A Programming Language for Quantum Control and Indefinite Causal Orders

Kathleen Barsse

Inria team Mocqua, LORIA, Nancy

Journées des doctorants Mocqua

March 26, 2026

Overview

1. Background: quantum states, channels and supermaps
2. Quantum control
3. Syntax and type system
4. Operational semantics
5. Denotational semantics (work in progress)
6. Conclusion

Background: quantum states, channels and supermaps

Quantum states

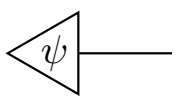
- The state of a quantum system is represented by a unit vector in a Hilbert space $\mathcal{H}$.
- Example: the state of a qubit is a vector in $\mathbb{C}^2$.

$$|0\rangle := \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle := \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Superpositions of $|0\rangle$ and $|1\rangle$:

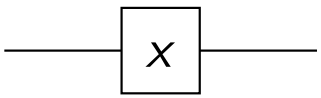
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

- Diagrammatically:



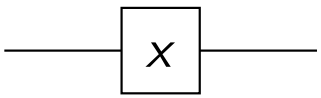
Quantum channels

- Transformations of quantum states are *quantum channels*, i.e. completely positive trace preserving maps.

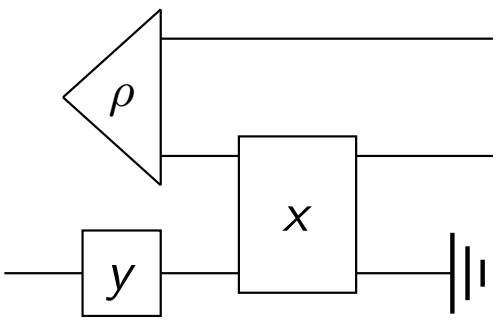


Quantum channels

- Transformations of quantum states are *quantum channels*, i.e. completely positive trace preserving maps.

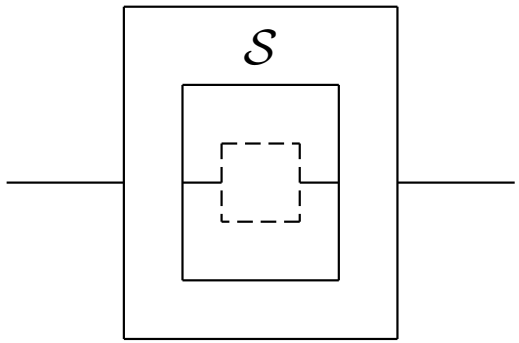


- Diagrams can be composed and interpreted as new quantum channels:



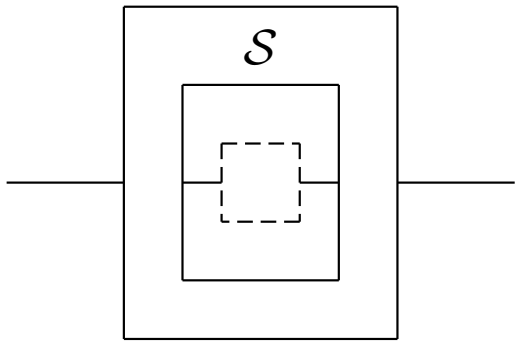
Quantum supermaps

- Quantum supermaps map quantum channels to quantum channels.

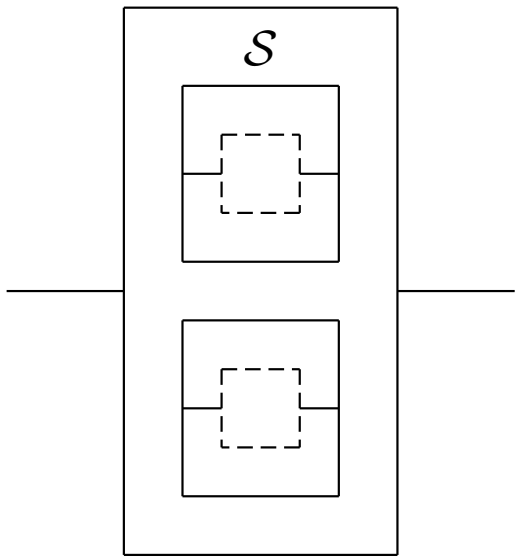


Quantum supermaps

- Quantum supermaps map quantum channels to quantum channels.



- With multiple input channels:



Quantum control

Classical and quantum control

- **"Quantum data, classical control"**

  The execution flow is based on classical information (for instance measurement outcomes).

Classical and quantum control

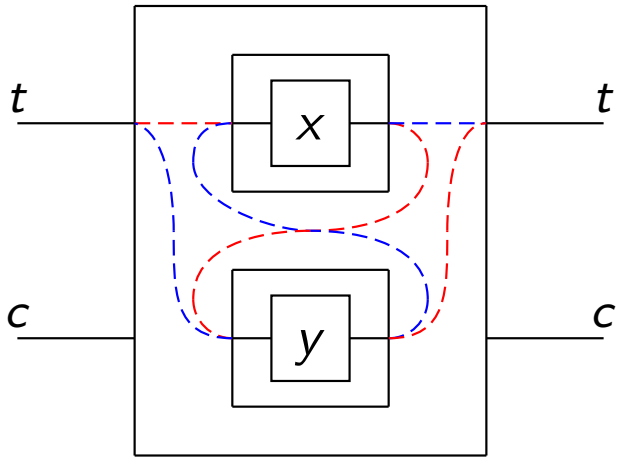
- **"Quantum data, classical control"**

  The execution flow is based on classical information (for instance measurement outcomes).

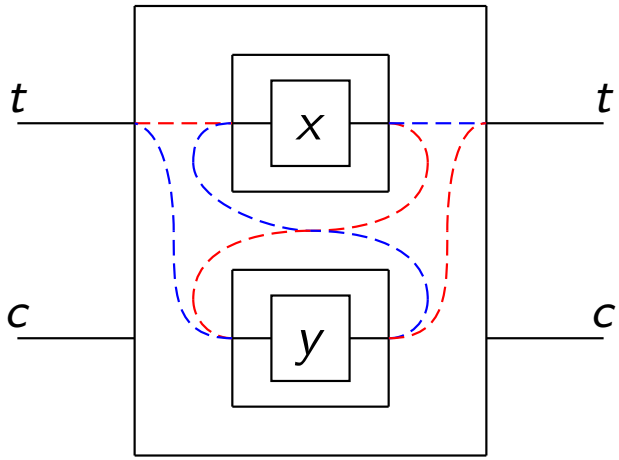
- **"Quantum data, quantum control"**

  The execution flow is based on quantum information. We allow for the superposition of processes during the execution of a program.

