Causality in Pure Quantum Computation with Quantum Control

↑slides & paper

LICS 2026. 21 July 2026.

Kengo Hirata, University of Edinburgh, Kyoto University
Takeshi Tsukada, Chiba University

Quantum Control

Quantum *if* statement

qif **x** then **M** else **N**

where **x** : qubit

Quantum Control

Previous talk:

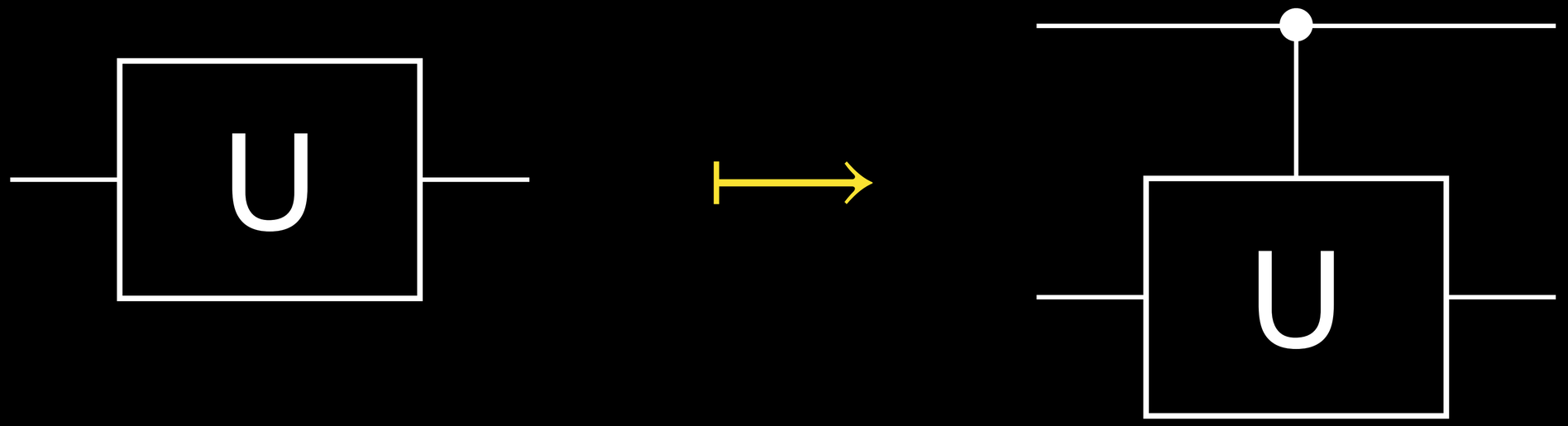
“Quantum Control and General Recursion beyond the Unitary Case”

Next talk:

“One rig to control them all”

Quantum Control

on Quantum Circuits

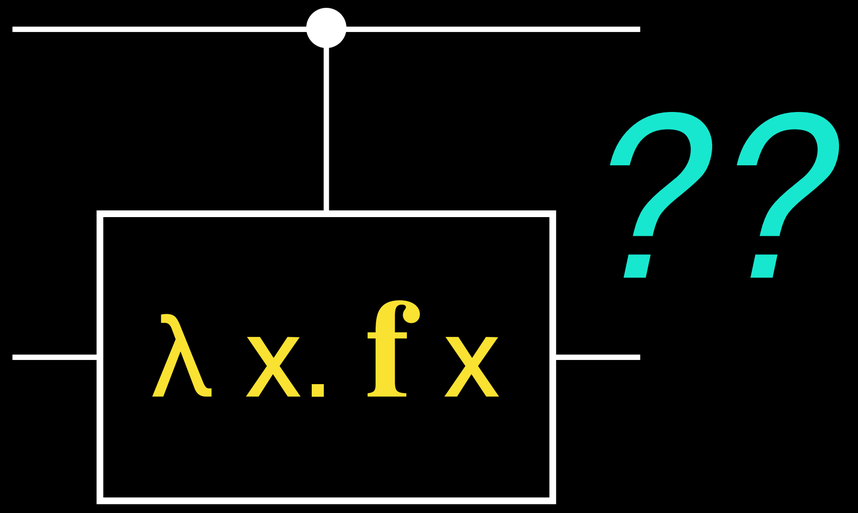


This paper:

Quantum Control

on Quantum **Programs**

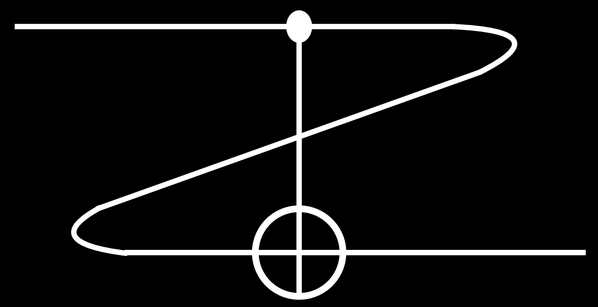
$\lambda\, x.\, f\, x$ $\longmapsto$



Quick Overview

Identify the Causality Problem

⇒ Naive type system allows problematic programs



Solution:

New type system that respects causal relations

type $A \triangleright B$

"A after B"

Categorical semantics justifies it

category ***CausHilb***

The Causality Problem

Naïve Language: Quantum Lambda + “qif”

Quantum lambda calculus

$$\Big(\quad = \text{ Linear lambda } + \text{ constants} \quad \Big)$$

U : unitary

$|0\rangle$: qubit init

+ quantum conditional

qif **M** then **N** else **L**

Naïve Language: Quantum Lambda + “qif”

Types:

$$A ::= \mathrm{unit} \mid \mathrm{qbit} \mid A \otimes B \mid A \multimap B$$

Terms:

$$M ::= () \mid (M, N) \mid \lambda x.\, M \mid \cdots$$

Naïve Language: Quantum Lambda + “qif”

Types:

$$A ::= \mathrm{unit} \mid \mathrm{qbit} \mid A \otimes B \mid A \multimap B$$

Terms:

$$M ::= () \mid (M, N) \mid \lambda x.\, M \mid \cdots$$

$$\frac{}{\vdash\ () : \mathrm{unit}}$$

Naïve Language: Quantum Lambda + “qif”

Types:

$$A ::= \mathrm{unit} \mid \mathrm{qbit} \mid A \otimes B \mid A \multimap B$$

Terms:

$$M ::= () \mid (M, N) \mid \lambda x.\, M \mid \cdots$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma' \vdash N : B}{\Gamma, \Gamma' \vdash (M, N) : A \otimes B}$$

Naïve Language: Quantum Lambda + “qif”

Types:

$$A ::= \mathrm{unit} \mid \mathrm{qbit} \mid A \otimes B \mid A \multimap B$$

Terms:

$$M ::= () \mid (M, N) \mid \lambda x.\, M \mid \cdots$$

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

Naïve Language: Quantum Lambda + “qif”

Types:

$$A ::= \text{unit} \mid \text{qbit} \mid A \otimes B \mid A \multimap B$$

Terms:

$$M ::= () \mid (M, N) \mid \lambda x.\, M \mid \cdots$$
$$\mid |0\rangle \mid U$$

Naïve Language: Quantum Lambda + “qif”

Types:

$$A ::= \text{unit} \mid \text{qbit} \mid A \otimes B \mid A \multimap B$$

Terms:

$$M ::= () \mid (M, N) \mid \lambda x.\, M \mid \cdots$$
$$\mid |0\rangle \mid U$$

$$\frac{}{\vdash |0\rangle : \text{qbit}}$$

Naïve Language: Quantum Lambda + “qif”

Types:

$$A ::= \text{unit} \mid \text{qbit} \mid A \otimes B \mid A \multimap B$$

Terms:

$$M ::= () \mid (M, N) \mid \lambda x.\, M \mid \cdots \mid |0\rangle \mid U$$

$$\frac{U : \text{unitary}}{\vdash U : \text{qbit}^{\otimes n} \multimap \text{qbit}^{\otimes n}}$$

Naïve Language: Quantum Lambda + “qif”

Types:

$$A ::= \text{unit} \mid \text{qbit} \mid A \otimes B \mid A \multimap B$$

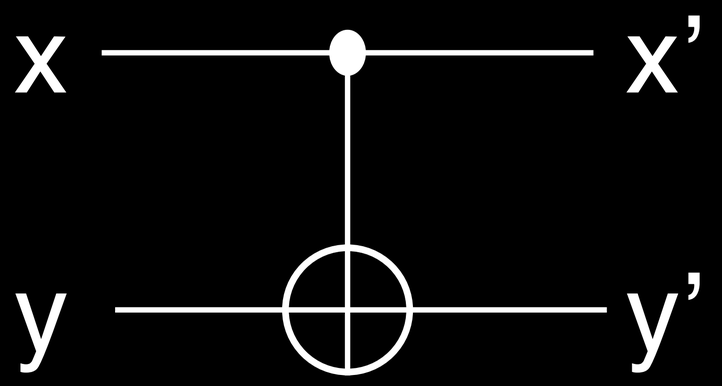
Terms:

$$M ::= () \mid (M, N) \mid \lambda x.\, M \mid \cdots$$
$$\mid |0\rangle \mid U \mid \text{qif } M \text{ then } N \text{ else } L$$

$$\frac{\Gamma \vdash M : \text{qbit} \qquad \Gamma' \vdash N : A \qquad \Gamma' \vdash L : A}{\Gamma, \Gamma' \vdash \text{qif } M \text{ then } N \text{ else } L : \text{qbit} \otimes A}$$

