↑slides

$1+1 \neq 2$, linearly

Causality in Pure Quantum Computation with Quantum Control

SPLS 10 Jun. 2026

Kengo Hirata, UoE

This paper (LICS 2026)

Causality in Pure Quantum Computation with Quantum Control

Kengo Hirata
University of Edinburgh, Edinburgh, United Kingdom
Kyoto University, Kyoto, Japan

Takeshi Tsukada
Chiba University, Chiba, Japan

Abstract

Indefinite causal order is a characteristic phenomenon in quantum computation, with examples including the quantum SWITCH and the OCB process. Not all such processes are believed to be physically realizable: while some implementations of the quantum SWITCH have been proposed, the OCB process is suspected to be unrealizable. This difference in realizability is commonly attributed to constraints imposed by physical causality.

This paper studies such a causality issue in a higher-order setting, proposing a typed lambda calculus with quantum control and its categorical semantics. Our calculus extends pure quantum computation with higher-order functions and quantum conditional branching, and it is equipped with a type system based on intuitionistic BV logic to enforce causality. We also present a novel model that is closely related to the Caus construction, by which we prove that some physically-unrealizable processes are not definable in our language.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Type theory; Theory of computation → Linear logic; Theory of computation → Categorical semantics

https://kengohirata.github.io/khirata/post/qif_lambda/

It’s about,

- Quantum Computing
- Programming Language
- Type System
- Linear Logic
- Categorical Semantics

It’s about,

- Quantum Computing
- Programming Language
- Type System
- Linear Logic
- Categorical Semantics

$1 \oplus 1 \neq 2$, linearly

Proof (?) by contradiction

Assumption 1: Number = Types

1 = Unit type

2 = Type for truth values

```
(*   x : 2       *)
(*   M, N : A    *)
if x then M else N
```

() : 1

qubit

Disclaimer: not topos

Assumption 2: Linearity

Linear type = ✗ duplication
✗ discarding

A, B : type $\Rightarrow$ $A \otimes B$: type pair

$A \multimap B$: type function

```
(*  M : A  *)
(*  N : B  *)
(M, N) : A ⊗ B
```

```
(*  x : A  *)
(*  M : B  *)
λx. M : A ⊸ B
```

Assumption 3: Plus type $\oplus$

$$A, B : \text{type} \quad \Rightarrow \quad A \oplus B : \text{type}$$

sum? type

Distributivity:

$$A \otimes (B \oplus C) \quad \cong \quad (A \otimes B) \oplus (A \otimes C)$$

Type of "if"

$$x : 2, \quad M, N : A \quad \Rightarrow \quad \text{if } x \text{ then } M \text{ else } N : 2 \otimes A$$

$$\frac{\lambda\_.M : 1 \multimap A \qquad \lambda\_.N : 1 \multimap A}{}$$

x: 2

$$2 \cong (1 \oplus 1) \xrightarrow{\lambda\_.M \oplus \lambda\_.N} A \oplus A$$

1 : unit

$$\cong (1 \otimes A) \oplus (1 \otimes A)$$

distrib.

$$\cong (1 \oplus 1) \otimes A$$

$$\cong 2 \otimes A$$

Condition stays alive

Problematic Program

Let

$$M \quad := \quad \lambda \text{x. not x} \quad : \quad 2 \multimap 2$$

$$N \quad := \quad \lambda \text{x. x} \quad : \quad 2 \multimap 2.$$

$$\text{if } x \text{ then } M \text{ else } N \quad : \quad 2 \otimes (2 \multimap 2)$$