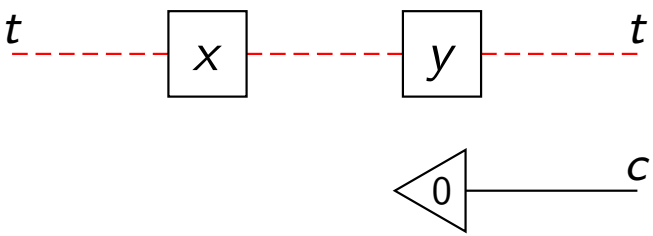
Example: the quantum switch



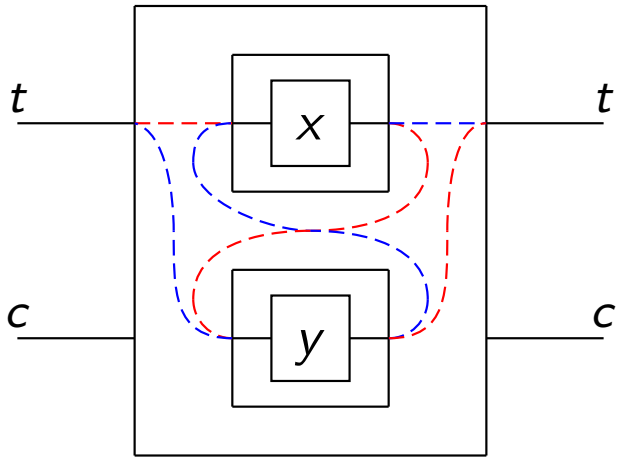
Example: the quantum switch



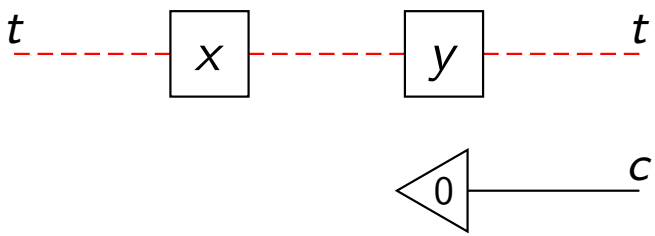
- If c is in state $|0\rangle$:



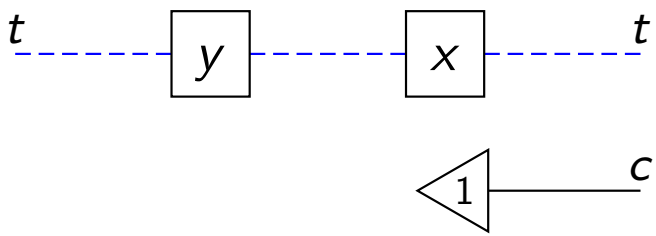
Example: the quantum switch



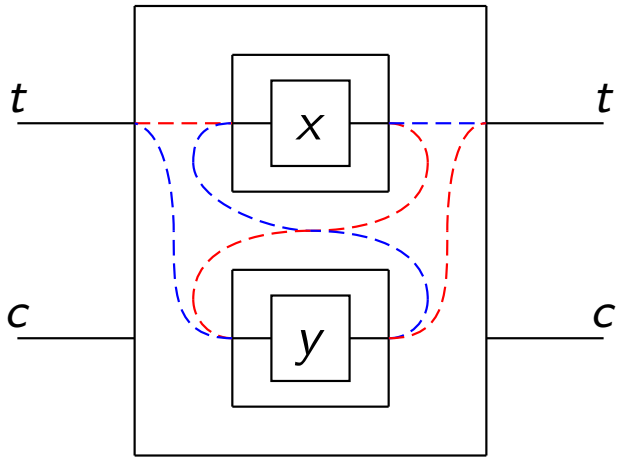
- If c is in state $|0\rangle$:



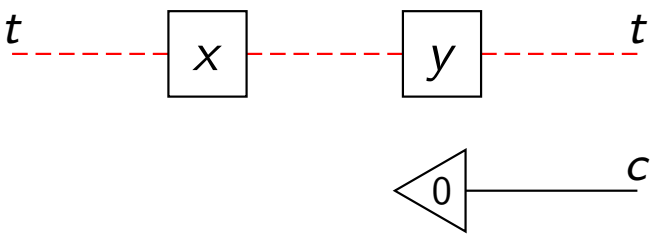
- If c is in state $|1\rangle$:



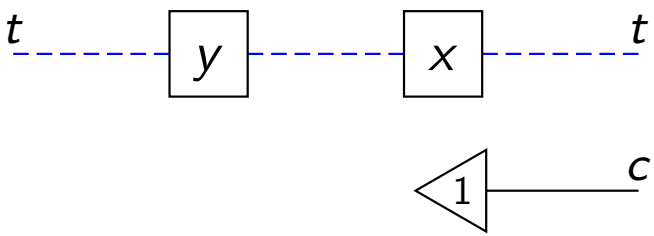
Example: the quantum switch



- If c is in state $|0\rangle$:

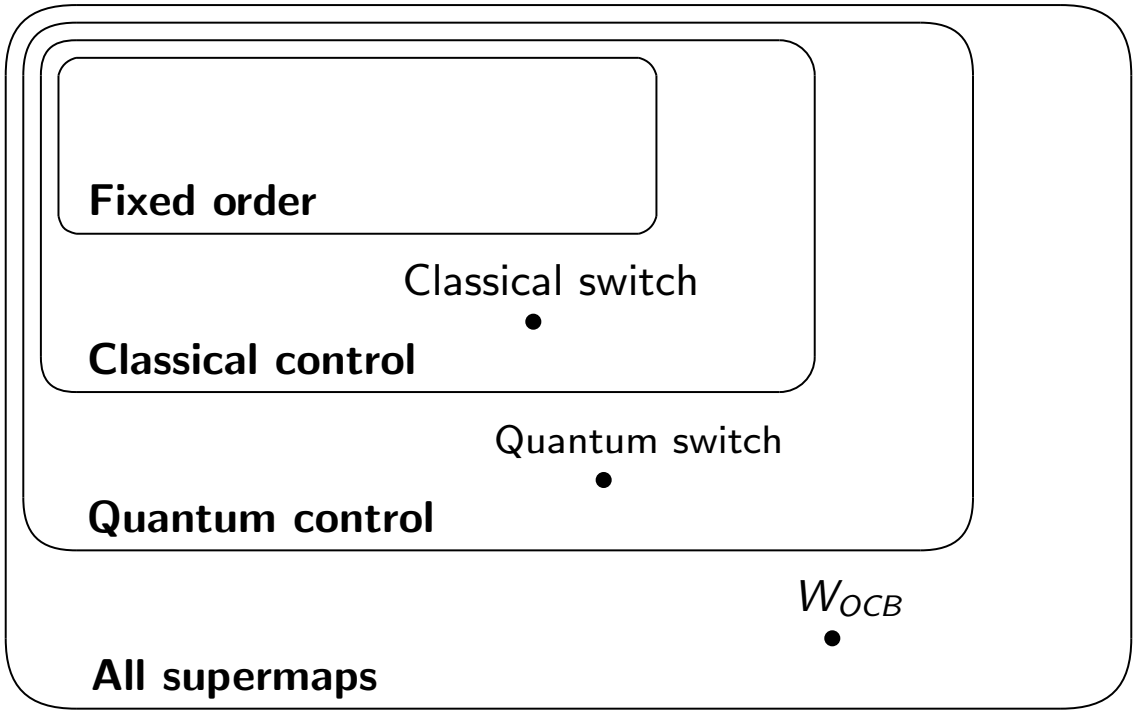


- If c is in state $|1\rangle$:

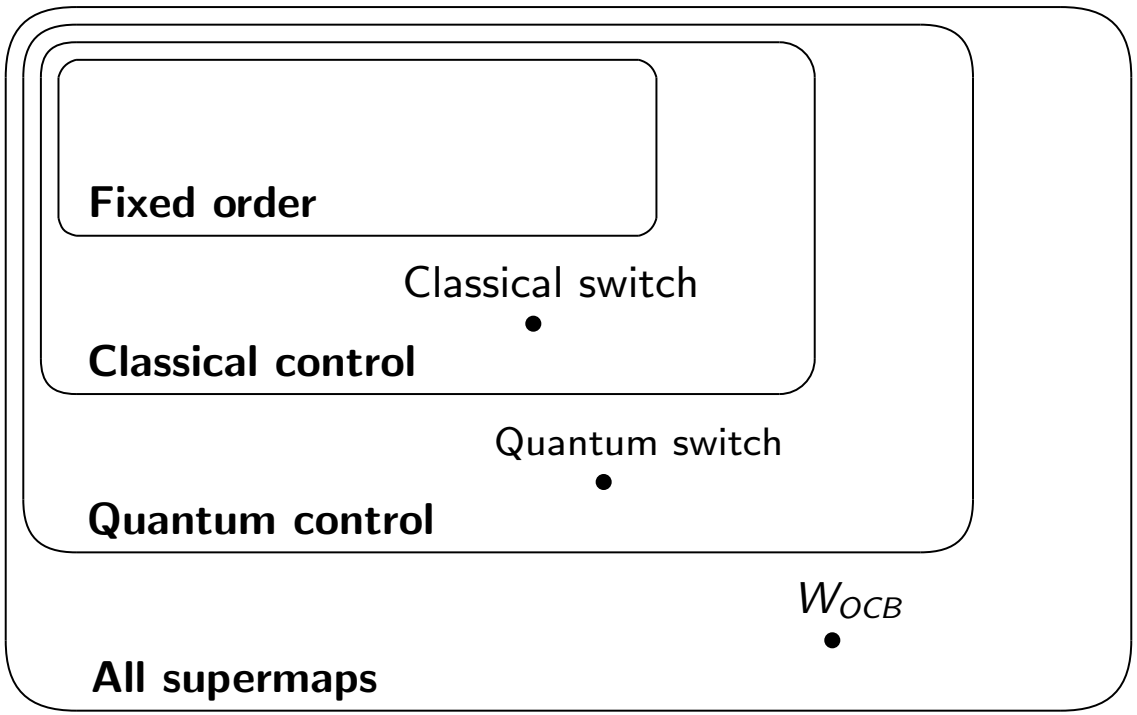


- If c is in a superposition, we get a superposition of the two orderings. x and y are said to be in an *indefinite causal order* (ICO).

Supermaps with quantum control



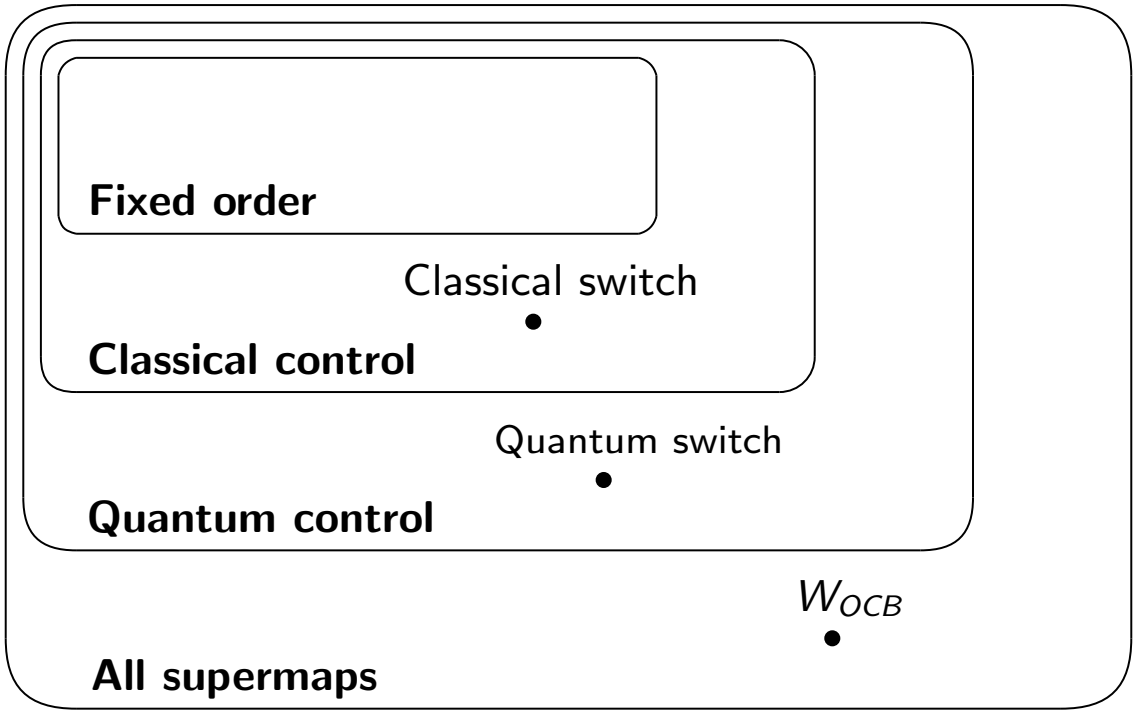
Supermaps with quantum control



Many examples of computational advantage from quantum control:

- Mateus Araújo, Fabio Costa, and Časlav Brukner. "Computational advantage from quantum-controlled ordering of gates". In: *Physical review letters* 113.25 (2014)
- Alastair A. Abbott et al. "Communication through coherent control of quantum channels". In: *Quantum* 4 (2020), p. 333
- Hlér Kristjánsson et al. "Exponential separation in quantum query complexity of the quantum switch with respect to simulations with standard quantum circuits". In: *arXiv preprint arXiv:2409.18420* (2024)
- ...

Supermaps with quantum control



Our goal is to develop a programming language with quantum control and indefinite causal orders.

Syntax and type system

A higher-order programming language

$$
\begin{aligned}
M, N, P \;::=\; & \overbrace{x \mid MN \mid \lambda x.M}^{\text{linear}} \;\mid\; \overbrace{u \mid M^* N \mid \lambda^* u.M}^{\text{unrestricted}} \\
& \mid \langle M, N \rangle \mid \mathtt{let}\ \langle x, y \rangle = M\ \mathtt{in}\ N \mid () \mid M; N \\
& \mid U \mid |0\rangle \mid |1\rangle \\
& \mid \mathtt{meas}\ P\ \{0 \to M \mid 1 \to N\} \\
& \mid \mathtt{qcase}\ P\ \{0 \to M \mid 1 \to N\} \\
& \mid \mathtt{letrec}\ f\ x = M \mid \mathtt{letrec}^*\ f\ u = M
\end{aligned}
$$

A higher-order programming language

$$
\begin{aligned}
M, N, P \;::=\; & \overbrace{x \mid MN \mid \lambda x.M}^{\text{linear}} \;\mid\; \overbrace{u \mid M^* N \mid \lambda^* u.M}^{\text{unrestricted}} \\
& \mid \langle M, N \rangle \mid \texttt{let}\ \langle x, y \rangle = M\ \texttt{in}\ N \mid () \mid M; N \\
& \mid U \mid |0\rangle \mid |1\rangle \\
& \mid \texttt{meas}\ P\ \{0 \to M \mid 1 \to N\} \\
& \mid \texttt{qcase}\ P\ \{0 \to M \mid 1 \to N\} \\
& \mid \texttt{letrec}\ f\ x = M \mid \texttt{letrec}^*\ f\ u = M
\end{aligned}
$$

- In $\lambda x.M$ there is exactly one occurence of x in M.
- In $\lambda^* u.M$ there is at least one occurence of u in M.

