrowleft
\end{array}
\end{array}
= \begin{array}{c}
\begin{array}{c}
M \\
\circlearrowleft
\end{array}
\end{array} = \text{tr}(\begin{array}{c}
\begin{array}{c}
M \\
\circlearrowleft
\end{array}
\end{array})
\]

**Leading example: Pfn**
**GoI situation**

**Defn.** (GoI situation [AHS02])
A GoI situation is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \rightarrow C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF \triangleleft F : e'\) Comultiplication
  - \(d : \text{id} \triangleleft F : d'\) Dereliction
  - \(c : F \otimes F \triangleleft F : c'\) Contraction
  - \(w : K_I \triangleleft F : w'\) Weakening

Here \(K_I\) is the constant functor into the monoidal unit \(I\):

- \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
  - \(j : U \otimes U \triangleleft U : k\)
  - \(I \triangleleft U\)
  - \(u : FU \triangleleft U : v\)

**Defn.** (Retraction)
A retraction from \(X\) to \(Y\),

\[ f : X \triangleleft Y : g, \]

is a pair of arrows

\[ \text{id} \arl{f} X \arl{g} Y \]

such that \(g \circ f = \text{id}_X\).

\[ \ast \] **Functor** \(F\)

\[ \ast \] For obtaining \(! : A \rightarrow A\)
GoI situation

Defn. (GoI situation [AHS02])
A GoI situation is a triple \((\mathcal{C}, F, U)\) where

- \(\mathcal{C} = (\mathcal{C}, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : \mathcal{C} \to \mathcal{C}\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF \bowtie F : e'\) Comultiplication
  - \(d : \text{id} \bowtie F : d'\) Dereliction
  - \(c : F \otimes F \bowtie F : c'\) Contraction
  - \(w : K_I \bowtie F : w'\) Weakening
Here \(K_I\) is the constant functor into the monoidal unit \(I\);
- \(U \in \mathcal{C}\) is an object (called reflexive object), equipped with the following retractions.
  - \(j : U \otimes U \bowtie U : k\)
  - \(I \bowtie U\)
  - \(u : FU \bowtie U : v\)

* Functor \(F\)

* For obtaining \(!: A \to A\)

* Pictorially:
**GoI situation**

*Functor $F$*

*For obtaining $!: A \rightarrow A$*

*Pictorially:*

*Example in Pfn:*

\[
\begin{array}{ccc}
\mathbb{N} \cdot _{-} & : & \text{Pfn} \longrightarrow \text{Pfn} \\
\downarrow f & & \downarrow \mathbb{N} \cdot f \\
X & & \mathbb{N} \cdot X \\
\end{array}
\]

---

**Defn.** (GoI situation [AHS02])

A GoI situation is a triple $(C, F, U)$ where

- $C = (C, \otimes, I)$ is a traced symmetric monoidal category (TSMC);
- $F : C \rightarrow C$ is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - $e : FF < F : e'$: Comultiplication
  - $d : \text{id} < F : d'$: Dereliction
  - $c : F \otimes F < F : c'$: Contraction
  - $w : K_I < F : w'$: Weakening

Here $K_I$ is the constant functor into the monoidal unit $I$;

- $U \in C$ is an object (called reflexive object), equipped with the following retractions.
  - $j : U \otimes U < U : k$
  - $I < U$
  - $u : FU < U : v$
**GoI situation**

**Defn.** (GoI situation [AHS02])

A GoI situation is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \to C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF \triangleleft F : e'\) Comultiplication
  - \(d : \text{id} \triangleleft F : d'\) Dereliction
  - \(c : F \otimes F \triangleleft F : c'\) Contraction
  - \(w : K_I \triangleleft F : w'\) Weakening

Here \(K_I\) is the constant functor into the monoidal unit \(I\):

- \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
  - \(j : U \otimes U \triangleleft U : k\)
  - \(I \triangleleft U\)
  - \(u : FU \triangleleft U : v\)

**The reflexive object** \(U\)

**Retr.** \(U \otimes U \xrightarrow{j} U \xleftarrow{k}\)

**Retr.** \(FU \xrightarrow{u} U \xleftarrow{v}\)
**GoI situation**

**Defn.** (GoI situation [AHS02])

A *GoI situation* is a triple $(\mathcal{C}, F, U)$ where

- $\mathcal{C} = (\mathcal{C}, \otimes, I)$ is a traced symmetric monoidal category (TSMC);
- $F: \mathcal{C} \to \mathcal{C}$ is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - $e: FF \triangleleft F : e'$ — Comultiplication
  - $d: \text{id} \triangleleft F : d'$ — Dereliction
  - $c: F \otimes F \triangleleft F : c'$ — Contraction
  - $w: K_I \triangleleft F : w'$ — Weakening

Here $K_I$ is the constant functor into the monoidal unit $I$;
- $U \in \mathcal{C}$ is an object (called *reflexive object*), equipped with the following retractions.
  - $j: U \otimes U \triangleleft U : k$
  - $u: FU \triangleleft U : v$

**The reflexive object $U$**

**Retr.** $U \otimes U \overset{j}{\underset{k}{\leftrightarrow}} U$

Here $K_I$ is the constant functor into the monoidal unit $I$.

**Retr.** $FU \overset{u}{\underset{v}{\leftrightarrow}} U$
Defn. (GoI situation [AHS02])
A GoI situation is a triple \((\mathcal{C}, F, U)\) where

- \(\mathcal{C} = (\mathcal{C}, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : \mathcal{C} \to \mathcal{C}\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  
  - \(e : FF \triangleleft F : e'\) Comultiplication
  - \(d : \text{id} \triangleleft F : d'\) Dereliction
  - \(c : F \otimes F \triangleleft F : c'\) Contraction
  - \(w : K_I \triangleleft F : w'\) Weakening

Here \(K_I\) is the constant functor into the monoidal unit \(I\):

- \(U \in \mathcal{C}\) is an object (called reflexive object), equipped with the following retractions.
  
  - \(j : U \otimes U \triangleleft U : k\)
  - \(I \triangleleft U\)
  - \(u : FU \triangleleft U : v\)

The reflexive object \(U\) with

- \(j \quad k\)

\(= \text{id}\)

Retr.

\[FU \quad U\]

\[u \quad v\]
**GoI situation**

* The reflexive object $U$

* Retr. $U \otimes U \xrightarrow{j} U$ with

$$j, k = \text{id}$$

* Retr. $FU \xrightarrow{u} U$ with

Hasuo (Tokyo)

---

**Defn. (GoI situation [AHS02])**

A *GoI situation* is a triple $(C, F, U)$ where

- $C = (C, \otimes, I)$ is a traced symmetric monoidal category (TSMC);
- $F : C \to C$ is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - $e : FF \lessdot F : e'$ Commultiplication
  - $d : \text{id} \lessdot F : d'$ Dereliction
  - $c : F \otimes F \lessdot F : c'$ Contraction
  - $w : K_I \lessdot F : w'$ Weakening

Here $K_I$ is the constant functor into the monoidal unit $I$.

- $U \in C$ is an object (called *reflexive object*), equipped with the following retractions.
  - $j : U \otimes U \lessdot U : k$
  - $I \lessdot U$
  - $u : FU \lessdot U : v$
**GoI situation**

* The reflexive object $U$

* Why for GoI?

---

**Defn.** (GoI situation [AHS02])

A *GoI situation* is a triple $(C, F, U)$ where:

1. $C = (C, \otimes, I)$ is a traced symmetric monoidal category (TSMC);
2. $F : C \to C$ is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
   - $e : FF \otimes F : e'$
     - *Comultiplication*
   - $d : id \otimes F : d'$
   - $c : F \otimes F \otimes F : c'$
   - $w : K_I \otimes F : w'$

   Here $K_I$ is the constant functor.
3. $U \in C$ is an object (called *reflexive object*), equipped with the following retractions.
   - $j : U \otimes U \otimes U : k$
   - $I \otimes U$
   - $u : FU \otimes U : v$

* Example in Pfn:

---

Hasuo (Tokyo)
GoI situation

Defn. (GoI situation [AHS02])
A GoI situation is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \rightarrow C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  
  \[
  \begin{align*}
  e : FF &\ll F : e' \\
  d : \text{id} &\ll F : d' \\
  c : F &\otimes F \ll F : c' \\
  w : K_I &\ll F : w'
  \end{align*}
  \]
  
  Here \(K_I\) is the constant functor.
- \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
  
  \[
  \begin{align*}
  j : U &\otimes U \ll U : k \\
  I &\ll U \\
  u : FU &\ll U : v
  \end{align*}
  \]

* The reflexive object \(U\)

* Why for GoI?

* Example in Pfn:

Hasuo (Tokyo)

Sunday, September 11, 2011
GoI situation

Defn. (GoI situation [AHS02])
A GoI situation is a triple \((C, F, U)\) where

1. \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
2. \(F : C \to C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
   - \(e : FF \triangleleft F \to e'\) (Comultiplication)
   - \(d : \text{id} \triangleleft F \to d'\)
   - \(c : F \otimes F \triangleleft F \to c'\)
   - \(w : K_I \triangleleft F \to w'\)

   Here \(K_I\) is the constant functor.
3. \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
   - \(j : U \otimes U \triangleleft U \to k\)
   - \(I \triangleleft U\)
   - \(u : FU \triangleleft U \to v\)

Why for GoI?