Problematic Program

Let

$$M \quad := \quad \lambda \text{x. not x} \quad : \quad 2 \multimap 2$$

$$N \quad := \quad \lambda \text{x. x} \quad : \quad 2 \multimap 2.$$

let $(x', f) : 2 \otimes (2 \multimap 2) \quad =$

if x then M else N

Problematic Program

Let

$$M \quad := \quad \lambda \text{x. not x} \quad : \quad 2 \multimap 2$$

$$N \quad := \quad \lambda \text{x. x} \quad : \quad 2 \multimap 2.$$

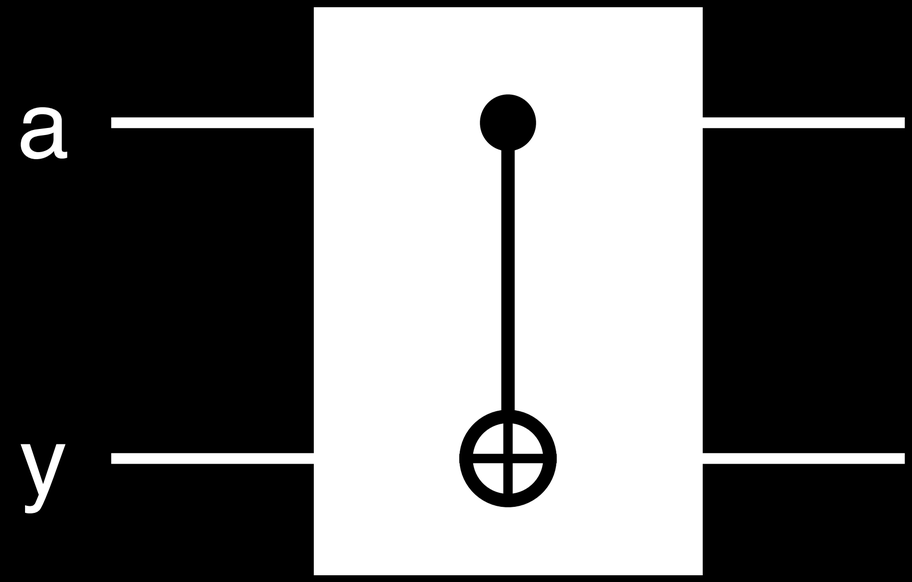
let $(x', f) : 2 \otimes (2 \multimap 2) \quad =$