Naïve type for qif

$$\frac{\Gamma \vdash M : \mathrm{qbit} \qquad \Gamma' \vdash N : A \qquad \Gamma' \vdash L : A}{\Gamma, \Gamma' \vdash \mathrm{qif}\ M\ \mathrm{then}\ N\ \mathrm{else}\ L\ :\ \mathrm{qbit} \otimes A}$$



let (x', y') : ***qbit*** ⊗ qbit =
qif x then ***NOT***(y) else y

Problematic Program

$\vdash$ ***NOT***: qbit $\multimap$ qbit $\vdash$ λy. y: qbit $\multimap$ qbit

---

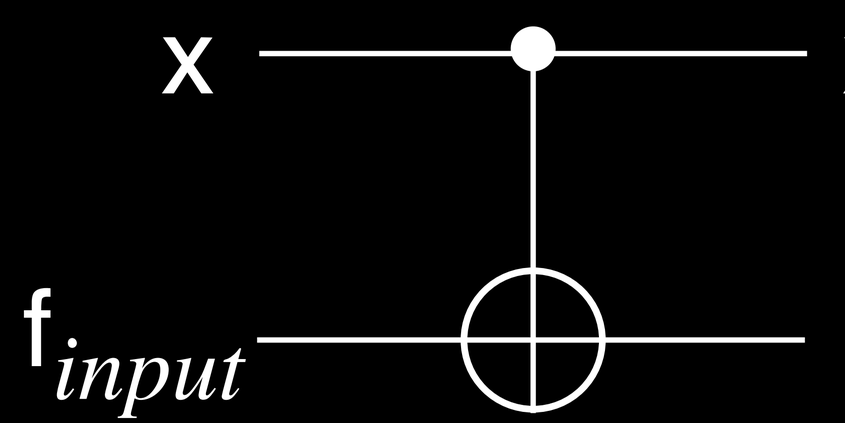
x : qbit $\vdash$ qif x then ***NOT*** else λy. y
: qbit $\otimes$ (qbit $\multimap$ qbit)

---

x : qbit $\vdash$ let (x', f) =
qif x then ***NOT*** else λy. y
in f x' : qbit
(f: qbit $\multimap$ qbit x': qbit)

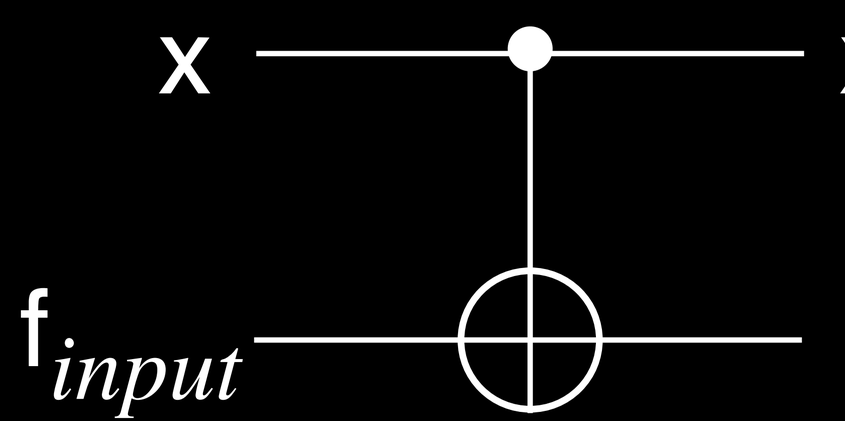
Problematic Program

let (x’, f) = qif x then ***NOT*** else λy. y



Problematic Program

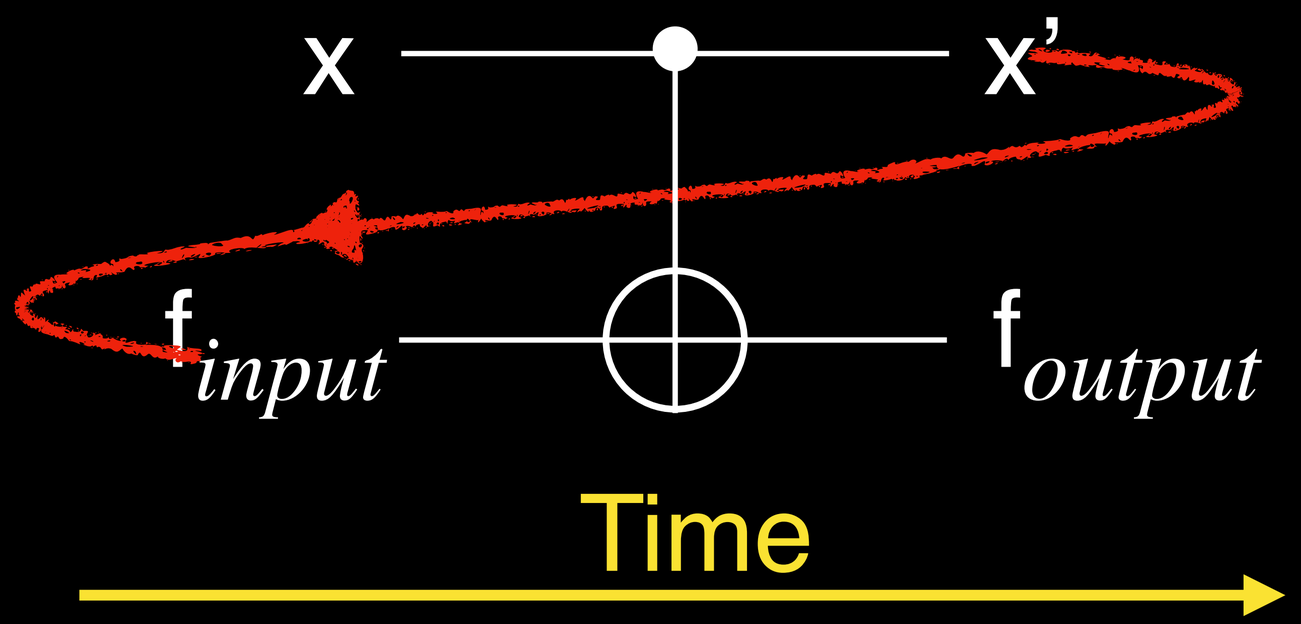
let (x’, f) = qif x then ***NOT*** else λy. y
in f x’ : qbit