Example in \(\text{Pfn}\):
\[N \in \text{Pfn}, \text{ with } N + N \cong N, \quad N \cdot N \cong N\]
**GoI Situation: Summary**

* Categorical axiomatics of the “GoI animation”

**Defn.** (GoI situation [AHS02])

A *GoI situation* is a triple $(C, F, U)$ where

- $C = (C, \otimes, I)$ is a traced symmetric monoidal category (TSMC);
- $F : C \to C$ is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).

\[ e : FF \triangleleft F : e' \quad \text{Comultiplication} \]
\[ d : \text{id} \triangleleft F : d' \quad \text{Dereliction} \]
\[ c : F \otimes F \triangleleft F : c' \quad \text{Contraction} \]
\[ w : K_I \triangleleft F : w' \quad \text{Weakening} \]

Here $K_I$ is the constant functor into the monoidal unit $I$;

- $U \in C$ is an object (called *reflexive object*), equipped with the following retractions.

\[ j : U \otimes U \triangleleft U : k \]
\[ I \triangleleft U \]
\[ u : FU \triangleleft U : v \]

**Example:**

$\mathcal{M} \downarrow \quad \mathcal{N}$

(Pfn, $\mathcal{N} \cdot \_ \cdot \mathcal{N}$)
**Categorical axiomatics of the “GoI animation”**

**Example:**

\[ \text{GoI Situation: Summary} \]

**Defn. (GoI situation [AHS02])**

A GoI situation is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \rightarrow C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF < F : e'\) Comultiplication
  - \(d : \text{id} < F : d'\) Dereliction
  - \(c : F \otimes F < F : c'\) Contraction
  - \(w : K_I < F : w'\) Weakening

Here \(K_I\) is the constant functor into the monoidal unit \(I\);

- \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
  - \(j : U \otimes U < U : k\)
  - \(I < U\)
  - \(u : FU < U : v\)
**Categorical axiomatics of the “GoI animation”**

**Example:**

\[(\text{Pfn}, N \cdot _, N)\]
A GoI situation is a triple \((C, F, U)\) where:

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \to C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations):
  
  \[ 
  \begin{align*}
  e : FF \triangleright F &\triangleleft e' \\
  d : \text{id} \triangleleft F &\triangleright d' \\
  c : F \otimes F \triangleright F &\triangleleft c' \\
  w : K_I \triangleleft F &\triangleright w'
  \end{align*} \]

Here \(K_I\) is the constant functor into the monoidal unit \(I\);
- \(U \in C\) is an object (called reflexive object), equipped with the following retractions:

\[ 
\begin{align*}
  j : U \otimes U \triangleright U &\triangleleft k \\
  I &\triangleright U \\
  u : FU \triangleright U &\triangleleft v
\end{align*} \]

\(\triangleright\) and \(\triangleleft\) are the retractions:

\[ 
\begin{align*}
  f \triangleright \triangleright &\rightarrow f \\
  f &\triangleleft \triangleleft
\end{align*} \]

\(\triangleright\) and \(\triangleleft\) are the retractions:

\[ 
\begin{align*}
  f \triangleright \triangleright &\rightarrow f \\
  f &\triangleleft \triangleleft
\end{align*} \]

\(\triangleright\) and \(\triangleleft\) are the retractions:

\[ 
\begin{align*}
  f \triangleright \triangleright &\rightarrow f \\
  f &\triangleleft \triangleleft
\end{align*} \]
**Categorical axiomatics of the “GoI animation”**

**Example:**

\[(\text{Pfn}, \, \mathbb{N} \cdot \_ \, , \, \mathbb{N})\]


**Situation: Summary**

* Categorical axiomatics of the “GoI animation”

**Defn.** (GoI situation [AHS02])

A **GoI situation** is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \to C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations):
  - \(e : FF \otimes F : e'\)
  - \(d : \text{id} \otimes F : d'\)
  - \(c : F \otimes F \otimes F : c'\)
  - \(w : K_I \otimes F : w'\)
- \(U \in C\) is an object (called **reflexive object**), equipped with the following retractions:
  - \(j : U \otimes U \otimes U : k\)
  - \(I \otimes U : k\)
  - \(u : FU \otimes U : v\)

Here \(K_I\) is the constant functor into the monoidal unit \(I\).

**Example:**

\((\text{Pfn}, N \cdot _{-}, N)\)
The Categorical GoI Workflow

Traced monoidal category $\mathcal{C}$
+ other constructs $\Rightarrow$ “GoI situation” [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

Model of typed calculus

!* Applicative str. + combinators
!* Model of untyped calculus
The Categorical GoI Workflow

Traced monoidal category $\mathcal{C}$
+ other constructs $\rightarrow$ "GoI situation" [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

Model of typed calculus

* Applicative str. + combinators
* Model of untyped calculus
Categorical GoI: Constr. of an LCA

**Thm. ([AHS02])**
Given a GoI situation \((\mathcal{C}, F, U)\), the homset 
\[
\mathcal{C}(U, U)
\]
carries a canonical LCA structure.
Categorical GoI: Constr. of an LCA

**Thm.** ([AHS02])
Given a GoI situation \((\mathcal{C}, F, U)\), the homset
\[
\mathcal{C}(U, U)
\]
carries a canonical LCA structure.

- Applicative str. ·
- ! operator
- Combinators B, C, I, ...
**Thm.** ([AHS02])

Given a GoI situation \((\mathcal{C}, F, U)\), the homset

\[ \mathcal{C}(U, U) \]

carries a canonical LCA structure.

- Applicative str. \(\cdot\)
- ! operator
- Combinators B, C, I, ...
Categorical GoI: Constr. of an LCA

Thm. ([AHS02])
Given a GoI situation \((\mathcal{C}, F, U)\), the homset \(\mathcal{C}(U, U)\) carries a canonical LCA structure.

- Applicative str. \(\cdot\)
- ! operator
- Combinators B, C, I, ...

\[ f : \mathcal{C}(U, U) \in \mathcal{C}(U, U) \]

\[ g \cdot f := \text{tr}( (U \otimes f) \circ k \circ g \circ j ) \]

Hasuo (Tokyo)
Thm. ([AHS02])
Given a GoI situation \((\mathcal{C}, F, U)\), the homset
\[
\mathcal{C}(U, U)
\]
carries a canonical LCA structure.

\[
!f := u \circ Ff \circ v
\]

- Applicative str.
- \(!\) operator
- Combinators B, C, I, ...
Categorical GoI: Constr. of an LCA

* Combinator $B_{xyz} = x(yz)$

Figure 7: Composition Combinator B

from [AHS02]

Hasuo (Tokyo)
Categorical GoI: Constr. of an LCA

* **Combinator** $B x y z = x(yz)$
Hasuo (Tokyo)

Categorical GoI:

Constr. of an LCA

\[ B \] = \[ x \](\[ yz \])
Constr. of an LCA

Combinator

\[ B_{xyz} = x_{yz} \]
Hasuo (Tokyo)

Categorical GoI:

Constr. of an LCA

B_{xyz} = x (yz)

Sunday, September 11, 2011
Categorical GoI: Constr. of an LCA

* Combinator $B_{xyz} = x(yz)$
Categorical GoI: Constr. of an LCA

* Combinator $B_{xyz} = x(yz)$

Figure 7: Composition Combinator B

Nice dynamic interpretation of (linear) computation!!
Summary: Categorical GoI

**Defn.** (GoI situation [AHS02])
A GoI situation is a triple \((\mathcal{C}, F, U)\) where

- \(\mathcal{C} = (\mathcal{C}, \otimes, I)\) is a **traced symmetric monoidal category** (TSMC);
- \(F : \mathcal{C} \to \mathcal{C}\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  
  \[
  \begin{align*}
  e : FF \triangleleft F & : e' \\
  d : \text{id} \triangleleft F & : d' \\
  c : F \otimes F \triangleleft F & : c' \\
  w : K_I \triangleleft F & : w'
  \end{align*}
  \]
  
  Comultiplication
  
  Dereliction
  
  Contraction
  
  Weakening

  Here \(K_I\) is the constant functor into the monoidal unit \(I\);

- \(U \in \mathcal{C}\) is an object (called **reflexive object**), equipped with the following retractions.

  \[
  \begin{align*}
  j : U \otimes U \triangleleft U & : k \\
  I & \triangleleft U \\
  u : FU \triangleleft U & : v
  \end{align*}
  \]

**Thm.** ([AHS02])
Given a GoI situation \((\mathcal{C}, F, U)\), the homset \(\mathcal{C}(U, U)\) carries a canonical LCA structure.
Why Categorical Generalization?: Examples Other Than Pfn

- Strategy: find a TSMC!

- “Wave-style” examples
  - $\otimes$ is Cartesian product(-like)
  - in which case,
    \[
    \text{trace} \approx \text{fixed point operator} \quad [\text{Hasegawa/Hyland}]
    \]

- An example: \(( (\omega-Cpo, \times, 1), (\_)^N, A^N )\)

- (... less of a dynamic flavor)
**Why Categorical Generalization?: Examples Other Than Pfn**

- "Particle-style" examples
  - Obj. $X \in C$ is set-like; $\otimes$ is coproduct-like
  - The GoI animation is valid

- Examples:
  - Partial functions
  - Non-det. functions (i.e. relations)
  - Probabilistic functions ("discrete stochastic relations")

\[ ((\text{Pfn}, +, 0), \mathbb{N} \cdot (_) \cdot \mathbb{N}) \]
\[ ((\text{Rel}, +, 0), \mathbb{N} \cdot (_) \cdot \mathbb{N}) \]
\[ ((\text{DSRel}, +, 0), \mathbb{N} \cdot (_) \cdot \mathbb{N}) \]
Why Categorical Generalization?:
Examples Other Than Pfn

* **Pfn** (partial functions)

\[ X \rightarrow Y \] in Pfn
\[ X \rightarrow Y, \text{ partial function} \]
\[ X \rightarrow \mathcal{L}Y \] in Sets

\[ \mathcal{L}Y = \{ \bot \} + Y \]

* **Rel** (relations)

\[ X \rightarrow Y \] in Rel
\[ R \subseteq X \times Y, \text{ relation} \]
\[ X \rightarrow \mathcal{P}Y \] in Sets

where \( \mathcal{P} \) is the powerset monad

* **DSRel**

\[ X \rightarrow Y \] in DSRel
\[ X \rightarrow \mathcal{D}Y \] in Sets

where \( \mathcal{D}Y = \{ d : Y \rightarrow [0, 1] \mid \sum_y d(y) \leq 1 \} \)
Why Categorical Generalization?
Examples Other Than Pfn

* **Pfn** (partial functions)

\[
\begin{align*}
X & \to Y \text{ in } \text{Pfn} \\
X & \to Y, \text{ partial function} \\
X & \to \mathcal{L}Y \text{ in } \text{Sets} \\
\end{align*}
\]

where \( \mathcal{L}Y = \{ \bot \} + Y \)

* **Rel** (relations)

\[
\begin{align*}
X & \to Y \text{ in } \text{Rel} \\
R & \subseteq X \times Y, \text{ relation} \\
X & \to \mathcal{P}Y \text{ in } \text{Sets} \\
\end{align*}
\]

where \( \mathcal{P} \) is the powerset monad

* **DSRel**

\[
\begin{align*}
X & \to Y \text{ in } \text{DSRel} \\
X & \to \mathcal{D}Y \text{ in } \text{Sets} \\
\text{where } \mathcal{D}Y = \{ d : Y \to [0, 1] \mid \sum_y d(y) \leq 1 \} \\
\end{align*}
\]
Why Categorical Generalization?