if x then M else N

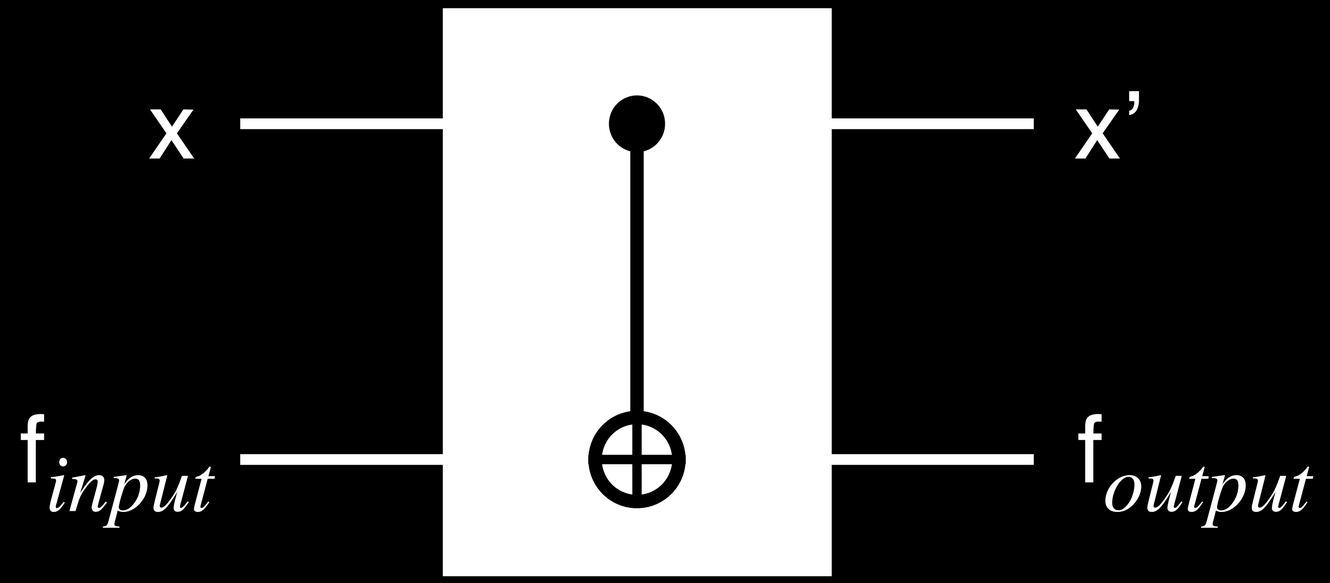
in $\quad f\, x'$

Building Block: controlled-not

let (a’, y’) = if a then (not y) else y

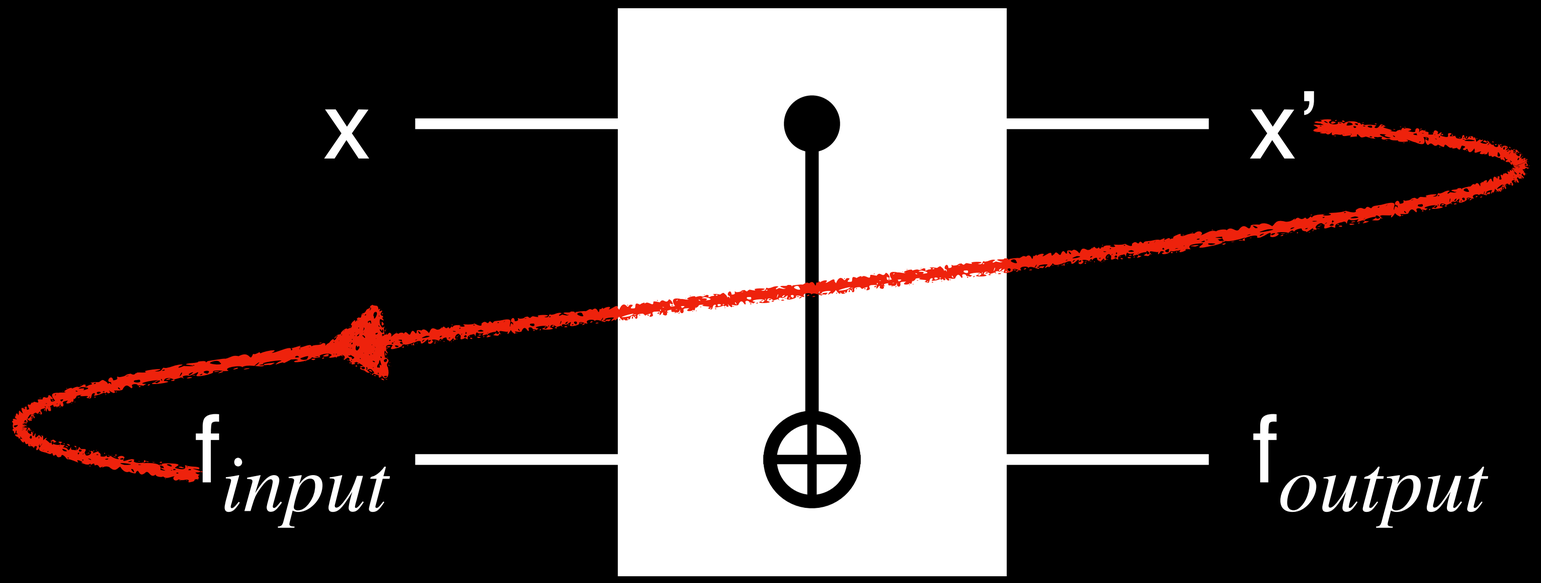


let (x’, f) = if x then (not) else (id)



let (x’, f) = if x then (not) else (id)