Problematic Program

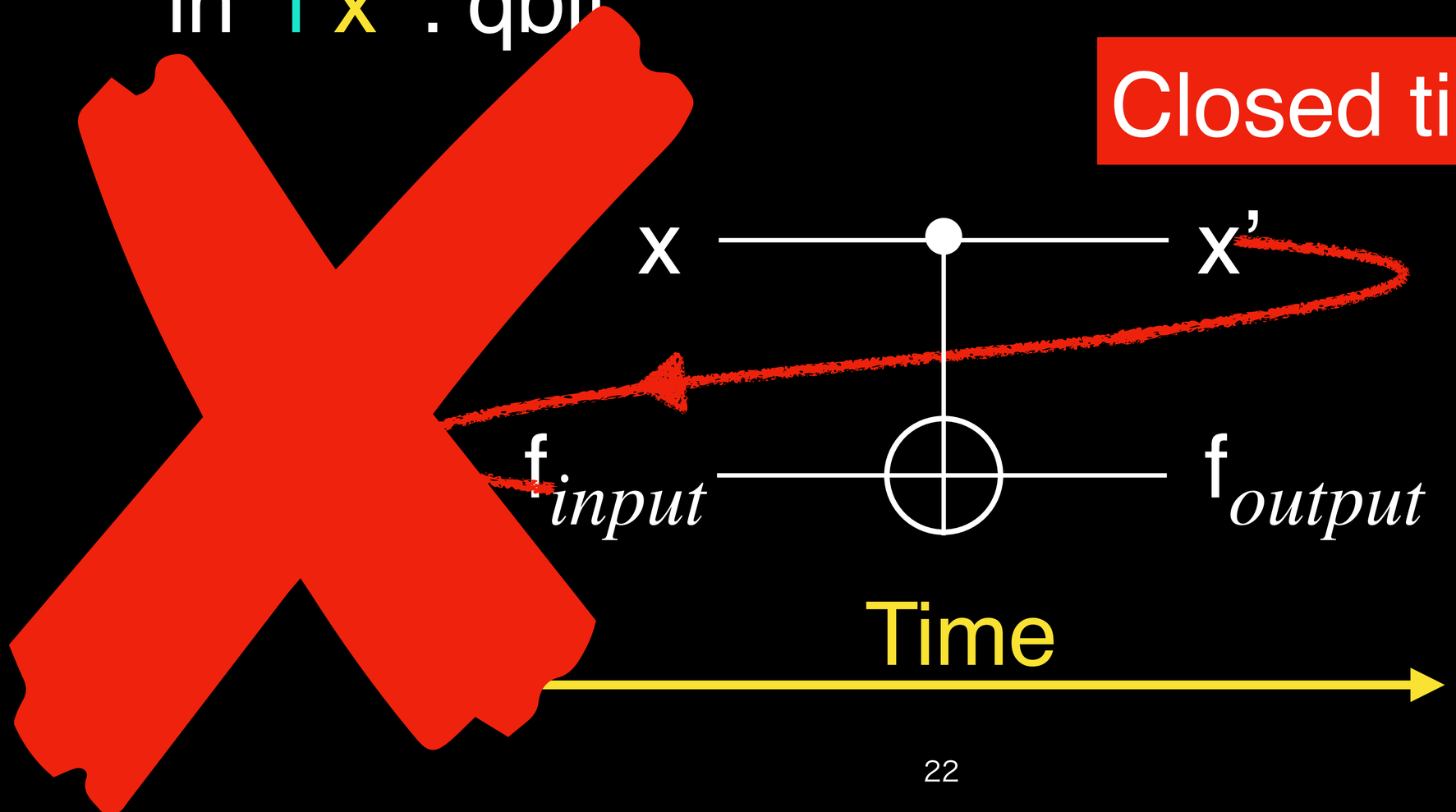
let (x', f) = qif x then ***NOT*** else λy. y
in f x' : qbit