Examples Other Than Pfn

* **Pfn** (partial functions)

\[ X \to Y \text{ in } \text{Pfn} \]
\[ \frac{X \to Y, \text{ partial function}}{X \to \mathcal{L}Y \text{ in } \text{Sets}} \]
where \( \mathcal{L}Y = \{ \bot \} + Y \)

* **Rel** (relations)

\[ X \to Y \text{ in } \text{Rel} \]
\[ \frac{R \subseteq X \times Y, \text{ relation}}{X \to \mathcal{P}Y \text{ in } \text{Sets}} \]
where \( \mathcal{P} \) is the powerset monad

* **DSRel**

\[ X \to Y \text{ in } \text{DSRel} \]
\[ X \to \mathcal{D}Y \text{ in } \text{Sets} \]
where \( \mathcal{D}Y = \{ d : Y \to [0, 1] \mid \sum_y d(y) \leq 1 \} \)

Categories of sets and (functions with different branching/partiality)

(Potential) non-termination

Non-determinism

Probabilistic branching
Different Branching in The GoI Animation

- **Pfn** (partial functions)
- Pipe can be stuck
- **Rel** (relations)
- Pipe can branch
- **DSRel**
- Pipe can branch probabilistically
Different Branching in The GoI Animation

- Pfn (partial functions)
- Pipe can be stuck
- Rel (relations)
- Pipe can branch
- DSRel
- Pipe can branch probabilistically
Different Branching in The GoI Animation

- **Pfn** (partial functions)
  - Pipe can be stuck
- **Rel** (relations)
  - Pipe can branch
- **DSRel**
  - Pipe can branch probabilistically
Different Branching in The GoI Animation

- **Pfn** (partial functions)
  - Pipe can be stuck

- **Rel** (relations)
  - Pipe can branch

- **DSRel**
  - Pipe can branch probabilistically
Different Branching in The GoI Animation

- **Pfn** (partial functions)
- **Pipe can be stuck**
- **Rel** (relations)
- **Pipe can branch**
- **DSRel**
- **Pipe can branch probabilistically**

Hasuo (Tokyo)

Sunday, September 11, 2011
Different Branching in The GoI Animation

- Pfn (partial functions)
- Pipe can be stuck
- Rel (relations)
- Pipe can branch
- DSRel
- Pipe can branch probabilistically
Different Branching in The GoI Animation

- \textbf{Pfn} (partial functions)
- Pipe can be stuck
- \textbf{Rel} (relations)
- Pipe can branch
- \textbf{DSRel}
- Pipe can branch probabilistically

Hasuo (Tokyo)
Why Categorical Generalization?: Examples Other Than Pfn

* **Pfn (partial functions)**
  \[ X \to Y \text{ in } \text{Pfn} \]
  \[ X \to Y, \text{ partial function} \]
  \[ X \to \mathcal{L}Y \text{ in } \text{Sets} \]
  where \( \mathcal{L}Y = \{ \perp \} + Y \)

* **Rel (relations)**
  \[ X \to Y \text{ in } \text{Rel} \]
  \[ R \subseteq X \times Y, \text{ relation} \]
  \[ X \to \mathcal{P}Y \text{ in } \text{Sets} \]
  where \( \mathcal{P} \) is the powerset monad

* **DSRel**
  \[ X \to Y \text{ in } \text{DSRel} \]
  \[ X \to \mathcal{D}Y \text{ in } \text{Sets} \]
  where \( \mathcal{D}Y = \{ d : Y \to [0, 1] \mid \sum_y d(y) \leq 1 \} \)
Why Categorical Generalization?: Examples Other Than Pfn

* Pfn (partial functions)

\[
\frac{X \rightarrow Y \text{ in Pfn}}{X \rightarrow Y, \text{ partial function}} \quad \text{where } \mathcal{L}Y = \{\bot\} + Y
\]

\[
\frac{X \rightarrow \mathcal{L}Y \text{ in Sets}}{X \rightarrow Y \text{ in Pfn}}
\]

* Rel (relations)

\[
\frac{X \rightarrow Y \text{ in Rel}}{R \subseteq X \times Y, \text{ relation}} \quad \text{where } \mathcal{P} \text{ is the powerset monad}
\]

\[
\frac{X \rightarrow \mathcal{P}Y \text{ in Sets}}{X \rightarrow Y \text{ in Rel}}
\]

* DSRel

\[
\frac{X \rightarrow Y \text{ in DSRel}}{X \rightarrow \mathcal{D}Y \text{ in Sets}}
\]

where \( \mathcal{D}Y = \{d : Y \rightarrow [0, 1] \mid \sum_y d(y) \leq 1\} \)

Essential to have subdistribution, for infinite loops
The Coauthor

Naohiko Hoshino

DSc

Kyoto U. (JP), 2011

Supervisor:
Masahito “Hassei” Hasegawa

Assist. Prof.,
RIMS, Kyoto U. (2011–)
A Coalgebraic View

* Theory of coalgebra = Categorical theory of state-based dynamic systems (LTS, automaton, Markov chain, ...)

* In my thesis (2008):
  * Coalgebras in a Kleisli category $Kl(B)$

  $X \rightarrow Y$ in $Kl(B)$

  $\frac{X \rightarrow BY}{X \rightarrow BY}$ in $Sets$

* $\Rightarrow$ Generic theory of “trace semantics”
Why Categorical Generalization?
Examples Other Than Pfn

* **Pfn** (partial functions)

\[
\frac{X \to Y \text{ in Pfn}}{X \to \mathcal{L}Y \text{ in Sets}}
\]

where \( \mathcal{L}Y = \{\bot\} + Y \)

* **Rel** (relations)

\[
\frac{X \to Y \text{ in Rel}}{R \subseteq X \times Y, \text{ relation}}
\]

where \( \mathcal{P} \) is the powerset monad

\[
\frac{X \to \mathcal{P}Y \text{ in Sets}}{}
\]

* **DSRel**

\[
\frac{X \to Y \text{ in DSRel}}{X \to \mathcal{D}Y \text{ in Sets}}
\]

where \( \mathcal{D}Y = \{d : Y \to [0, 1] \mid \sum_y d(y) \leq 1\} \)

Potentials

- Non-termination
- Non-determinism
- Probabilistic branching

Categories of sets and (functions with different branching/partiality)
Why Categorical Generalization?
Examples Other Than Pfn

* **Pfn (partial functions)**

\[
\begin{align*}
X \to Y & \text{ in Pfn} \\
X \to Y, \text{ partial function} & \Rightarrow X \to \mathcal{L}Y \text{ in Sets} \\
\text{where } \mathcal{L}Y &= \{\bot\} + Y
\end{align*}
\]

* **Rel (relations)**

\[
\begin{align*}
X \to Y & \text{ in Rel} \\
R \subseteq X \times Y, \text{ relation} & \Rightarrow X \to \mathcal{P}Y \text{ in Sets} \\
\text{where } \mathcal{P} & \text{ is the powerset monad}
\end{align*}
\]

* **DSRel**

\[
\begin{align*}
X \to Y & \text{ in DSRel} \\
X \to \mathcal{D}Y \text{ in Sets} & \Rightarrow \text{where } \mathcal{D}Y = \{d : Y \to [0, 1] \mid \sum_y d(y) \leq 1\}
\end{align*}
\]
Thm. ([Jacobs,CMCS10])
Given a “branching monad” $B$ on $\text{Sets}$, the monoidal category

$$(\mathcal{K}\ell(B), +, 0)$$

is

- a unique decomposition category [Haghverdi,PhD00], hence is
- a traced symmetric monoidal category.

Cor.

$$( (\mathcal{K}\ell(B), +, 0), \mathbb{N}\cdot\underline{\phantom{x}}, \mathbb{N} )$$ is a GoI situation.
Thm. ([Jacobs,CMCS10])
Given a “branching monad” $B$ on $\text{Sets}$, the monoidal category

$$(\mathcal{K}(B), +, 0)$$

is

- a unique decomposition category
  [Haghverdi,PhD00], hence is
- a traced symmetric monoidal category.

Cor.
$$(\mathcal{K}(B), +, 0), \mathbb{N} \cdot \_ , \mathbb{N})$$ is a GoI situation.

Monads in
[Hasuo,Jacobs&Sokolova07]

- $\mathbb{Kl}(B)$ is Cpo⊥-enriched
- like $\mathcal{L}, \mathcal{P}, \mathcal{D}$
Branching Monad: Source of Particle-Style GoI Situations

**Thm.** ([Jacobs, CMCS10])
Given a “branching monad” $B$ on Sets, the monoidal category

$$(\mathcal{K}l(B), +, 0)$$

is

- a *unique decomposition category* [Haghverdi, PhD00], hence is
  - a traced symmetric monoidal category.

**Cor.**

$$(\mathcal{K}l(B), +, 0), N \cdot \_ , N$$ is a GoI situation.

Monads in [Hasuo, Jacobs & Sokolova07]

- $\mathcal{K}l(B)$ is Cpo$_\perp$-enriched
- like $\mathcal{L}, \mathcal{P}, \mathcal{D}$

Particle-style: trace via the execution formula

$$\text{tr}(f) = f_{XY} \sqcup \left( \bigsqcup_{n \in \mathbb{N}} f_{ZY} \circ (f_{ZZ})^n \circ f_{XZ} \right)$$
The Categorical GoI Workflow

Traced monoidal category $\mathcal{C}$
+ other constructs $\rightarrow$ "GoI situation" [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category
The Categorical GoI Workflow

- Branching monad $B$
- Coalgebraic trace semantics
- Traced monoidal category $\mathcal{C}$
  + other constructs $\Rightarrow$ "GoI situation" [AHS02]
- Categorical GoI [AHS02]
- Linear combinatory algebra
- Realizability
- Linear category
The Categorical GoI Workflow

Branching monad B

Coalgebraic trace semantics

Traced monoidal category $\mathcal{C}$
+ other constructs $\rightarrow$ “GoI situation” [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

Model of fancy language
The Categorical GoI Workflow

- Branching monad $B$
- Coalgebraic trace semantics
- Traced monoidal category $\mathcal{C}$
  + other constructs $\Rightarrow$ "GoI situation" [AHS02]
- Categorical GoI [AHS02]
- Linear combinatory algebra
- Realizability
- Linear category

Fancy LCA
Model of fancy language

Hasuo (Tokyo)

Sunday, September 11, 2011
The Categorical GoI Workflow

- Linear category
- Realizability
- Linear combinatory algebra
- Categorical GoI [AHS02]
- Traced monoidal category $\mathbb{C}$
  + other constructs $\Rightarrow$ "GoI situation" [AHS02]
- Branching monad $B$
- Coalgebraic trace semantics

Fancy TSMC
Fancy LCA
Model of fancy language

Hasuo (Tokyo)
The Categorical GoI Workflow

- Linear category
  - Realizability
  - Linear combinatory algebra
    - Categorical GoI [AHS02]
      - Traced monoidal category \( \mathbb{C} \)
        + other constructs \( \rightarrow \) “GoI situation” [AHS02]
          - Coalgebraic trace semantics
            - Branching monad B

- Fancy monad
  - TSMC
    - Fancy
      - LCA
        - Model of fancy language
          - Hasuo (Tokyo)
What is Fancy, Nowadays?
What is Fancy, Nowadays?

* Biology?
What is Fancy, Nowadays?

* Biology?
What is Fancy, Nowadays?

* Biology?

* Hybrid systems?
  * Both discrete and continuous data, typically in cyber-physical systems (CPS)
  * → Our approach via non-standard analysis
    [Suenaga&Hasuo,ICALP11]
What is Fancy, Nowadays?

- Biology?