A higher-order programming language

$$
\begin{aligned}
M, N, P \;::=\; & \overbrace{x \mid MN \mid \lambda x.M}^{\text{linear}} \;\mid\; \overbrace{u \mid M^* N \mid \lambda^* u.M}^{\text{unrestricted}} \\
& \mid \langle M, N \rangle \mid \texttt{let}\ \langle x, y \rangle = M\ \texttt{in}\ N \mid () \mid M; N \\
& \mid U \mid |0\rangle \mid |1\rangle \\
& \mid \texttt{meas}\ P\ \{0 \to M \mid 1 \to N\} \\
& \mid \texttt{qcase}\ P\ \{0 \to M \mid 1 \to N\} \\
& \mid \texttt{letrec}\ f\ x = M \mid \texttt{letrec}^*\ f\ u = M
\end{aligned}
$$

- In $\lambda x.M$ there is exactly one occurence of x in M.
- In $\lambda^* u.M$ there is at least one occurence of u in M.

- In $\texttt{letrec}\ f\ x = M$ there is exactly one occurence of x in M.
- In $\texttt{letrec}^*\ f\ u = M$ there is at at least one occurence of u in M.

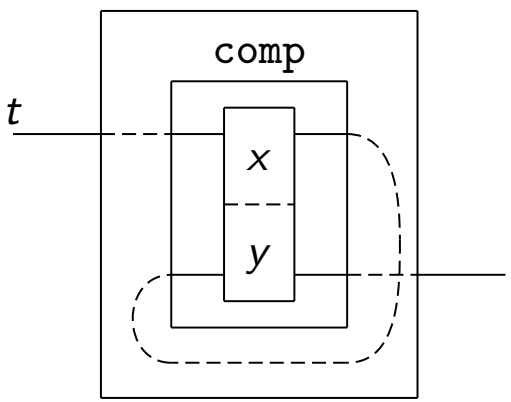
Examples

$$\texttt{discard} := \lambda c.\texttt{meas}\ c\ \{0 \to () \mid 1 \to ()\} \qquad (\text{Not } \lambda c.()\ !)$$

Examples

$$\texttt{discard} \;:=\; \lambda c.\texttt{meas}\ c\ \{0 \to () \mid 1 \to ()\} \qquad (\text{Not } \lambda c.()\ !)$$

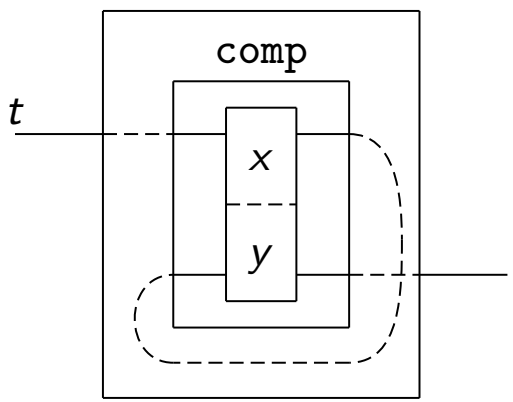
$$\texttt{comp} \;:=\; \lambda z.\texttt{let}\ \langle x, y\rangle = z\ \texttt{in}\ \lambda t.y(xt)$$



Examples

$$\texttt{discard} \;:=\; \lambda c.\texttt{meas}\ c\ \{0 \to () \mid 1 \to ()\} \qquad (\text{Not } \lambda c.()\ !)$$

$$\texttt{comp} \;:=\; \lambda z.\texttt{let}\ \langle x, y\rangle = z\ \texttt{in}\ \lambda t.y(xt)$$



$$\texttt{switch} \;:=\; \lambda\langle x, y\rangle.\lambda c.\texttt{qcase}\ c\ \{0 \to \texttt{comp}\ \langle x, y\rangle \mid 1 \to \texttt{comp}\ \langle y, x\rangle\}$$

Linear type system

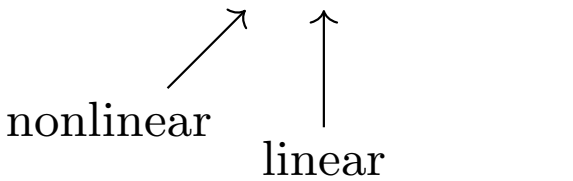
$$\mathit{Type} \quad A, B \quad ::= \quad \mathbf{1} \mid q \mid A \otimes B \mid A \multimap B \mid A \Rightarrow B$$

Linear type system

$$\textit{Type} \quad A, B \quad ::= \quad \mathbf{1} \mid q \mid A \otimes B \mid A \multimap B \mid A \Rightarrow B$$

- Typing judgements:

$$\Gamma; \Delta \vdash M : A$$



Linear type system

$$\textit{Type} \quad A, B \quad ::= \quad \mathbf{1} \mid q \mid A \otimes B \mid A \multimap B \mid A \Rightarrow B$$

- Typing judgements:

$$\Gamma; \Delta \vdash M : A$$

nonlinear

linear

- Some typing rules:

$$\frac{}{\cdot; \cdot \vdash U : q \multimap q} \qquad \frac{\Gamma; \Delta \vdash M : A \qquad \Gamma'; \Delta' \vdash N : B}{\Gamma \cup \Gamma'; \Delta \uplus \Delta' \vdash \langle M, N \rangle : A \otimes B}$$

Linear type system

$$\textit{Type} \quad A, B \quad ::= \quad \mathbf{1} \mid q \mid A \otimes B \mid A \multimap B \mid A \Rightarrow B$$

- Typing judgements:

$$\Gamma; \Delta \vdash M : A$$

nonlinear, linear

- Some typing rules:

$$\frac{}{\cdot; \cdot \vdash U : q \multimap q} \qquad \frac{\Gamma; \Delta \vdash M : A \qquad \Gamma'; \Delta' \vdash N : B}{\Gamma \cup \Gamma'; \Delta \uplus \Delta' \vdash \langle M, N \rangle : A \otimes B}$$

$$\frac{\Gamma; (\Delta, x : A) \vdash M : B}{\Gamma; \Delta \vdash \lambda x.M : A \multimap B}$$

Examples

- $\texttt{discard} := \lambda c.\texttt{meas}\ c\ \{0 \to () \mid 1 \to ()\}$
  We can derive
  $$\cdot;\cdot \vdash \texttt{discard} : q \multimap \mathbf{1}$$

Examples

- $\texttt{discard} := \lambda c.\texttt{meas}\ c\ \{0 \to () \mid 1 \to ()\}$
  We can derive
  $$\cdot;\cdot \vdash \texttt{discard} : q \multimap \mathbf{1}$$

- $\texttt{comp} := \lambda z.\texttt{let}\ \langle x, y\rangle = z\ \texttt{in}\ \lambda t.y(xt)$
  We can derive
  $$\cdot;\cdot \vdash \texttt{comp} : (q \multimap q) \otimes (q \multimap q) \multimap q \multimap q$$

Examples

- $\texttt{discard} := \lambda c.\texttt{meas}\ c\ \{0 \to () \mid 1 \to ()\}$
  We can derive
  $$\cdot;\cdot \vdash \texttt{discard} : q \multimap \mathbf{1}$$