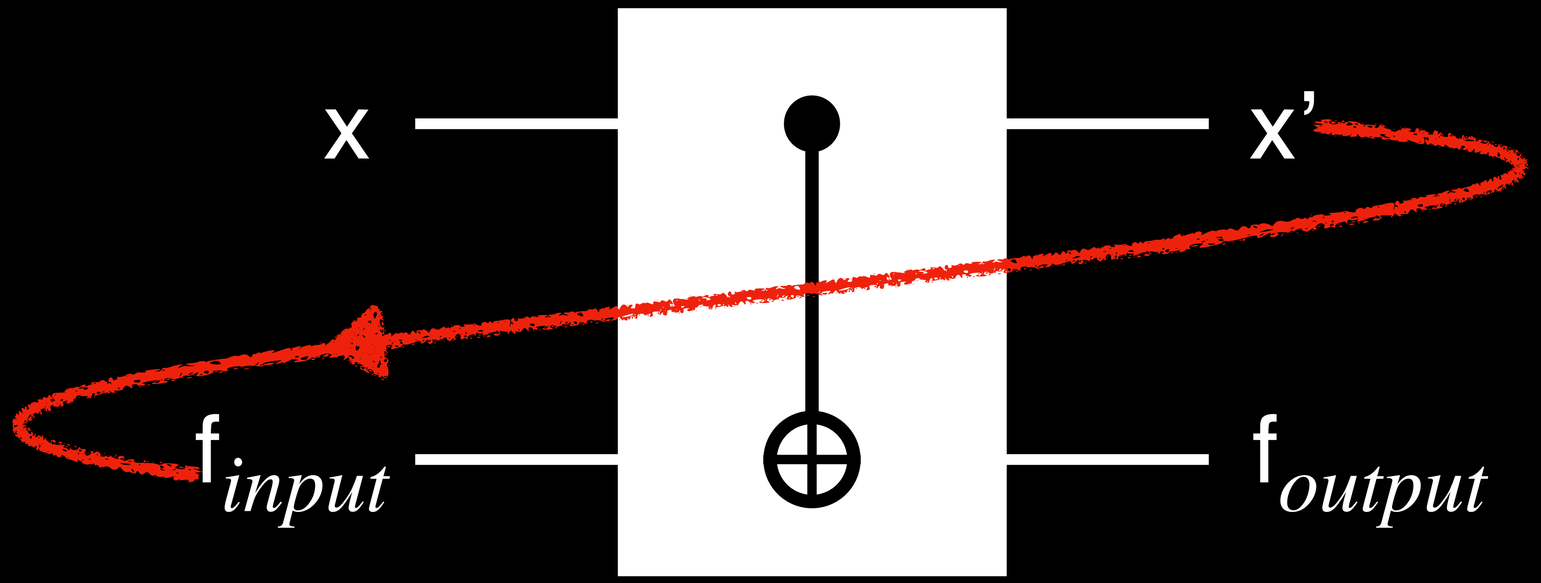
in f x’



Time

let (x’, f) = if x then (not) else (id)

in f x’