- Hybrid systems?
  - Both discrete and continuous data, typically in cyber-physical systems (CPS)
  - Our approach via non-standard analysis [Suenaga&Hasuo,ICALP11]

- Quantum?
  - Yes this worked!
The Categorical GoI Workflow

- Traced monoidal category $\mathcal{C}$
  + other constructs → “GoI situation” [AHS02]

- Categorical GoI [AHS02]

- Linear combinatory algebra

- Realizability

- Linear category

- Model of typed calculus

- Model of untyped calculus

- Applicative str. + combinators

Branching monad $B$

Coalgebraic trace semantics

Hasuo (Tokyo)

Sunday, September 11, 2011
The Categorical GoI Workflow

- Categorical GoI \[\text{(AHS02)}\]
- Linear combinatory algebra
- Realizability
- Linear category
- Model of typed calculus
- Model of untyped calculus
- Applicative str. + combinators
- Coalgebraic trace semantics
- Traced monoidal category \(C\) + other constructs \(\rightarrow\) “GoI situation” \[\text{[AHS02]}\]
- Branching monad \(B\)

\[
\begin{array}{c}
A \\
\downarrow \\
B \\
\downarrow \\
C \\
\downarrow \\
A \\
\end{array}
\]

\[
\begin{array}{c}
A \\
\downarrow \\
B \\
\downarrow \\
C \\
\downarrow \\
A \\
\end{array}
\]

\[
\begin{array}{c}
A \\
\downarrow \\
B \\
\downarrow \\
C \\
\downarrow \\
A \\
\end{array}
\]

\[
\begin{array}{c}
A \\
\downarrow \\
B \\
\downarrow \\
C \\
\downarrow \\
A \\
\end{array}
\]

Hasuo (Tokyo)
Part 2

Realizability: from Untyped to Typed
Realizability

* Dates back to Kleene

* Cf. the Brouwer–Heyting–Kolmogorov (BHK) interpretation

* A p’f of $A \land B$ is a pair: $(\text{p’f of } A, \text{p’f of } B)$

* A p’f of $A \rightarrow B$ is a function carrying $(\text{p’f of } A)$ to $(\text{p’f of } B)$

* Proof = “realizer”
Realizability

- Our technical view on realizability: a construction
  - from a combinatorial algebra,
  - of a categorical model of a typed calculus

Here: construct a linear category from an LCA

References:

Realizability

* Either by \( \omega \)-sets (intuitive) or by PERs (tech. convenient)

Defn.
Given an LCA \( A \), an \( \omega \)-set is a pair

\[
( S, \ r : S \to P_+(A) )
\]

where

- \( S \) is a set;
- for each \( x \in S \), the nonempty subset \( r(x) \subseteq A \) is the set of realizers.
Realizability

* Either by $\omega$-sets (intuitive) or by PERs (tech. convenient)

Defn. Given an LCA $A$, an $\omega$-set is a pair

$$(S, r : S \rightarrow \mathcal{P}_+(A))$$

where

- $S$ is a set;
- for each $x \in S$, the nonempty subset $r(x) \subseteq A$ is the set of realizers.

Could as well be a partial combinatory algebra. Its examples:

* $\mathbb{N}$ with $n \cdot m = \text{comp}(n,m)$
* $\{ \text{closed } \lambda\text{-terms} \}$
Realizability

* Either by \( \omega \)-sets (intuitive) or by PERs (tech. convenient)

**Defn.**
Given an LCA \( A \), an \( \omega \)-set is a pair

\[
(S, \quad r : S \to \mathcal{P}_+(A))
\]

where

- \( S \) is a set;

- for each \( x \in S \), the nonempty subset \( r(x) \subseteq A \) is the set of realizers.

Could as well be a partial combinatory algebra. Its examples:

* \( \mathbb{N} \) with \( n \cdot m = \text{comp}(n,m) \)
* \{ closed \( \lambda \)-terms \}

\( a \in r(x) : \)

* "realizes" \( x \), or
* "witnesses existence of" \( x \)
Defn.
A partial equivalence relation (PER) $X$ is a transitive and symmetric relation on $A$.

$$|X| := \{a \mid (a, a) \in X\}$$

$$= \{a \mid \exists b. (a, b) \in X\}$$

$$= \{a \mid \exists b. (b, a) \in X\}$$

is the domain of $X$. 

Realizability
Realizability

Defn.

A partial equivalence relation (PER) $X$ is a transitive and symmetric relation on $A$.

\[ |X| := \{a \mid (a, a) \in X\} \]
\[ = \{a \mid \exists b. (a, b) \in X\} \]
\[ = \{a \mid \exists b. (b, a) \in X\} \]

is the domain of $X$. 

$\ast$ PER = eq. rel. − refl.
Realizability

Defn. A partial equivalence relation (PER) $X$ is a transitive and symmetric relation on $A$.

$$|X| := \{a \mid (a, a) \in X\}$$
$$= \{a \mid \exists b. (a, b) \in X\}$$
$$= \{a \mid \exists b. (b, a) \in X\}$$

is the domain of $X$.

• $\text{PER} = \text{eq. rel. - refl.}$
• An eq. rel. when restricted to $|X|$
Realizability

**Defn.**
A partial equivalence relation (PER) \( X \) is a transitive and symmetric relation on \( A \).

\[ |X| := \{ a | (a, a) \in X \} \]
\[ = \{ a | \exists b. (a, b) \in X \} \]
\[ = \{ a | \exists b. (b, a) \in X \} \]

is the domain of \( X \).

- **PER = eq. rel. - refl.**
- **An eq. rel. when restricted to \( |X| \)**
- **PER to \( \omega \)-set:**

\[
\left( \frac{|X|}{X}, \frac{|X|}{X} \xrightarrow{r} \mathcal{P}_+(A) \right)
\]

with \([a] \xrightarrow{r} \{ b | (a, b) \in X \}\)
Realizability

Defn.
A partial equivalence relation (PER) $X$ is a transitive and symmetric relation on $A$.

$|X| := \{a \mid (a, a) \in X\}$

$= \{a \mid \exists b. (a, b) \in X\}$

$= \{a \mid \exists b. (b, a) \in X\}$

is the domain of $X$.

PER = eq. rel. − refl.

An eq. rel. when restricted to $|X|$

PER to $\omega$-set:

$\left( |X|/X, \quad |X|/X \xrightarrow{r} \mathcal{P}_+(A) \right)$

with $[a] \xrightarrow{r} \{b \mid (a, b) \in X\}$

Also: $\omega$-set to PER

Hasuo (Tokyo)
PER$_A$:
The Category of PERs

* **Obj.** A PER $X$ on $A$

* **Arr.** The homset is

$$\text{PER}_A(X, Y)$$
$$= \left\{ c \in A \mid (x, x') \in X \implies (cx, cx') \in Y \right\}$$

Thus:

* Often put:
**PER$_A$: The Category of PERs**

* **Obj.** A PER $X$ on $A$

* **Arr.** The homset is

\[
\text{PER}_A(X, Y) = \left\{ c \in A \mid (x, x') \in X \implies (cx, cx') \in Y \right\}
\]

Thus:

* Often put:
**PER}_A: The Category of \textit{PERs}

* **Obj.** A \textit{PER} \(X\) on \(A\)

* **Arr.** The homset is

\[
\text{PER}_A(X, Y) = \left\{ c \in A \mid (x, x') \in X \implies (cx, cx') \in Y \right\}
\]

\[
\{ (c, c') \mid \forall x \in |X|. \ (cx, c'x) \in Y \}
\]

* Thus:
* Often put:

Modulo “the same function”

All the valid \textit{codes} \(c\) (well-dfd?)

Hasuo (Tokyo)
PER$_A$: The Category of PERs

* **Obj.** A PER $X$ on $A$

* **Arr.** The homset is

$$\text{PER}_A(X, Y) = \{ c \in A \mid (x, x') \in X \implies (cx, cx') \in Y \}$$

Thus:  $[c] : X \longrightarrow Y$ (with $c \in A$)
PER$_A$: The Category of PERs

* **Obj.** A PER $X$ on $A$

* **Arr.** The homset is

$$\text{PER}_A(X, Y) = \left\{ c \in A \mid (x, x') \in X \implies (cx, cx') \in Y \right\}$$

Thus: $[c] : X \longrightarrow Y$ (with $c \in A$)

Often put: $\text{PER}_A(X, Y) = \left\{ (c, c') \mid (x, x') \in X \implies (cx, c'x') \in Y \right\}$

Modulo “the same function”

All the valid codes $c$ (well-dfd?)

Sunday, September 11, 2011
**Type Constructors in \( \text{PER}_A \)**

**Thm. ([AL05])**
If \( A \) is an affine LCA, then \( \text{PER}_A \) is a linear category. Furthermore, \( \text{PER}_A \) has finite products and coproducts.

* **Linear category** [Benton&Wadler,LICS’96][Bierman,TLCA’95]
  
* **Categorical model of linear logic/linear \( \lambda \), with**
  
* **Monoidal closed with \( \otimes, I, \_\_ \)**

* **Linear exponential comonad \( ! \)**
Type Constructors in \( \text{PER}_A \)

with full \( K: K_{xy} = x \)

**Thm. ([AL05])**
If \( A \) is an affine LCA, then \( \text{PER}_A \) is a linear category. Furthermore, \( \text{PER}_A \) has finite products and coproducts.

* Linear category [Benton&Wadler,LICS’96][Bierman,TLCA’95]
  * Categorical model of linear logic/linear \( \lambda \), with
    * Monoidal closed with \( \otimes, I, \rightarrow \)
    * Linear exponential comonad !
Thm. ([AL05])
If $A$ is an affine LCA, then $\text{PER}_A$ is a linear category. Furthermore, $\text{PER}_A$ has finite products and coproducts.