Closed timelike curve



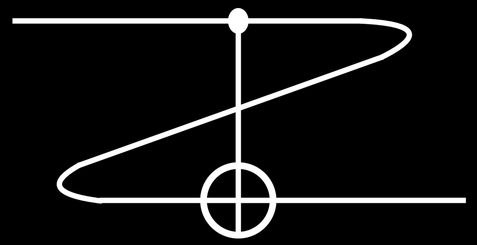
Problematic Program

let (x', f) = qif x then ***NOT*** else λy. y
in f x' : qbit



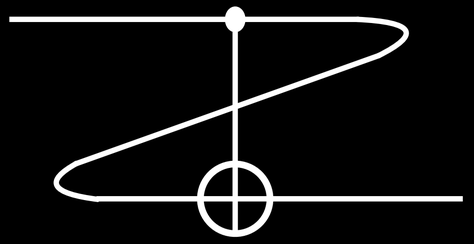
Causality Problem

✗ Operationally wrong



Denotationally?

(category FHilb)

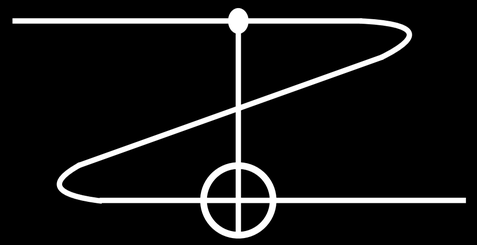


= ⟦ the problematic program ⟧

as string diagram

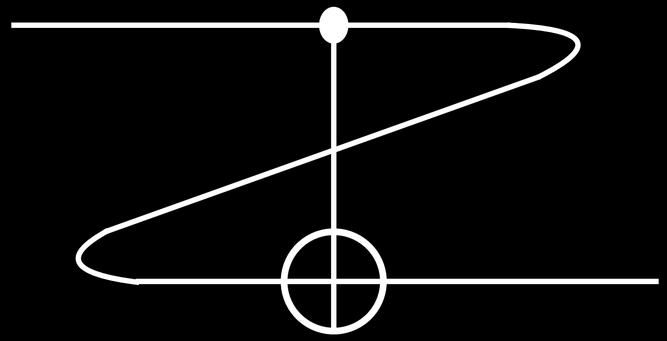
Causality Problem

Operationally wrong



Denotationally:

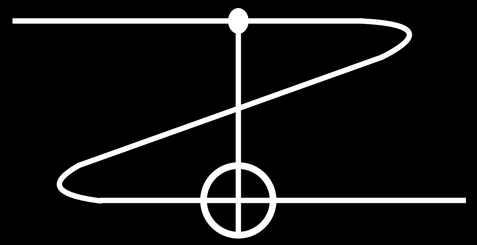
Input: $|0\rangle$



Output: $|0\rangle$

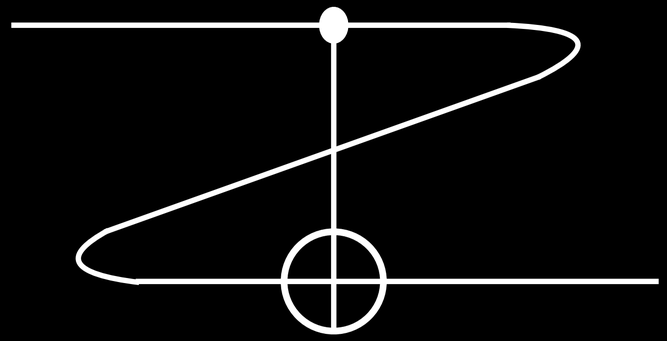
Causality Problem

Operationally wrong



Denotationally:

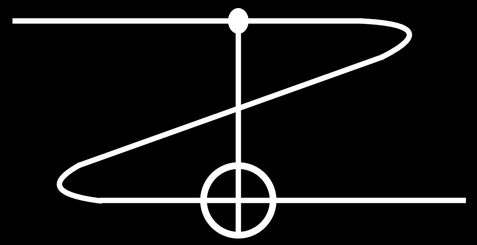
Input: $|1\rangle$



Output: $|0\rangle$

Causality Problem

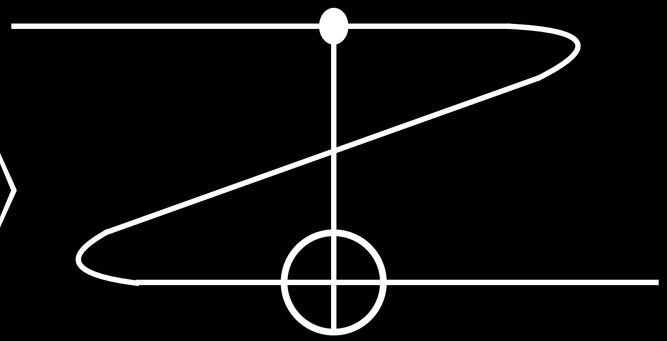
✗ Operationally wrong



✗ Denotationally **wrong**

Input: $|+\rangle$

$= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$



Output: $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|0\rangle$

$= \sqrt{2}|0\rangle$

NOT isometry ***Probability 2***

Causality Problem

Naïve type system for qif allows
a program with a causal loop

$\Longleftrightarrow$ non-isometry

$\Longleftrightarrow$ probability > 1

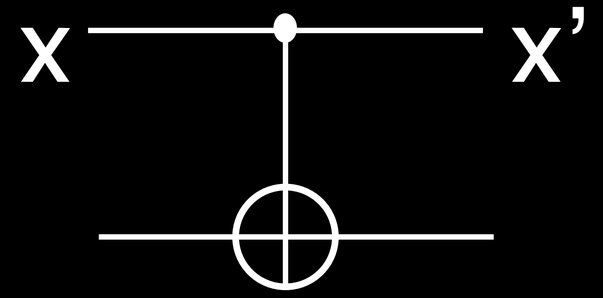
Let's fix the type system!!

New Type System

Idea: Closure and timing

let (x', f) = qif x then ***NOT*** else λy. y

Create function closure
that captures x



x' can be used only **AFTER** f is resolved

Solution: Causality relations as Types

New Type

$A, B: \mathrm{Type} \quad \Rightarrow \quad A \triangleright B : \mathrm{Type}$

$= (A \text{ after } B)$

BV-logic (extension of MLL)

Typing rule

$$\frac{\Gamma \vdash M: \mathrm{qbit} \qquad \Gamma' \vdash N: A \qquad \Gamma' \vdash L: A}{\Gamma, \Gamma' \vdash \mathrm{qif}\ M\ \mathrm{then}\ N\ \mathrm{else}\ L\ :\ \mathrm{qbit} \triangleright A}$$

Solution: Causality relations as Types

Meaning of Types

$A \otimes B$: pair (no causal dependency)

$A \triangleright B$: pair (A may depend on B)

What you can do?

- ✓ $A \otimes B \multimap A \triangleright B$ ✗ $A \triangleright B \multimap A \otimes B$
- app: $A \otimes (A \multimap B) \multimap B$ app(x, f) = f x

✗ $A \triangleright (A \multimap B) \multimap B$

Solution: Causality relations as Types

The problematic program

x : qbit ⊢ let (x’, f) : qbit ▷ (qbit ⊸ qbit) =

qif x then ***NOT*** else λy. y

in f x’ : qbit

= app(x’, f)

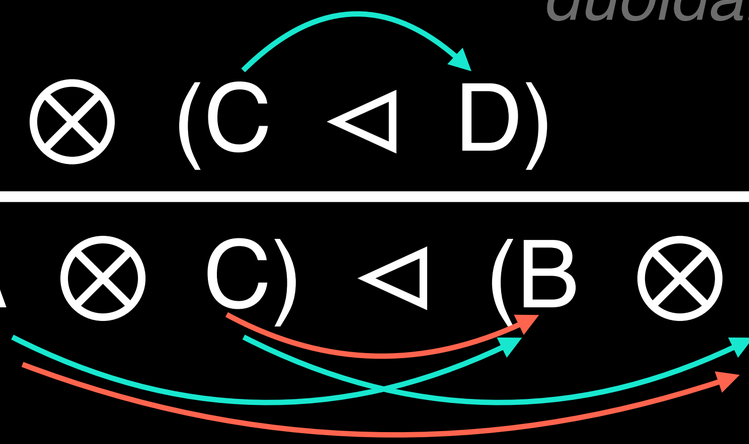
does not type check!

More structures: interchange & first order types

Structure 1. Interchange law

duoidal category

$$\frac{\Gamma \vdash M : (A \lhd B) \otimes (C \lhd D)}{\Gamma \vdash \mathsf{interch}(M) : (A \otimes C) \lhd (B \otimes D)}$$



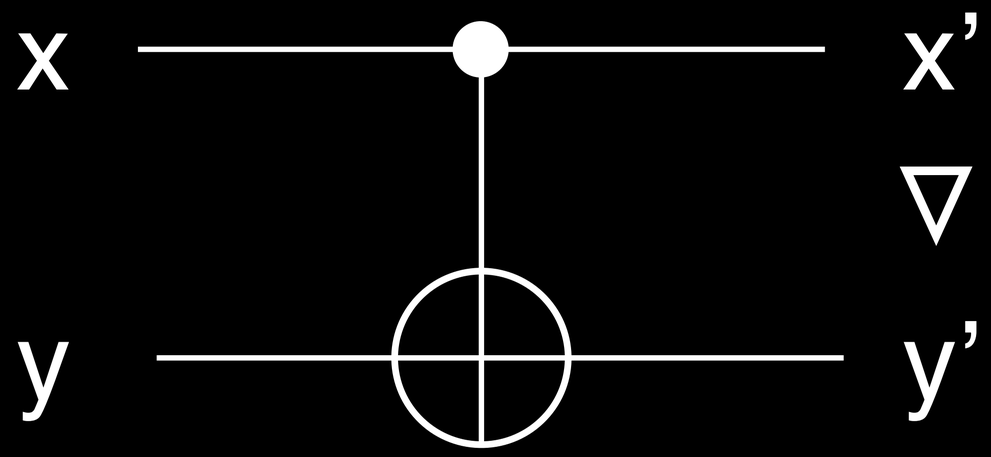
Structure 2. First order types (= w/o $\multimap$)

$$\frac{\Gamma \vdash M : A \lhd B \qquad A : \mathsf{FO}}{\Gamma \vdash \mathsf{await}(M) : A \otimes B}$$