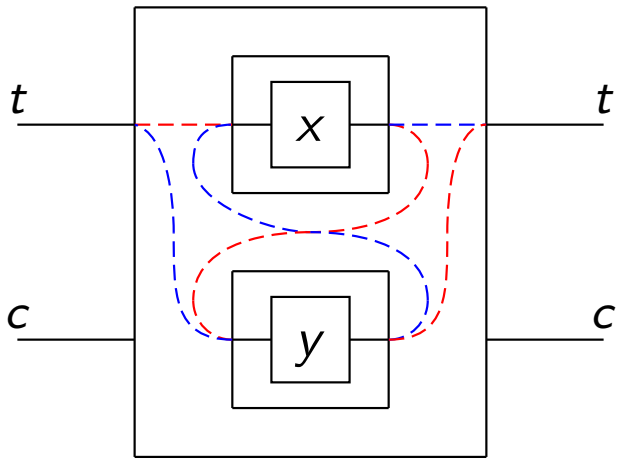
- $\texttt{comp} := \lambda z.\texttt{let}\ \langle x, y\rangle = z\ \texttt{in}\ \lambda t.y(xt)$
  We can derive
  $$\cdot;\cdot \vdash \texttt{comp} : (q \multimap q) \otimes (q \multimap q) \multimap q \multimap q$$

- $\texttt{switch} := \lambda\langle x, y\rangle.\lambda c.\texttt{qcase}\ c\ \{0 \to \texttt{comp}\ \langle x, y\rangle \mid 1 \to \texttt{comp}\ \langle y, x\rangle\}$
  We can derive
  $$\cdot;\cdot \vdash \texttt{switch} : (q \multimap q) \otimes (q \multimap q) \multimap q \multimap q \otimes (q \multimap q)$$

Operational semantics

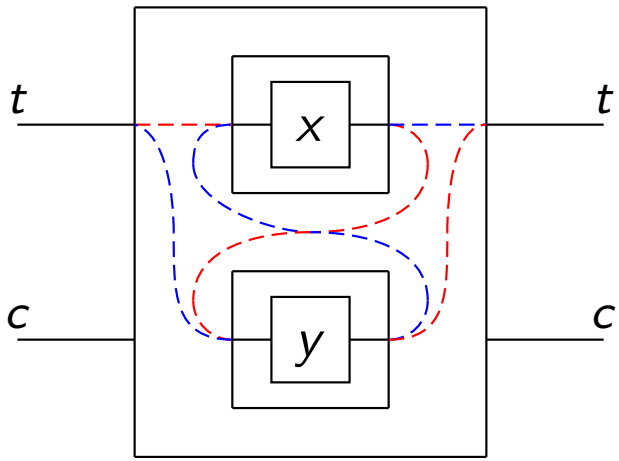
How to define the operational semantics?

$$\texttt{switch} := \lambda\langle x, y\rangle.\lambda c.\texttt{qcase}\ c\ \{0 \to \texttt{comp}\ \langle x, y\rangle \mid 1 \to \texttt{comp}\ \langle y, x\rangle\}$$



How to define the operational semantics?

$$\texttt{switch} \;:=\; \lambda\langle x, y\rangle.\lambda c.\texttt{qcase}\ c\ \{0 \to \texttt{comp}\ \langle x, y\rangle \mid 1 \to \texttt{comp}\ \langle y, x\rangle\}$$

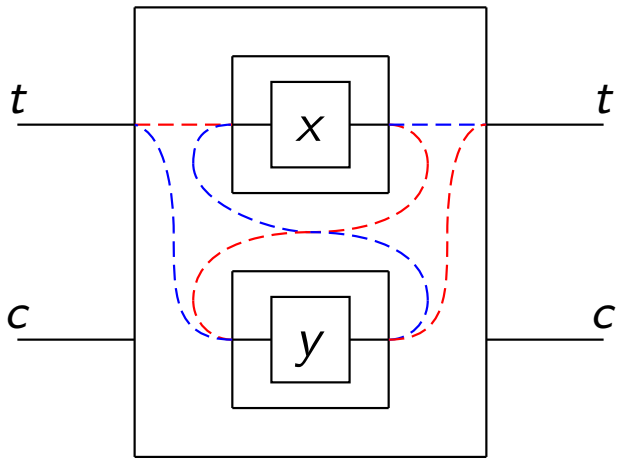


Each alternative path is a separate evolution. This will be written using **sums of terms**. We want:

$$\texttt{switch}\,\langle c, d\rangle\,(H|0\rangle) \quad \to^* \quad \frac{1}{\sqrt{2}}\cdot\langle|0\rangle, \texttt{comp}\ \langle c, d\rangle\rangle + \frac{1}{\sqrt{2}}\cdot\langle|1\rangle, \texttt{comp}\ \langle d, c\rangle\rangle$$

How to define the operational semantics?

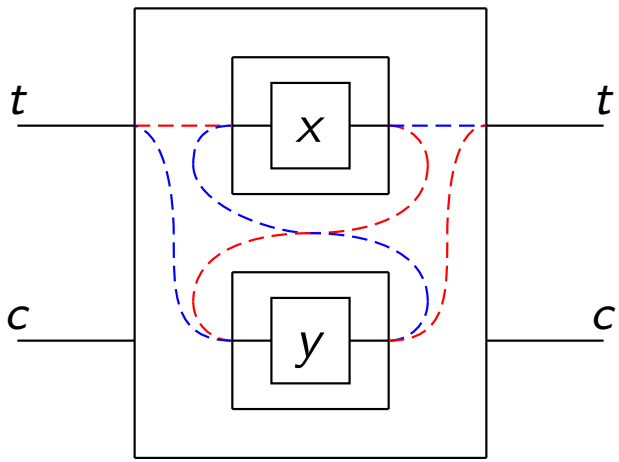
$$\texttt{switch} := \lambda\langle x, y\rangle.\lambda c.\texttt{qcase}\ c\ \{0 \to \texttt{comp}\ \langle x, y\rangle \mid 1 \to \texttt{comp}\ \langle y, x\rangle\}$$



$$\frac{1}{\sqrt{2}} \cdot \langle |0\rangle, \texttt{comp}\ \langle c, d\rangle\rangle + \frac{1}{\sqrt{2}} \cdot \langle |1\rangle, \texttt{comp}\ \langle d, c\rangle\rangle$$

How to define the operational semantics?

$$\texttt{switch} := \lambda\langle x, y\rangle.\lambda c.\texttt{qcase}\ c\ \{0 \to \texttt{comp}\ \langle x, y\rangle \mid 1 \to \texttt{comp}\ \langle y, x\rangle\}$$

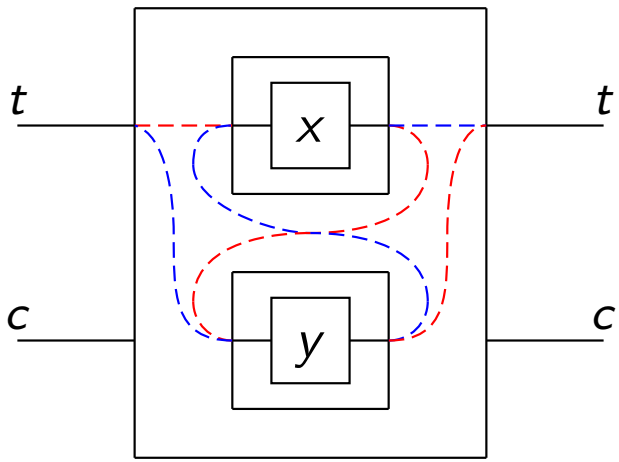


$$\frac{1}{\sqrt{2}} \cdot \langle |0\rangle, \texttt{comp}\ \langle c, d\rangle\rangle + \frac{1}{\sqrt{2}} \cdot \langle |1\rangle, \texttt{comp}\ \langle d, c\rangle\rangle$$

Problem: What if $d = \lambda t.\texttt{meas}\ t\ (0 \to |0\rangle \mid 1 \to |1\rangle)$?