- Linear category [Benton&Wadler,LICS’96][Bierman,TLCA’95]
  - Categorical model of linear logic/linear $\lambda$, with
    - Monoidal closed with $\otimes, I, \multimap$
    - Linear exponential comonad !
Type Constructors in PER<sub>A</sub>

* How to get operators $\otimes, \times, +, \ldots$

* Like "programming in untyped λ"!
Type Constructors in \( \text{PER}_A \)

* How to get operators \( \otimes, \times, +, \ldots \)

* Like “programming in untyped \( \lambda \)"!
Type Constructors in \( \text{PER}_A \)

* How to get operators \( \boxtimes, \times, +, \ldots \)

* Like “programming in untyped \( \lambda \)”!

\[
\begin{align*}
\text{n} & := \lambda f x. f (f \cdots (f x) \cdots) & \text{Church numeral} \\
\text{K} & := \text{KI} \\
\text{P} & := \lambda x y z. z x y & \text{Paring} \\
\text{P}_L & := \lambda w. w \text{K} & \text{Left projection} \\
\text{P}_R & := \lambda w. w \text{K} & \text{Right projection}
\end{align*}
\]
Type Constructors in $\text{PER}_A$

* How to get operators $\boxplus$, $\times$, $\div$, $\ldots$

* Like “programming in untyped $\lambda$”!

\[
\begin{align*}
\text{n} & := \lambda fx. f(f \cdots (fx) \cdots) \\
\overline{K} & := KI \\
P & := \lambda xyz. zxy \\
P_l & := \lambda w. wK \\
P_r & := \lambda w. w\overline{K}
\end{align*}
\]

Church numeral

Paring

Left projection

Right projection

\[
\begin{align*}
P_l(Pxy) & = x \\
P_r(Pxy) & = y
\end{align*}
\]
Type Constructors in \textit{PER}_A

\* How to get operators $\otimes, \times, \oplus, \ldots$

\* Like "programming in untyped $\lambda$"!

\[
\begin{align*}
\text{n} & := \lambda fx.f(f \cdots (fx) \cdots) & \text{Church numeral} \\
\overline{K} & := KI \\
P & := \lambda xyz.zxy & \text{Paring} \\
P_l & := \lambda w.wK & \text{Left projection} \\
P_r & := \lambda w.w\overline{K} & \text{Right projection}
\end{align*}
\]

\* Cf. Combinatorcompleteness

\[
\begin{align*}
P_l(Pxy) & = x \\
P_r(Pxy) & = y
\end{align*}
\]
Type Constructors in \( \text{PER}_A \)

\[ \frac{X \in \text{PER}_A}{X \subseteq A \times A, \text{ sym., trans.}} \]
Type Constructors in \( \text{PER}_A \)

\[
X \boxdot Y := \left\{ (P_{xy}, P_{x'y'}) \mid (x, x') \in X \land (y, y') \in Y \right\}
\]

\[
X \times Y := \left\{ (Pk_1(Pk_2u), Pk'_1(Pk'_2u')) \mid (k_1u, k'_1u') \in X \land (k_2u, k'_2u') \in Y \right\}
\]

\[
! X := \left\{ (!x, !x') \mid (x, x') \in X \right\}
\]

\[
X + Y := \left\{ (PKx, PKx') \mid (x, x') \in X \right\} \cup \left\{ (PKy, PKy') \mid (y, y') \in Y \right\}
\]

\[
X \rightarrow Y := \left\{ (c, c') \mid (x, x') \in X \Rightarrow (cx, c'x') \in Y \right\}
\]

\[
\frac{X \in \text{PER}_A}{X \subseteq A \times A, \text{ sym., trans.}}
\]
Type Constructors in \( \text{PER}_A \)

\[
X \boxtimes Y := \left\{ (P_{x,y}, P_{x',y'}) \mid (x, x') \in X \land (y, y') \in Y \right\}
\]

\[
X \times Y := \left\{ (P_{k_1}k_2u, P_{k_1}k'_2u') \mid (k_1u, k'_1u') \in X \land (k_2u, k'_2u') \in Y \right\}
\]

\[
! X := \left\{ (!x, !x') \mid (x, x') \in X \right\}
\]

\[
X + Y := \left\{ (PK_x, PK_{x'}) \mid (x, x') \in X \right\} \cup \left\{ (PK_y, PK_{y'}) \mid (y, y') \in Y \right\}
\]

\[
X \rightarrow Y := \left\{ (c, c') \mid (x, x') \in X \implies (cx, c'x') \in Y \right\}
\]
Type Constructors in \( \text{PER}_A \)

\[
X \boxprod Y := \{ (Pxy, Px'y') \mid (x, x') \in X \land (y, y') \in Y \}
\]

\[
X \times Y := \{ (Pk_1(Pk_2u), Pk'_1(Pk'_2u')) \mid (k_1u, k'_1u') \in X \land (k_2u, k'_2u') \in Y \}
\]

\[
! X := \{ (!x, !x') \mid (x, x') \in X \}
\]

\[
X + Y := \{ (PKx, PKx') \mid (x, x') \in X \}
\]

\[
\cup \{ (PKy, PKy') \mid (y, y') \in Y \}
\]

\[
X \rightsquigarrow Y := \{ (c, c') \mid (x, x') \in X \implies (cx, c'x') \in Y \}
\]
Type Constructors in \( \text{PER}_A \)

\[
\begin{align*}
X \boxtimes Y & := \{ (Px y, Px' y') \mid (x, x') \in X \land (y, y') \in Y \} \\
X \times Y & := \{ (Pk_1(Pk_2 u), Pk'_1(Pk'_2 u')) \mid (k_1 u, k'_1 u') \in X \land (k_2 u, k'_2 u') \in Y \} \\
! X & := \{ (!x, !x') \mid (x, x') \in X \} \\
X + Y & := \{ (PK x, PK x') \mid (x, x') \in X \} \\
& \quad \cup \{ (PK y, PK y') \mid (y, y') \in Y \} \\
X \rightarrow Y & := \{ (c, c') \mid (x, x') \in X \implies (cx, c' x') \in Y \}
\end{align*}
\]

\( X \in \text{PER}_A \)

\( X \subseteq A \times A, \) sym., trans.

\( \) multiplicative and

\( \) additive and

CPS-style. \( k_1, k_2: \) “access methods”
Summary: Realizability

Affine LCA $A$

- $a \cdot b$, $!a$, $B, C, I, \ldots$

Linear category $\text{PER}_A$

- Type constructors via “programming in untyped $\lambda$”
- Symmetric monoidal closed $\otimes, I, \rightarrow$
- Finite product, coproduct

$(a, c \in A)$
**Summary: Realizability**

**Affine LCA** \( A \)

\( a \cdot b, \quad !a, \quad B, C, I, \ldots \)

**Linear category** \( \text{PER}_A \)

\[
\begin{array}{ccc}
X & \overset{[c]}{\longrightarrow} & Y \\
[a] & \overset{[c \cdot a]}{\longrightarrow} & [c \cdot a]
\end{array}
\]

\((a,c \in A)\)

- Type constructors via “programming in untyped \( \lambda \)”
- Symmetric monoidal closed \( \otimes, I, \rightarrow \)
- Finite product, coproduct

*Not \( \otimes\), for distinction*
Branching monad $B$

Coalgebraic trace semantics

Traced monoidal category $\mathcal{C}$
+ other constructs $\Rightarrow$ “GoI situation” [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

\[ \text{Model of typed calculus} \]

\[ \text{Model of untyped calculus} \]

* Applicative str. + combinators

\[ \text{Hasuo (Tokyo)} \]
The Categorical GoI Workflow

Branching monad $B$

Coalgebraic trace semantics

Traced monoidal category $\mathcal{C}$
+ other constructs $\rightarrow$ “GoI situation” [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

Model of typed calculus

$g \cdot f = \begin{array}{c} g \\ \downarrow \\ f \end{array}$

Applicative str. + combinators

Model of untyped calculus

$\begin{array}{c} f \\ \downarrow \\ U \end{array} \in \mathcal{C}(U, U)$
**Summary: Realizability**

**Affine LCA** $A$

$a \cdot b, \ ! a, \ B, C, I, \ldots$

**Linear category** $\text{PER}_A$

* Type constructors via “programming in untyped $\lambda$”
  * Symmetric monoidal closed $\otimes, I, \multimap$
  * Finite product, coproduct

Not $\otimes$, for distinction
Affine LCA

\[ a \cdot b, \; !a, \; B, C \]

Linear category \( \text{PER}_A \)

\[
\begin{align*}
X & \overset{[c]}{\longrightarrow} Y \\
[a] & \overset{[c \cdot a]}{\longrightarrow}
\end{align*}
\]

\((a, c \in A)\)

Type constructors via “programming in untyped \( \lambda \)”

- Symmetric monoidal closed \( \otimes, I, \rightarrow \)
- Finite product, coproduct

Not \( \otimes \), for distinction
Affine LCA

\[ a \cdot b, !a, B, C \]

Linear category \( \text{PER}_A \)

\[
\begin{array}{c}
\text{X} \\
\quad [c] \\
\quad [a] \\
\quad [c \cdot a] \\
\downarrow \\
\text{Y}
\end{array}
\]

- Type constructors via “programming in untyped \( \lambda \)"
- Symmetric monoidal closed \( \boxtimes, I, \rightarrow \)
- Finite product, coproduct

Not \( \otimes \), for distinction
Time to Wake Up!!
Part 3

Quantum Computation in 5 min.
What You Need to Know

* Not much, really!
* Our principal reference:
  * Its Chap. 3 & Chap. 8
  * Hilbert space formulation
  * Quantum operation formalism (Kraus)
  * No need for the Bloch sphere
What You Need to Know

* Not much, really!

* Our principal reference:
  * Its Chap. 3 & Chap. 8
  * Hilbert space formulation
  * Quantum operation formalism (Kraus)
  * No need for the Bloch sphere
Some Principles

* A state of a 1-qubit system = a normalized vector

\[ |\varphi\rangle = \alpha |0\rangle + \beta |1\rangle \in \mathbb{C}^2 \]

* with \( \| |\varphi\rangle \|^2 = |\alpha|^2 + |\beta|^2 = 1 \)

* Various notations for base:
  \( \{ |0\rangle, |1\rangle \}, \{ |+\rangle, |-\rangle \}, \{ |\uparrow\rangle, |\downarrow\rangle \}, \ldots \)
Some Principles

* Composed system: \( \otimes \), not \( \times \).

* not \( \mathbb{C}^2 \times \mathbb{C}^2 \times \mathbb{C}^2 \cong \mathbb{C}^6 \), with base \( \{ |01\rangle, |02\rangle, |03\rangle \} \)

* but \( \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 \cong \mathbb{C}^8 \),
with base \( \{ |000\rangle, |001\rangle, |010\rangle, |011\rangle \} \)

Hasuo (Tokyo)
Some Principles

* Composed system: $\otimes$, not $\times$.