Example: await

let (x', y') : qbit ▷ qbit

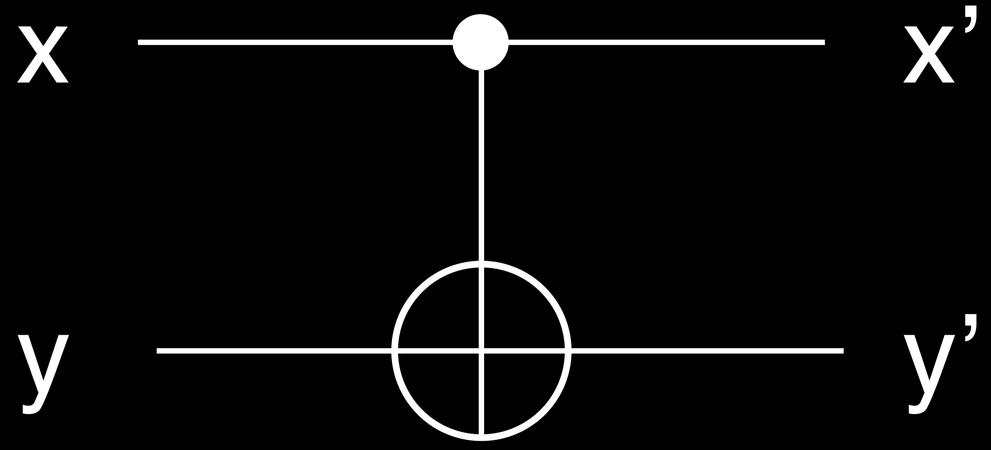
= qif x then ***NOT*** y else y



Example: await

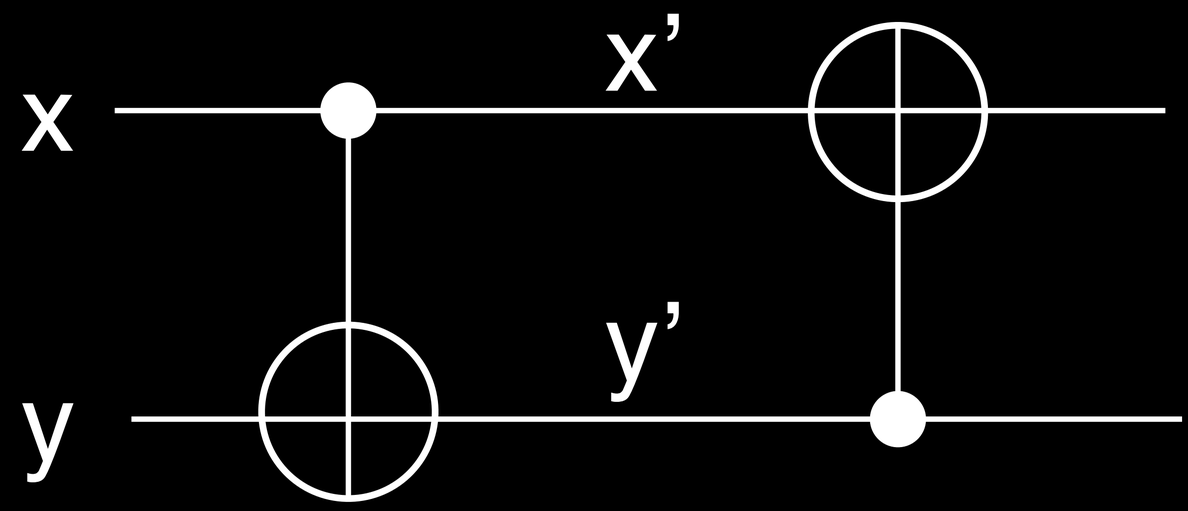
let (x', y') : qbit ⊗ qbit

= await(qif x then ***NOT*** y else y)



Example: await

```
let  (x', y')  :  qbit ⊗ qbit
             =  await(qif x then NOT y else y)
in await(qif y' then NOT x' else x')
```



(If everything is first-order, we don't need ◁)

Categorical Semantics

BV-logic and BV-categories

BV-categories interpret BV-logic

BV-cat $\mathscr{C}$: 3 monoidal structures $\otimes$, $\triangleleft$, $⅋$

- $*$-autonomous ($\otimes$, $⅋$)
- duoidal ($\otimes$, $\triangleleft$) $\Leftrightarrow$ interchange
- duoidal ($\triangleleft$, $⅋$)

BV-categories: compact closed

Prop.