How to define the operational semantics?

$$\texttt{switch} := \lambda\langle x, y\rangle.\lambda c.\texttt{qcase}\ c\ \{0 \to \texttt{comp}\ \langle x, y\rangle \mid 1 \to \texttt{comp}\ \langle y, x\rangle\}$$



$$\frac{1}{\sqrt{2}} \cdot \langle |0\rangle, \texttt{comp}\ \langle c, d\rangle\rangle + \frac{1}{\sqrt{2}} \cdot \langle |1\rangle, \texttt{comp}\ \langle d, c\rangle\rangle$$

Problem: What if $d = \lambda t.\texttt{meas}\ t\ (0 \to |0\rangle \mid 1 \to |1\rangle)$?

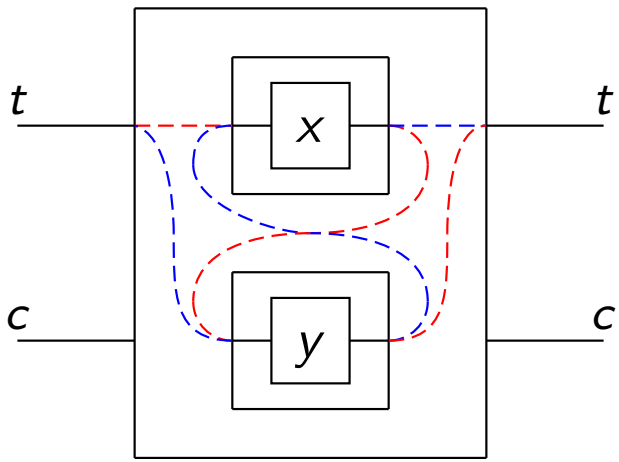
The measurement outcome should be the same in each copy.

We introduce **labels**:

$$\frac{1}{\sqrt{2}} \cdot \langle |0\rangle, \texttt{comp}\ \langle c, [d]^{\ell}\rangle\rangle + \frac{1}{\sqrt{2}} \cdot \langle |1\rangle, \texttt{comp}\ \langle [d]^{\ell}, c\rangle\rangle$$

How to define the operational semantics?

$$\texttt{switch} := \lambda\langle x, y\rangle.\lambda c.\texttt{qcase}\ c\ \{0 \to \texttt{comp}\ \langle x, y\rangle \mid 1 \to \texttt{comp}\ \langle y, x\rangle\}$$



Two main features of the operational semantics:

1. linear combinations of terms;
2. labels for probabilistic processes.

Adding labels

We have access to a set of label symbols $\mathscr{L}$.
Given a term M, $[M]^\ell$ ($\ell \in \mathscr{L}$) is obtained by replacing:

$$\begin{array}{rcl} \texttt{meas} & \rightsquigarrow & \texttt{meas}^\ell \\ u & \rightsquigarrow & u^\ell \end{array}$$

Adding labels

We have access to a set of label symbols $\mathscr{L}$.
Given a term M, $[M]^\ell$ ($\ell \in \mathscr{L}$) is obtained by replacing:

$$\begin{array}{rcl} \texttt{meas} & \rightsquigarrow & \texttt{meas}^\ell \\ u & \rightsquigarrow & u^\ell \end{array}$$

For example:

$$[\texttt{meas}\ x\ \{0 \to |1\rangle \mid 1 \to u|0\rangle\}]^\ell \quad = \quad \texttt{meas}^\ell\ x\ \{0 \to |1\rangle \mid 1 \to u^\ell|0\rangle\}$$

Adding superpositions of terms

Term superpositions $\quad \vec{M}, \vec{N} ::= \vec{0} \mid M \mid \vec{M} + \vec{N} \mid \alpha \cdot \vec{M}$

Adding superpositions of terms

$$\textit{Term superpositions} \qquad \vec{M}, \vec{N} \ ::= \ \vec{0} \mid M \mid \vec{M} + \vec{N} \mid \alpha \cdot \vec{M}$$

Sums of terms only appear at the topmost level:

$$\cancel{(\lambda x.x)(\alpha|0\rangle + \beta|1\rangle)} \qquad \rightsquigarrow \qquad \alpha(\lambda x.x)|0\rangle + \beta(\lambda x.x)|1\rangle$$

Reduction rules

Some reduction rules for terms without sums:

$$\frac{N \to \vec{N'}}{MN \to M\vec{N'}} \qquad \frac{M \to \vec{M'}}{MV \to \vec{M'}V} \qquad \frac{k \in \{0,1\} \qquad [\![U]\!] = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}}{U|k\rangle \to u_{0k} \cdot |0\rangle + u_{1k} \cdot |1\rangle}$$

Reduction rules

Some reduction rules for terms without sums:

$$\frac{N \to \vec{N'}}{MN \to M\vec{N'}} \qquad \frac{M \to \vec{M'}}{MV \to \vec{M'}V} \qquad \frac{k \in \{0,1\} \qquad [\![U]\!] = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}}{U|k\rangle \to u_{0k} \cdot |0\rangle + u_{1k} \cdot |1\rangle}$$

$$\frac{\ell \in \mathscr{L} \text{ is fresh}}{(\lambda x.M)V \to M[[V]^{\ell}/x]} \qquad \frac{k \in \{0,1\}}{\texttt{qcase}\ |k\rangle\ \{0 \to M_0 \mid 1 \to M_1\} \to \langle |k\rangle, M_k\rangle}$$

Reduction rules

Some reduction rules for terms without sums:

$$\frac{N \to \vec{N'}}{MN \to M\vec{N'}} \qquad \frac{M \to \vec{M'}}{MV \to \vec{M'}V} \qquad \frac{k \in \{0,1\} \qquad [\![U]\!] = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}}{U|k\rangle \to u_{0k} \cdot |0\rangle + u_{1k} \cdot |1\rangle}$$

$$\frac{\ell \in \mathscr{L} \text{ is fresh}}{(\lambda x.M)V \to M[[V]^{\ell}/x]} \qquad \frac{k \in \{0,1\}}{\texttt{qcase}\ |k\rangle\ \{0 \to M_0 \mid 1 \to M_1\} \to \langle |k\rangle, M_k\rangle}$$

For terms with sums, we reduce one summand at a time:

$$\frac{M \to \vec{M'}}{\alpha \cdot M + \vec{N} \twoheadrightarrow \alpha \cdot \vec{M'} + \vec{N}} \qquad (+ \text{ consistency hypotheses})$$

An extended language for executions

$\mathscr{T}$ $\subseteq$ $\vec{\mathscr{T}}$

terms of the language

$+$labels
$+$term superpositions

An extended language for executions

$$\mathscr{T} \qquad \subseteq \qquad \vec{\mathscr{T}}$$

terms of the language

$+$labels
$+$term superpositions

$$\underbrace{M_0}_{\text{in } \mathscr{T}} \twoheadrightarrow \underbrace{\vec{M}_1 \twoheadrightarrow \vec{M}_2 \twoheadrightarrow \vec{M}_3 \twoheadrightarrow \cdots}_{\text{in } \vec{\mathscr{T}}}$$

An extended language for executions

$$\mathscr{T} \qquad \subseteq \qquad \vec{\mathscr{T}}$$

terms of the language

$+$labels
$+$term superpositions

$$\underbrace{M_0}_{\text{in } \mathscr{T}} \twoheadrightarrow \overbrace{\vec{M}_1 \twoheadrightarrow \vec{M}_2 \twoheadrightarrow \vec{M}_3 \twoheadrightarrow \cdots}^{\text{in } \vec{\mathscr{T}}}$$

- The extended language is not accessible to the programmer.
- We avoid having to check unitarity.