* Source of power of quantum comp./comm.
  * N-qubit $\rightarrow 2^N$-dim (not $2N$-dim)

* Entanglement; superposition
Three Quantum Primitives

* Preparation

* Unitary transformation

* Measurement

Hasuo (Tokyo)
Three Quantum Primitives

* Preparation

* Creates/“prepares” a quantum state (typically $|0\rangle$)
Three Quantum Primitives

* Unitary transformation

\[ \alpha |0\rangle + \beta |1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \xrightarrow{U} U \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \]

* Unitary matrix: \( UU^\dagger = U^\dagger U = I \)

* Invertible. “Rotation”

* Also for N-dim systems (of course)
Three Quantum Primitives

**Measurement**

When one measures

\[ \alpha |0\rangle + \beta |1\rangle \]

- \( |0\rangle \) is observed, and the state becomes \( |0\rangle \) with prob. \( |\alpha|^2 \)
- \( |1\rangle \) is observed, and the state becomes \( |1\rangle \) with prob. \( |\beta|^2 \)
Three Quantum Primitives

* Measurement

When one measures

\[ \alpha|0\rangle + \beta|1\rangle \]

- \( |0\rangle \) is observed, and
- The state becomes \( |0\rangle \) with prob. \( |\alpha|^2 \)

- \( |1\rangle \) is observed, and
- The state becomes \( |1\rangle \) with prob. \( |\beta|^2 \)
Three Quantum Primitives

* Measurement

When one measures

\[ \alpha |0\rangle + \beta |1\rangle \]

* \( |0\rangle \) is observed, and
  * the state becomes \( |0\rangle \) with prob. \( |\alpha|^2 \)

* \( |1\rangle \) is observed, and
  * the state becomes \( |1\rangle \) with prob. \( |\beta|^2 \)

state reduction
Measurement

When one measures

\[ \alpha |0\rangle + \beta |1\rangle \]

* \( |0\rangle \) is observed, and with prob. \( |\alpha|^2 \)
* the state becomes \( |0\rangle \)

* \( |1\rangle \) is observed, and with prob. \( |\beta|^2 \)
* the state becomes \( |1\rangle \)

state reduction

Also: for other dimensions, bases

Hasuo (Tokyo)
Entanglement

$qubit_1$ $qubit_2$
Entanglement

\[ \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle \]
Entanglement

\[ \frac{1}{\sqrt{2}} \left| 00 \right\rangle + \frac{1}{\sqrt{2}} \left| 11 \right\rangle \]
Entanglement

\[ \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle \]

qubit_1  qubit_2

\[ \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle \]

qubit_1  qubit_2
Entanglement

\[ \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle \]
Entanglement

\[ \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle \]

|00\rangle with prob. \( \frac{1}{2} \)

|11\rangle with prob. \( \frac{1}{2} \)

measure!!
Density Matrix, Quantum Operation

- Advanced, mathematically convenient formalisms
- State vector $\rightarrow$ density matrix
- Use $|\varphi\rangle\langle\varphi|$ in place of $|\varphi\rangle$
- Can also represent mixed states, e.g.

| $|00\rangle$ with prob. $\frac{1}{2}$ |
| $|11\rangle$ with prob. $\frac{1}{2}$ |

- Quantum operation (QO) [Kraus]
  - $\{\text{QOs}\} = \{\text{any combinations of preparation, Unitary transf., measurement}\}$
  - But no classical control (like case-distinction)
- Used in [Selinger,MSCS’04] and other
Density Matrix, Quantum Operation

Defn.

- An \textit{m-dimensional density matrix} is an \(m \times m\) matrix \(\rho \in \mathbb{C}^{m \times m}\) which is positive and satisfies \(\text{tr}(\rho) \in [0, 1]\).

  - Notation: \(D_m = \{m\text{-dim. density matrices}\}\)

- A \textit{quantum operation (QO)} is a mapping \(\mathcal{E} : D_m \rightarrow D_n\) subject to the following axioms.

  1. (Trace condition) \(\text{tr}[\mathcal{E}(\rho)] \in [0, 1]\) for any \(\rho \in D_m\).

  2. (Linearity) Let \((\rho_i)_{i \in I}\) be a family of \(m\)-dim. density matrices; and \((p_i)_{i \in I}\) be a probability subdistribution (meaning \(\sum_i p_i \leq 1\)). Then: \(\mathcal{E}(\sum_{i \in I} p_i \rho_i) = \sum_{i \in I} p_i \mathcal{E}(\rho_i)\).

  3. (Complete positivity) An arbitrary “extension” of \(\mathcal{E}\) of the form \(\mathcal{I}_k \otimes \mathcal{E} : M_k \otimes M_m \rightarrow M_k \otimes M_n\) carries a positive matrix to a positive one.

  - Notation: \(\mathcal{Q}O_{m,n} = \{\text{QOs from } m\text{-dim. to } n\text{-dim.}\}\)

- For specialists: we allow trace \(\leq 1\)

- So that \textit{probabilities are implicitly carried by density matrices}
Quantum Computation: Summary

* A quantum state $= a$ vector $|\varphi\rangle$

* Composition by $\otimes$
  $\Rightarrow$ Dimension grows exponentially

* Three primitives:
  * Preparation
  * Unitary transformation
  * Measurement ($\Rightarrow$ st. reduction)
Quantum Computation: 
Summary

* A quantum state = a vector $|\psi\rangle$

* Composition by $\otimes$
  $\Rightarrow$ Dimension grows exponentially

* Three primitives:
  * Preparation
  * Unitary transformation
  * Measurement ($\Rightarrow$ st. reduction)

Generalized to density matrix

Hasuo (Tokyo)
Quantum Computation: Summary

- A quantum state = a vector $|\varphi\rangle$
- Composition by $\otimes$
  - Dimension grows exponentially
- Three primitives:
  - Preparation
  - Unitary transformation
  - Measurement ($\rightarrow$ st. reduction)

Generalized to density matrix
Unified to quantum operation (QO)
Part 4

Quantum GoI
The Language $q\lambda^e$

- Roughly: **linear $\lambda$ + quantum primitives**
- “Quantum data, classical control”
- No superposed threads
- Based on [Selinger&Valiron’09]
- With slight modifications
- Notably: quantum $\otimes$ vs. linear logic $\otimes$
  - The same in [Selinger&Valiron’09]
    - clean type system, aids programming
- But... problem with GoI-style semantics
The Language $q\lambda_\ell$

The *types* of $q\lambda_\ell$ are:

$$A, B ::= n\text{-qbit} \mid !A \mid A \rightarrow B \mid \top \mid A \otimes B \mid A + B,$$

with conventions $\text{qbit} ::= 1\text{-qbit}$ and $\text{bit} ::= \top + \top$.

The *terms* of $q\lambda_\ell$ are:

$$M, N, P ::= 
\begin{align*}
x \mid \lambda x^A.M & \mid MN \mid \langle M, N \rangle \mid \ast \mid \\
\text{let } \langle x^A, y^B \rangle = M \text{ in } N & \mid \text{let } \ast = M \text{ in } N \\
\text{inj}_B M & \mid \text{inj}_r A M \\
\text{match } P \text{ with } (x^A \mapsto M \mid y^B \mapsto N) & \\
\text{letrec } f^A x = M \text{ in } N \\
\text{new } |0\rangle & \mid \text{meas}_{i}^{n+1} U \mid \text{cmp}_{m,n}
\end{align*}$$

with conventions $\text{tt} ::= \text{inj}_\ell^\top(*)$ and $\text{ff} ::= \text{inj}_r^\top(*)$. 

Hasuo (Tokyo) 

Sunday, September 11, 2011
The types of $q\lambda_e$ are:

$$A, B ::= \text{n-qbit} \mid !A \mid A \rightarrow B \mid \top \mid A \bigotimes B \mid A + B,$$

with conventions $\text{qbit} := 1\text{-qbit}$ and $\text{bit} := \top + \top$.

The terms of $q\lambda_e$ are:

$$M, N, P ::=$$

$$x \mid \lambda x^A.M \mid MN \mid \langle M, N \rangle \mid * \mid$$

$$\text{let} \langle x^A, y^B \rangle = M \text{ in } N \mid \text{let } * = M \text{ in } N \mid$$

$$\text{inj}_B^M \mid \text{inj}_r^A M \mid$$

$$\text{match } P \text{ with } (x^A \mapsto M \mid y^B \mapsto N) \mid$$

$$\text{letrec } f^A x = M \text{ in } N \mid$$

$$\text{new } |0\rangle \mid \text{meas}^{n+1}_i U \mid \text{cmp}_{m,n} \mid$$

with conventions $\text{tt} := \text{inj}_l^\top(*)$ and $\text{ff} := \text{inj}_r^\top(*)$. 

Different from quantum $\otimes$ (Unlike [Selinger-Valiron'09]); same as the one in PER.
The terms of $q\lambda_\ell$ are:

$$M, N, P ::= x | \lambda x^A.M | MN | \langle M, N \rangle | * |$$

let $\langle x^A, y^B \rangle = M$ in $N$ | let $* = M$ in $N$ |

$\text{inj}^B M | \text{inj}^A M |

match $P$ with $(x^A \mapsto M | y^B \mapsto N)$ |

letrec $f^A x = M$ in $N$ |

new $|0\rangle | \text{meas}_i^{n+1} | U | \text{cmp}_{m,n}$ ,

with conventions $\text{tt} := \text{inj}^\top_\ell(*)$ and $\text{ff} := \text{inj}^\top_r(*)$.
The Language

2-qbit \cong qbit \otimes qbit

\[ A, B ::= n\text{-}qbit \mid !A \mid A \rightarrow B \mid \top \mid A \boxtimes B \mid A + B, \]
with conventions \( qbit := 1\text{-}qbit \) and \( \text{bit} := \top + \top \).

The terms of \( q\lambda_\ell \) are:

\[ M, N, P ::= \]
\[ x \mid \lambda x^A.M \mid MN \mid \langle M, N \rangle \mid * \mid \]
\[ \text{let} \langle x^A, y^B \rangle = M \text{ in } N \mid \text{let } * = M \text{ in } N \mid \]
\[ \text{inj}_{\ell}^B M \mid \text{inj}_{r}^A M \mid \]
\[ \text{match } P \text{ with } (x^A \mapsto M \mid y^B \mapsto N) \mid \]
\[ \text{letrec } f^A x = M \text{ in } N \mid \]
\[ \text{new } |0\rangle \mid \text{meas}_{i}^{n+1} \mid U \mid \text{cmp}_{m,n}, \]
with conventions \( \text{tt} := \text{inj}_{\ell}^{\top}(\ast) \) and \( \text{ff} := \text{inj}_{r}^{\top}(\ast) \).
The Language

2-qbit \cong qbit \otimes qbit

\[ A, B ::= n\text{-qbit} \mid !A \mid A \rightarrow B \mid \top \mid A \boxtimes B \mid A + B , \]
with conventions qbit := 1-qbit and bit := \top + \top .