A compact closed category is a BV-category, where $\otimes = \lhd = \parr$.

Example.

FHilb : Hilbert spaces + linear maps

CPM : Hilbert spaces + completely positive maps

BV-categories: Caus[-] construction

Theorem. [Simmons & Kissinger, 2022]

For each *additive precausal category* $\mathscr{C}$,

i,e., Compact closed + product

\+ discarding processes $\top_A \in \mathscr{C}(A, I)$

one can construct a BV-category Caus[$\mathscr{C}$].

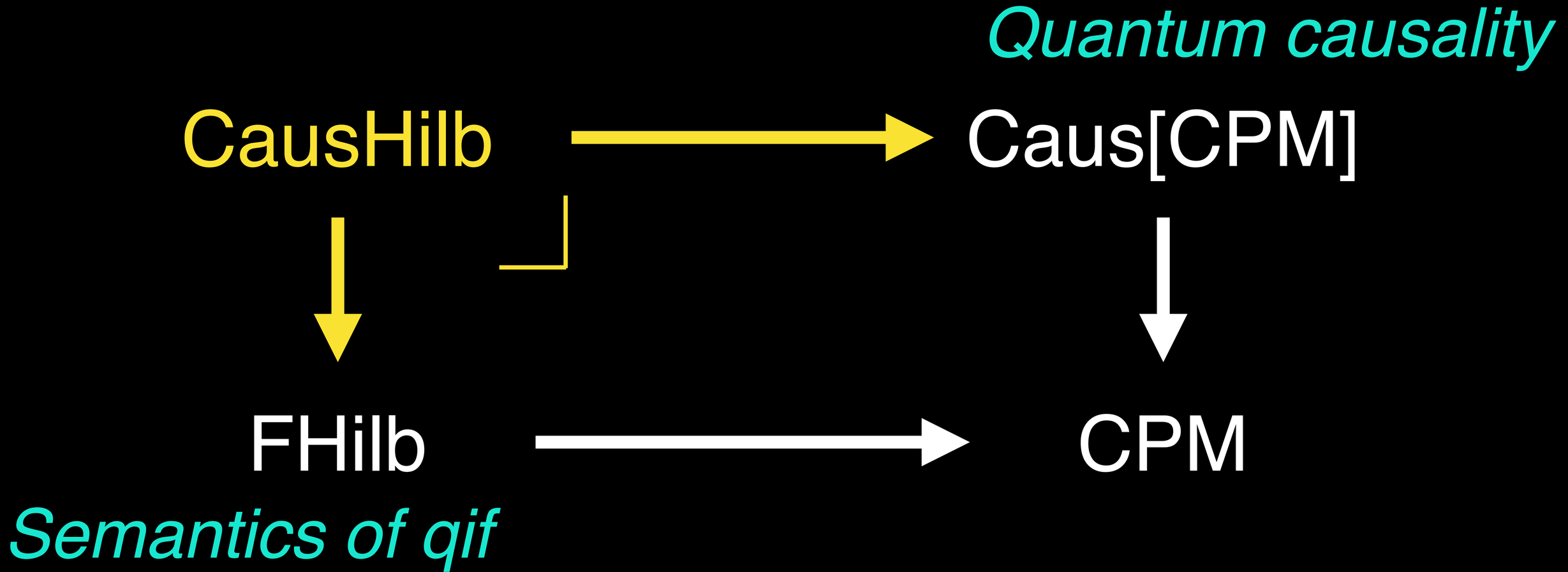
Example. Caus[CPM]

Model of our language

Quantum causality

Semantics of qif

Model of our language: Pullback!!



Main theorem

Theorem.

In our language,

$$[\![\ x : \mathrm{qbit}^{\otimes n} \vdash M : \mathrm{qbit}^{\otimes m}\ ]\!]$$

is an isometry.

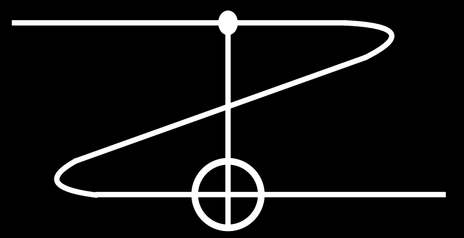
Proof.

CausHilb($[\![\mathrm{qbit}^{\otimes n}]\!]$, $[\![\mathrm{qbit}^{\otimes m}]\!]$) consists only of isometries.

Summary

We identifed the Causality Problem

$\Rightarrow$ Causal loop



↑slides & paper

Solution:

New type system from BV-logic $A \triangleright B$

Categorical model

$$\begin{array}{ccc} \textbf{\textit{CausHilb}} & \longrightarrow & \text{Caus[CPM]} \\ \downarrow & \text{p.b.} & \downarrow \\ \text{FHilb} & \longrightarrow & \text{CPM} \end{array}$$

Main theorem:

$\text{qbit}^{\otimes n} \multimap \text{qbit}^{\otimes m} \Rightarrow$ isometry $\Rightarrow$ No causal loop