Properties of the operational semantics

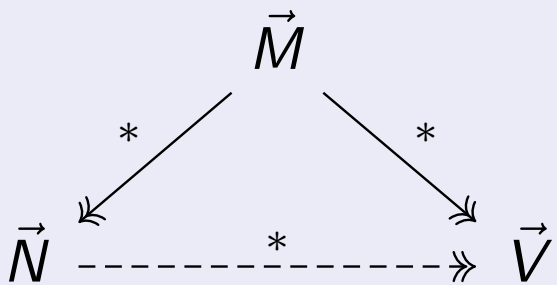
- Standard safety properties:
  - Substitution,
  - Subject reduction,
  - Progress.

Properties of the operational semantics

- Standard safety properties:
  - Substitution,
  - Subject reduction,
  - Progress.
- A property on reductions to values: values are terms from which no reduction is possible. We have:

Reductions to values

Suppose $\vec{M} \twoheadrightarrow^* \vec{N}$ and $\vec{M} \twoheadrightarrow^* \vec{V}$, where $\vec{V}$ is a superposition of values, and $\vec{N}$ and $\vec{V}$ have compatible measurement outcomes. Then $\vec{N} \twoheadrightarrow^* \vec{V}$.
Diagrammatically,



Denotational semantics (work in progress)

What do we need to define?

1. A suitable categorical structure
   - For example we can interpret $\otimes$ with a symmetric monoidal structure.
   - We also need to interpret the connectives $\multimap$ and $\Rightarrow$, and recursive procedures.
2. Find a particular category that has this structure and that can model quantum processes.

First step: the multiplicative fragment

$$
\begin{aligned}
M, N, P \;::=\; & x \mid MN \mid \lambda x.M \mid \cancel{u \mid M^* N \mid \lambda^* u.M} \\
& \mid \langle M, N \rangle \mid \texttt{let}\ \langle x, y \rangle = M\ \texttt{in}\ N \mid () \mid M; N \\
& \mid U \mid |0\rangle \mid |1\rangle \\
& \mid \texttt{meas}\ P\ \{0 \to M \mid 1 \to N\} \\
& \mid \texttt{qcase}\ P\ \{0 \to M \mid 1 \to N\} \\
& \mid \cancel{\texttt{letrec}\ f\ x = M \mid \texttt{letrec}^*\ f\ u = M}
\end{aligned}
$$

Finding a suitable category

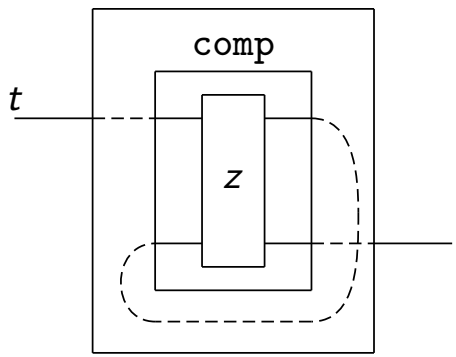
First idea: Interpret programs in **CPM** (the category of completely positive maps).

Finding a suitable category

First idea: Interpret programs in **CPM** (the category of completely positive maps).

However:

$$\lambda z.\texttt{let}\ \langle x, y\rangle = z\ \texttt{in}\ \lambda t.y(xt) \text{ of type } (q \multimap q) \otimes (q \multimap q) \multimap q \multimap q$$

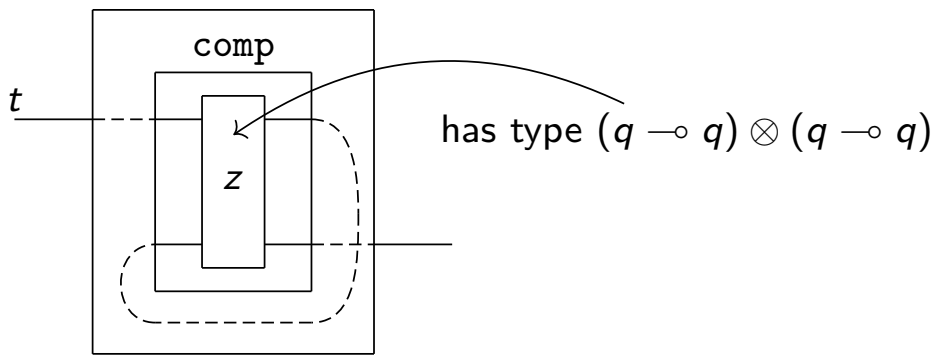


Finding a suitable category

First idea: Interpret programs in **CPM** (the category of completely positive maps).

However:

$$\lambda z.\texttt{let}\ \langle x, y\rangle = z\ \texttt{in}\ \lambda t.y(xt) \text{ of type } (q \multimap q) \otimes (q \multimap q) \multimap q \multimap q$$

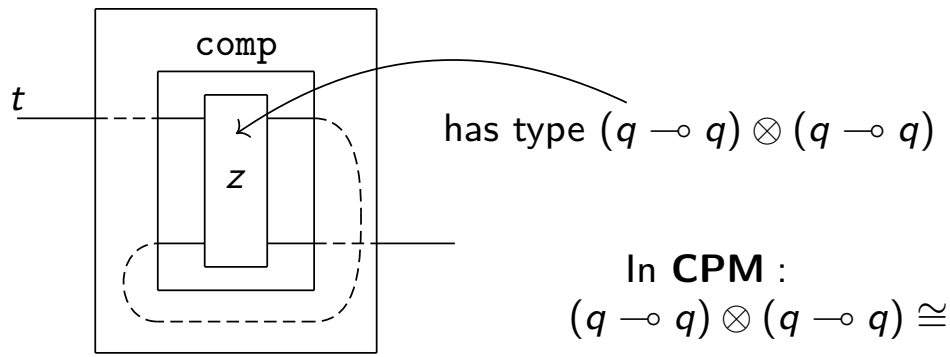


Finding a suitable category

First idea: Interpret programs in **CPM** (the category of completely positive maps).

However:

$$\lambda z.\texttt{let}\ \langle x, y\rangle = z\ \texttt{in}\ \lambda t.y(xt) \text{ of type } (q \multimap q) \otimes (q \multimap q) \multimap q \multimap q$$



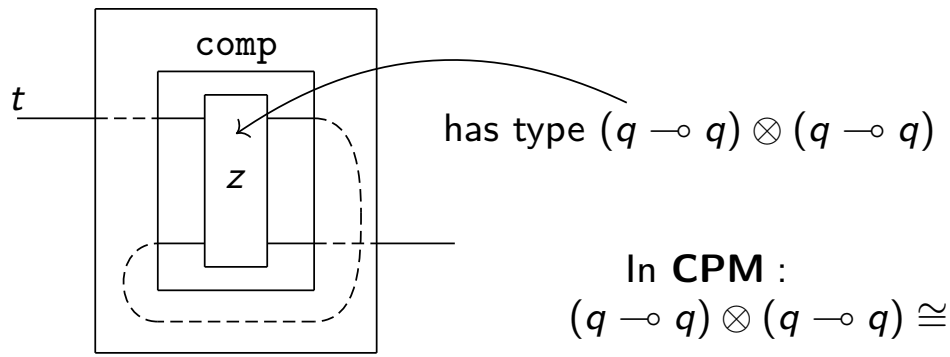
In **CPM** :
$(q \multimap q) \otimes (q \multimap q) \cong (q \otimes q) \multimap (q \otimes q)$

Finding a suitable category

First idea: Interpret programs in **CPM** (the category of completely positive maps).