The terms of \( q\lambda_\ell \) are:

\[ M, N, P ::= \]
\[ x \mid \lambda x^A.M \mid MN \mid \langle M, N \rangle \mid \ast \mid \]
\[ \text{let } \langle x^A, y^B \rangle = M \text{ in } N \mid \text{let } \ast = M \text{ in } N \mid \]
\[ \text{inj}_B M \mid \text{inj}_r^A M \mid \]
\[ \text{match } P \text{ with } (x^A \mapsto M \mid y^B \mapsto N) \mid \]
\[ \text{letrec } f^A x = M \text{ in } N \mid \]
\[ \text{new } \mid 0 \mid \text{meas}^{n+1} \mid U \mid \text{cmp}_{m,n} , \]
with conventions \( \text{tt} := \text{inj}_{\ell}^{\top}(\ast) \) and \( \text{ff} := \text{inj}_{r}^{\top}(\ast) \).
Implicit linearity tracking via subtyping <:

\[ \text{e.g. } !A <: A, \text{ !}A <: !!A \]

(following [Selinger-Valiron’09])

\[ n = 0 \Rightarrow m = 0 \]  

\[ k \text{-qubit} \]

\[ A_1 <: B_1, A_2 <: B_2 \]

\[ \Delta \vdash M : !^n A \]

\[ \Delta \vdash N : !^n B \]

\[ \Delta \vdash \text{inj}^B_\ell M : !^n (A + B) \]

\[ \Delta \vdash \text{inj}^B_r N : !^n (A + B) \]

\[ \Delta, \Gamma_1 \vdash P : !^n (A + B) \]

\[ \Delta, \Gamma_2, x : !^n A \vdash M : C \]

\[ \Delta, \Gamma_2, y : !^n B \vdash N : C \]

\[ \Delta, \Gamma_1, \Gamma_2 \vdash \text{match } P \text{ with } (x'^n A \mapsto M | y'^n B \mapsto N) : C \]

\[ x : A, \Delta \vdash M : B \]

\[ \Delta \vdash !x^A.M : A \rightarrow B \]

\[ \Delta \vdash !^n (A \rightarrow B) \]

\[ \Delta, \Gamma_1 \vdash M : A \rightarrow B \]

\[ \Delta, \Gamma_2 \vdash N : A \]

\[ \Delta, \Gamma_1, \Gamma_2 \vdash MN : B \]

\[ \Delta, \Gamma_1 \vdash !^n A_1 \]

\[ \Delta, \Gamma_2 \vdash !^n A_2 \]

\[ \Delta, \Gamma_1, \Gamma_2 \vdash \langle M_1, M_2 \rangle : !^n (A_1 \otimes A_2) \]

\[ !\Delta, \Gamma_1, \Gamma_2 \vdash \text{let } \langle x_1'^n A_1, x_2'^n A_2 \rangle = M \text{ in } N : A \]

\[ \Delta, \Gamma_1 \vdash M : \top \]

\[ \Delta, \Gamma_2 \vdash N : A \]

\[ \Delta, \Gamma_1 \vdash \text{let } * = M \text{ in } N : A \]

\[ \Delta, \Gamma, f : !(A \rightarrow B) \vdash N : C \]

\[ \Delta, \Gamma, f : !(A \rightarrow B), x : A \vdash M : B \]

\[ \Delta, \Gamma \vdash \text{letrec } f^{A \rightarrow B} x = M \text{ in } N : C \]

---

**Measurements**

\[ A_{\text{new}}^{(0)} : = \text{qbit} \]

\[ A_{\text{meas}}^{(n+1)} : = (n + 1)\text{-qubit} \rightarrow (\text{bit } \otimes n\text{-qubit}) \text{ for } n \geq 1 \]

\[ A_{\text{meas}}^{(1)} : = \text{qbit} \rightarrow \text{bit} \]

\[ A_U : = n\text{-qubit} \rightarrow n\text{-qubit} \text{ for a } 2^n \times 2^n \text{ matrix } U \]

\[ A_{\text{cmp}}_{m,n} : = (m\text{-qubit } \otimes n\text{-qubit}) \rightarrow (m + n)\text{-qubit} \]

---

**Bookkeeping**

(due to \( \otimes \) vs. \( \boxdot \))
Operational Semantics

\[ E[(\lambda x^A.M)V] \rightarrow_1 E[M[V/x]] \]
\[ E[\text{let} \langle x^A, y^B \rangle = \langle V, W \rangle \text{in} M] \rightarrow_1 E[M[V/x, W/y]] \]
\[ E[\text{let} \ast = \ast \text{in} M] \rightarrow_1 E[M] \]
\[ E[\text{match} (\text{inj}^B_V) \text{with} (x'^n_A \mapsto M \mid y'^n_B \mapsto N)] \rightarrow_1 E[M[V/x]] \]
\[ E[\text{match} (\text{inj}^A_V) \text{with} (x'^n_A \mapsto M \mid y'^n_B \mapsto N)] \rightarrow_1 E[N[V/y]] \]
\[ E[\text{letrec} f^{A\rightarrow B} x = M \text{in} N] \rightarrow_1 E[N[\lambda x^A.\text{letrec} f^{A\rightarrow B} x = M \text{in} M/f]] \]
\[ E[\text{meas}_i^{n+1}(\text{new} \rho)] \rightarrow_1 E[\langle \text{tt}, \text{new} \langle 0_i|\rho|0_i \rangle \rangle] \]
\[ E[\text{meas}_i^{n+1}(\text{new} \rho)] \rightarrow_1 E[\langle \text{ff}, \text{new} \langle 1_i|\rho|1_i \rangle \rangle] \]
\[ E[\text{meas}_1^1(\text{new} \rho)] \rightarrow \langle 0|\rho|0 \rangle E[\text{tt}] \]
\[ E[\text{meas}_1^1(\text{new} \rho)] \rightarrow \langle 1|\rho|1 \rangle E[\text{ff}] \]
\[ E[U(\text{new} \rho)] \rightarrow_1 E[\text{new} (U \rho)] \]
\[ E[\text{cmp}_{m,n}(\text{new} \rho, \text{new} \sigma)] \rightarrow_1 E[\text{new} (\rho \otimes \sigma)] \]

* Standard small-step one, CBV, but with probabilistic branching (measurement)
The Language $q\lambda^e$

- Roughly: linear $\lambda$ + quantum primitives
- "Quantum data, classical control"
- No superposed threads
- Based on [Selinger&Valiron’09]
- With slight modifications
- Notably: quantum $\otimes$ vs. linear logic
- The same in [Selinger&Valiron’09]
  - clean type system, aids programming
- But... problem with GoI-style semantics
The Categorical GoI Workflow

Branching monad $B$

Coalgebraic trace semantics

Traced monoidal category $\mathbb{C}$
+ other constructs $\rightarrow$ "GoI situation" [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category
The Categorical GoI Workflow

Branching monad $B$

Coalgebraic trace semantics

Traced monoidal category $\mathcal{C}$
+ other constructs $\mapsto$ "GoI situation" [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

Model of quantum language
The Categorical GoI Workflow

- **Branching monad** $B$
- **Coalgebraic trace semantics**
- **Traced monoidal category** $\mathcal{C}$
  - + other constructs $\Rightarrow$ "GoI situation" [AHS02]
- **Categorical GoI** [AHS02]
- **Linear combinatory algebra**
- **Realizability**
- **Linear category**

Quantum LCA

Model of quantum language
The Categorical GoI Workflow

- Branching monad $B$
- Coalgebraic trace semantics
- Traced monoidal category $\mathbb{C}$
  + other constructs $\Rightarrow$ “GoI situation” [AHS02]
- Categorical GoI [AHS02]
- Linear combinatory algebra
- Realizability
- Linear category

Quantum TSMC

Quantum LCA

Model of quantum language
The Categorical GoI Workflow

- Branching monad \( B \)
  - Coalgebraic trace semantics
  - Traced monoidal category \( \mathbb{C} \) + other constructs \( \rightarrow \) “GoI situation” [AHS02]
  - Categorical GoI [AHS02]
  - Linear combinatory algebra
  - Realizability
  - Linear category

- Quantum branching monad
  - Quantum TSMC
  - Quantum LCA
  - Model of quantum language

Hasuo (Tokyo)
Different Branching in The GoI Animation

- **Pfn (partial functions)**
- **Pipe can be stuck**
- **Rel (relations)**
- **Pipe can branch**
- **DSRel**
- **Pipe is probabilistically branched**
Different Branching in The GoI Animation

- Pfn (partial functions)
- Pipe can be stuck
- Rel (relations)
- Pipe can branch
- DSRel
- Pipe is probabilistically branched
Different Branching in The GoI Animation

- Pfn (partial functions)
- Pipe can be stuck
- Rel (relations)
- Pipe can branch
- DSRel
- Pipe is probabilistically branched
Different Branching in The GoI Animation

- **Pfn** (partial functions)
- Pipe can be stuck
- **Rel** (relations)
- Pipe can branch
- **DSRel**
- Pipe is probabilistically branched
Different Branching in The GoI Animation

- **Pfn (partial functions)**
- Pipe can be stuck
- **Rel (relations)**
- Pipe can branch
- **DSRel**
- Pipe is probabilistically branched
Different Branching in The GoI Animation

- Pfn (partial functions)
- Pipe can be stuck
- Rel (relations)
- Pipe can branch
- DSRel
- Pipe is probabilistically branched
Different Branching in The GoI Animation

* Pfn (partial functions)
* Pipe can be stuck
* Rel (relations)
* Pipe can branch
* DSRel
* Pipe is probabilistically branched
Different Branching in The Animation

- **Pfn** (partial functions)
- Pipe can be stuck
- **Rel** (relations)
- Pipe can branch
- **DSRel**
- Pipe is probabilistically branched

$K_l(\mathcal{L})$, non-termination

Hasuo (Tokyo)
Different Branching in The GoI Animation

- Pfn (partial functions)
  - Pipe can be stuck
- Rel (relations)
  - Pipe can branch
- DSRel
  - Pipe is probabilistically branched

$Kl(\mathcal{L})$, non-termination

$Kl(\mathcal{P})$, non-determinism
Different Branching in The GoI Animation

- **Pfn** (partial functions)
  - Pipe can be stuck
- **Rel** (relations)
  - Pipe can branch
- **DSRel**
  - Pipe is probabilistically branched

- $Kl(\mathcal{L})$, non-termination
- $Kl(\mathcal{P})$, non-determinism
- $Kl(\mathcal{D})$, probability
Quantum Geometry of Interaction

\[ [M] = M \]

... (countably many)
Quantum Geometry of Interaction

Not just a token/particle, but quantum state!

\[
[M] = M
\]
Quantum
Geometry of Interaction

Not just a token/particle, but quantum state!
Quantum Geometry of Interaction

\[
\begin{bmatrix} M \end{bmatrix} = M
\]

Not just a token/ particle, but quantum state!

“Quantum Data”
Quantum Geometry of Interaction

\[
[M] = M
\]

"Quantum Data"

"Classical Control"

Not just a token/particle, but quantum state!

(countably many)

Hasuo (Tokyo)
Quantum Geometry of Interaction

\[
\begin{bmatrix} M \end{bmatrix} = M
\]

“Quantum Data”

Not just a token/particle, but quantum state!