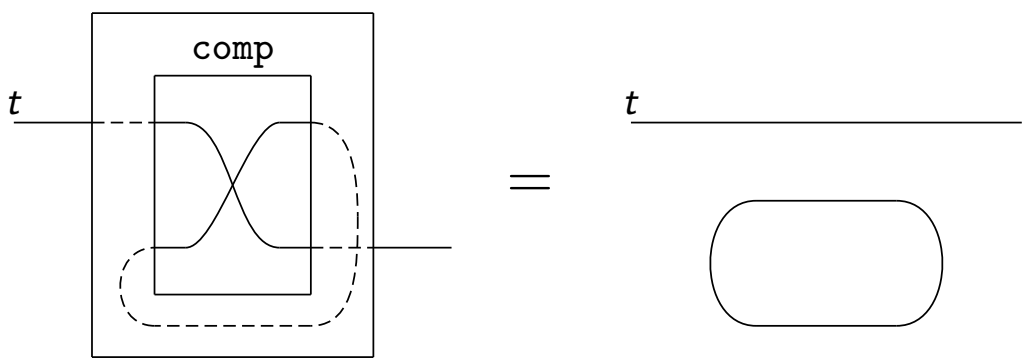
However:

$$\lambda z.\texttt{let}\ \langle x, y\rangle = z\ \texttt{in}\ \lambda t.y(xt) \text{ of type } (q \multimap q) \otimes (q \multimap q) \multimap q \multimap q$$



In **CPM** :
$(q \multimap q) \otimes (q \multimap q) \cong (q \otimes q) \multimap (q \otimes q)$

But this has no physical meaning:



Finding a suitable category

A solution has already been studied: **causal categories**.

Aleks Kissinger and Sander Uijlen. "A categorical semantics for causal structure". In: *Logical Methods in Computer Science* 15 (2019)

We will interpret programs in **Caus**[**CPM**], where

$$(q \multimap q) \otimes (q \multimap q) \subsetneq (q \otimes q) \multimap (q \otimes q)$$

- $\otimes$ models "no-signalling" channels.

Denotational semantics

- Typing judgments now have the form

$$\Delta \vdash M : A$$

- to each type A we assign an object $[\![A]\!]$ of **Caus**[**CPM**];
- to each typing judgment $\Delta \vdash M : A$ we assign a morphism $[\![\Delta]\!] \xrightarrow{f} [\![A]\!]$.

Denotational semantics

- Typing judgments now have the form

$$\Delta \vdash M : A$$

- to each type A we assign an object $[\![A]\!]$ of **Caus[CPM]**;
- to each typing judgment $\Delta \vdash M : A$ we assign a morphism $[\![\Delta]\!] \xrightarrow{f} [\![A]\!]$.

Some rules:

$$\frac{[\![\Delta \vdash M : A]\!] = [\![\Delta]\!] \xrightarrow{f} [\![A]\!] \qquad [\![\Delta' \vdash N : B]\!] = [\![\Delta']\!] \xrightarrow{g} [\![B]\!]}{[\![\Delta \uplus \Delta' \vdash \langle M, N \rangle : A \otimes B]\!] = [\![\Delta]\!] \otimes [\![\Delta']\!] \xrightarrow{f \otimes g} [\![A]\!] \otimes [\![B]\!]}$$

$$\frac{[\![(\Delta, x : A) \vdash M : B]\!] = [\![\Delta]\!] \otimes [\![A]\!] \xrightarrow{f} [\![B]\!]}{[\![\Delta \vdash \lambda x.M : A \multimap B]\!] = [\![\Delta]\!] \xrightarrow{\phi(f)} [\![A]\!] \multimap [\![B]\!]}$$

Remaining questions

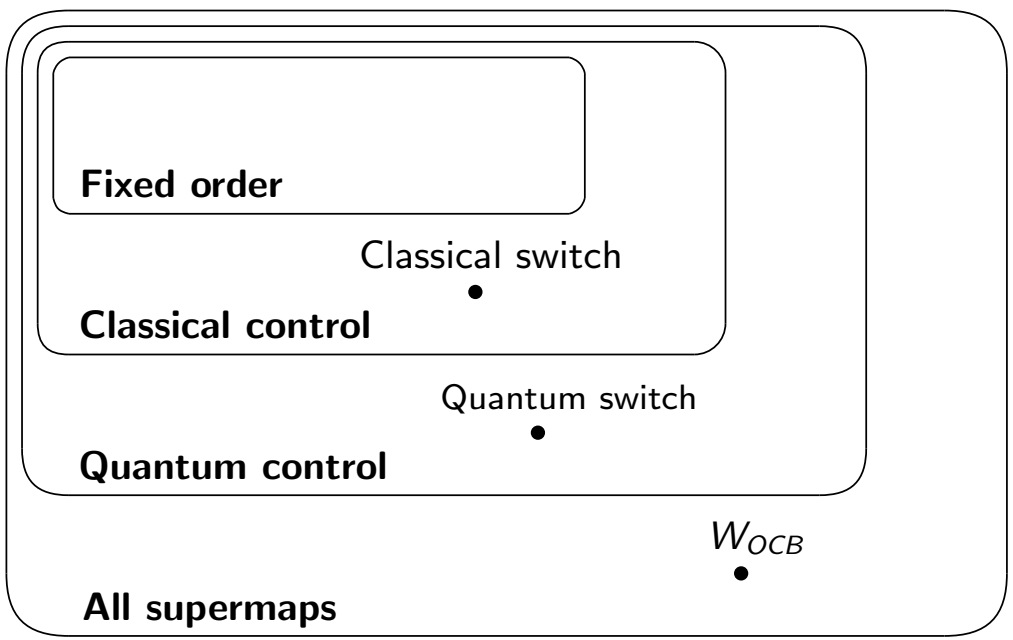
- Interpreting the full language:
  - interpreting $\Rightarrow$;
  - interpreting terms with recursion.

Remaining questions

- Interpreting the full language:
  - interpreting $\Rightarrow$;
  - interpreting terms with recursion.
- Prove adequacy between the semantics.

Remaining questions

- Interpreting the full language:
  - interpreting $\Rightarrow$;
  - interpreting terms with recursion.
- Prove adequacy between the semantics.
- Expressivity of the language?



Conclusion

Conclusion

- We introduced a higher programming language for quantum control.
- The operational semantics uses the following features:
  - Superpositions of terms;
  - Labeling probabilistic processes.
- We can interpret the multiplicative fragment in **Caus**[**CPM**].

Conclusion

- We introduced a higher programming language for quantum control.
- The operational semantics uses the following features:
  - Superpositions of terms;
  - Labeling probabilistic processes.
- We can interpret the multiplicative fragment in **Caus**[**CPM**].

Questions?