“in which pipe”

(measurement \(\rightarrow\) case-distinction) leads a token to different pipes

“Classical Control”

(countably many)

Hasuo (Tokyo)
The Quantum Branching Monad

\[ QY = \left\{ c : Y \rightarrow \prod_{m,n \in \mathbb{N}} QO_{m,n} \mid \text{the trace condition} \right\} \]
The Quantum Branching Monad

\[ QY = \{ c : Y \rightarrow \prod_{m,n \in \mathbb{N}} QO_{m,n} \mid \text{the trace condition} \} \]

\[ \sum_{y \in Y} \sum_{n \in \mathbb{N}} \text{tr} \left[ (c(y))_{m,n}(\rho) \right] \leq 1, \]

\[ \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

(trace of matrix \( \approx \) probability)
The Quantum Branching Monad

\[ QY = \left\{ c : Y \rightarrow \prod_{m,n \in \mathbb{N}} QO_{m,n} \mid \text{the trace condition} \right\} \]

* Compare with

\[ PY = \left\{ c : Y \rightarrow 2 \right\} \]

\[ DY = \left\{ c : Y \rightarrow [0, 1] \mid \sum_{y \in Y} c(y) \leq 1 \right\} \]

\[ \sum_y \sum_n \text{tr} \left[ (c(y))_{m,n}(\rho) \right] \leq 1 , \]

\[ \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

(trace of matrix \( \approx \) probability)
The Quantum Branching Monad

\[ QY = \left\{ c : Y \rightarrow \prod_{m,n \in \mathbb{N}} QO_{m,n} \right\} \quad \text{the trace condition} \]

\[ \sum \sum_{y \in Y} \sum_{n \in \mathbb{N}} \text{tr} \left[ (c(y))_{m,n}(\rho) \right] \leq 1, \quad \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

(trace of matrix \( \approx \) probability)

\[ \mathcal{P}Y = \left\{ c : Y \rightarrow 2 \right\} \]

\[ \mathcal{D}Y = \left\{ c : Y \rightarrow [0, 1] \left| \sum_{y \in Y} c(y) \leq 1 \right. \right\} \]

* Compare with
The Quantum Branching Monad

\[ QY = \{ c : Y \rightarrow \prod_{m,n \in \mathbb{N}} QO_{m,n} \mid \text{the trace condition} \} \]

\[ \sum \sum_{y \in Y, n \in \mathbb{N}} \text{tr} \left[ (c(y))_{m,n}(\rho) \right] \leq 1, \quad \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

(Trace of matrix \( \approx \) probability)

Compare with

\[ PY = \{ c : Y \rightarrow \mathbb{2} \} \]

\[ DY = \{ c : Y \rightarrow [0, 1] \mid \sum_{y \in Y} c(y) \leq 1 \} \]
The Quantum Branching Monad

\[ QY = \left\{ c : Y \to \prod_{m, n \in \mathbb{N}} QO_{m,n} \right\} \quad \text{the trace condition} \]

\[ \sum \sum \text{tr} \left[ (c(y))_{m,n}(\rho) \right] \leq 1, \quad \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

(\text{trace of matrix } \approx \text{ probability})

\[ \mathcal{P}Y = \left\{ c : Y \to 2 \right\} \]

\[ \mathcal{D}Y = \left\{ c : Y \to [0, 1] \mid \sum_{y \in Y} c(y) \leq 1 \right\} \]
The Quantum Branching Monad

\[ QY = \left\{ c : Y \rightarrow \prod_{m,n \in \mathbb{N}} QO_{m,n} \right\} \text{ the trace condition} \]

\[ \sum \sum_{y \in Y \ n \in \mathbb{N}} \text{tr} \left[ (c(y))_{m,n}(\rho) \right] \leq 1, \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

(Trace of matrix \(\approx\) probability)

* Compare with

\[ \mathcal{P}Y = \left\{ c : Y \rightarrow 2 \right\} \]

\[ \mathcal{D}Y = \left\{ c : Y \rightarrow [0,1] \right\} \text{ the trace condition} \]

\[ \sum_{y \in Y} c(y) \leq 1 \]
The Quantum Branching Monad

\[ QY = \left\{ c : Y \rightarrow \prod_{m,n \in \mathbb{N}} QO_{m,n} \mid \text{the trace condition} \right\} \]

\[ \sum_{y \in Y} \sum_{n \in \mathbb{N}} \text{tr}[(c(y))_{m,n}(\rho)] \leq 1, \quad \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

* Given \( x \in X, y \in Y, m \in \mathbb{N}, n \in \mathbb{N} \) determines a quantum operation

\[ (f(x)(y))_{m,n} : D_m \rightarrow D_n \]

* Subject to the \text{trace condition}

Any opr. on quantum data: combination of
- preparation
- unitary transf.
- measurement
The Quantum Branching Monad

\[ QY = \left\{ c : Y \rightarrow \prod_{m,n \in \mathbb{N}} QO_{m,n} \mid \text{the trace condition} \right\} \]

\[ \sum_{y \in Y} \sum_{n \in \mathbb{N}} \text{tr}[(c(y))_{m,n}(\rho)] \leq 1, \quad \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

* Given \( x \in X, y \in Y, m \in \mathbb{N}, n \in \mathbb{N} \) determines a quantum operation \((f(x)(y))_{m,n}\)

* trace cond.:
The Quantum Branching Monad

\[ Q_Y = \{ c : Y \to \prod_{m,n \in \mathbb{N}} Q_{O_{m,n}} \mid \text{the trace condition} \} \]

\[ \sum_{y \in Y} \sum_{n \in \mathbb{N}} \text{tr}[(c(y))_{m,n} (\rho)] \leq 1, \quad \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

Given \( x \in X, y \in Y, m \in \mathbb{N}, n \in \mathbb{N} \)
determines a quantum operation \( (f(x)(y))_{m,n} \)

\[ X \xrightarrow{f} Y \text{ in } \mathcal{K}\ell(\mathbb{Q}) \quad \frac{X \rightarrow QY \text{ in Sets}}{	ext{entrance}} \quad \text{exit} \quad \text{in.} \quad \text{dim.} \quad \text{out.} \quad \text{dim.} \]

[Diagram of quantum branching monad]

\[ \text{trace cond.:} \]

Hasuo (Tokyo)

Sunday, September 11, 2011
The Quantum Branching Monad

\[ XY = \{ c : Y \to \prod_{m,n \in \mathbb{N}} QO_{m,n} \mid \text{the trace condition} \} \]

\[ \sum_{y \in Y} \sum_{n \in \mathbb{N}} \text{tr}[(c(y))_{m,n}(\rho)] \leq 1 , \quad \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

* Given \( x \in X, y \in Y, m \in \mathbb{N}, n \in \mathbb{N} \)
determines a quantum operation \( (f(x)(y))_{m,n} \)

* trace cond.:
The Quantum Branching Monad

\[ \mathcal{Q}Y = \left\{ c : Y \to \prod_{m,n \in \mathbb{N}} QO_{m,n} \mid \text{the trace condition} \right\} \]

\[ \sum_{y \in Y} \sum_{n \in \mathbb{N}} \text{tr}[(c(y))_{m,n}(\rho)] \leq 1, \quad \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

\[
\begin{align*}
X & \xrightarrow{f} Y \quad \text{in} \ K\ell(\mathcal{Q}) \\
X & \to \mathcal{Q}Y \quad \text{in} \ \text{Sets}
\end{align*}
\]

* Given \( x \in X, y \in Y, m \in \mathbb{N}, n \in \mathbb{N} \)

determines a quantum operation \( (f(x)(y))_{m,n} \)

* trace cond.:

\( \rho \in D_m \)

\( \text{measure (and others)} \)

\( \text{entrance} \quad \text{exit} \quad \text{dim.} \quad \text{dim.} \)

\( x \quad y \quad y' \)

Hasuo (Tokyo)
The Quantum Branching Monad

\[ X \xrightarrow{f} Y \text{ in } \mathcal{K}_\ell(Q) \]

\[ X \rightarrow QY \text{ in Sets} \]

Given \( x \in X, y \in Y, m \in \mathbb{N}, n \in \mathbb{N} \)

determines a quantum operation \((f(x)(y))_{m,n}\)

\[ \mathcal{Q}Y = \left\{ c : Y \rightarrow \prod_{m,n \in \mathbb{N}} QO_{m,n} \right\} \text{ the trace condition} \]

\[ \sum_{y \in Y} \sum_{n \in \mathbb{N}} \text{tr}[(c(y))_{m,n}(\rho)] \leq 1 \]

\[ \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

\( \rho \in D_m \)

trace cond.:

\[ \left( f(x)(y) \right)_{m,n}(\rho) \in D_n \]

for each \( n \)

entrance exit dim. dim.

in. out.
The Quantum Branching Monad

\[ QY = \left\{ c : Y \to \prod_{m,n \in \mathbb{N}} QO_{m,n} \mid \text{the trace condition} \right\} \]

\[ \sum_{y \in Y} \sum_{n \in \mathbb{N}} \text{tr}[(c(y))_{m,n}(\rho)] \leq 1, \quad \forall m \in \mathbb{N}, \forall \rho \in D_m. \]

\[ f : X \to Y \text{ in } \mathcal{Kl}(Q) \]
\[ X \to QY \text{ in Sets} \]

* Given \( x \in X, y \in Y, m \in \mathbb{N}, n \in \mathbb{N} \)
determines a quantum operation \( (f(x)(y))_{m,n} \)

* trace cond.:
\[ \sum \text{Pr}(\text{Token led to} y \text{ with dim. } n) \leq 1 \]

Given \( f \) in \( \mathcal{Kl}(Q) \),
\[ X \to Y \]

in. \quad out. \quad dim. \quad dim.

\( x \quad \ldots \quad x \)

\( \rho \in D_m \)

\[ y \quad \ldots \quad y' \]

\[ \text{measure (and others)} \]

\( y \quad \ldots \quad y' \)

Token led to \( y \) with dim. \( n \)

\( (f(x)(y))_{m,n}(\rho) \in D_n \)

for each \( n \)
Quantum Geometry of Interaction

\[ [M] = M \]

1 2 3 4 ...

(countably many)
Quantum
Geometry of Interaction

Not just a token/particle, but quantum state!

\[
\begin{bmatrix} M \end{bmatrix} = M
\]
Quantum Geometry of Interaction

Not just a token/particle, but quantum state!
Quantum Geometry of Interaction

\[ [M] = M \]

Not just a token/particle, but quantum state!

“Quantum Data”

Hasuo (Tokyo)
Quantum

Geometry of Interaction

$[M] = M$

"Quantum Data"

Not just a token/particle, but quantum state!

"Classical Control"

(countably many)
Quantum Geometry of Interaction

\[ [M] = M \]

Not just a token/particle, but quantum state!

"Quantum Data"

"in which pipe"

(measurement \(\rightarrow\) case-distinction) leads a token to different pipes

"Classical Control"

(countably many)
Indeed...

* The monad $\mathcal{Q}$ qualifies as a “branching monad”

* The quantum GoI workflow leads to a linear category $\text{PER}_\mathcal{Q}$

* From which we construct an adequate denotational model
End of the Story?

* No! All the technicalities are yet to come:
  * CPS-style interpretation (for partial measurement)
  * Result type: a final coalgebra in $\text{PER}_Q$
  * Admissible PERs for recursion
  * ...

* On the next occasion :-)
Conclusion: the Categorical GoI Workflow

Branching monad \( B \)

\[ \text{Coalgebraic trace semantics} \]

Traced monoidal category \( C \)

+ other constructs \( \Rightarrow \) “GoI situation” [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category
Conclusion: the Categorical GoI Workflow

Branching monad $B$

Coalgebraic trace semantics

Traced monoidal category $\mathcal{C}$

+ other constructs $\Rightarrow$ “GoI situation” [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

Quantum branching monad

Quantum TSMC

Quantum LCA

Model of quantum language

Thank you for your attention!

Ichiro Hasuo (Dept. CS, U Tokyo)

http://www-mmm.is.s.u-tokyo.ac.jp/~ichiro/