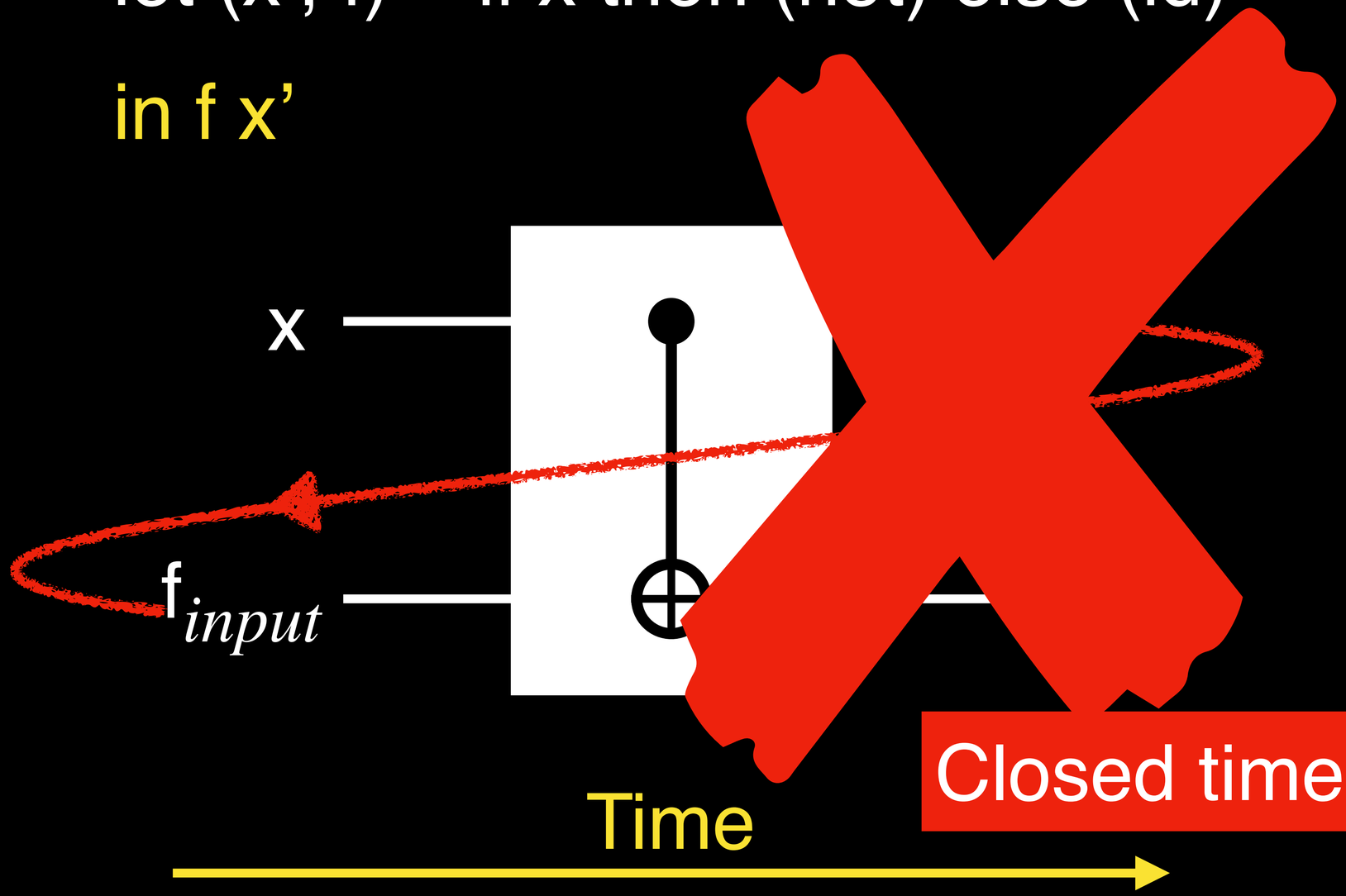


Closed timelike curve

Time

let (x’, f) = if x then (not) else (id)

in f x’



What was wrong?

- Need more assumption?

  e.g. $\oplus$ = coproduct

  $\Rightarrow$ problem remain

- My answer

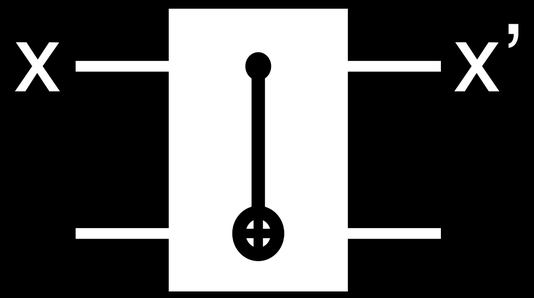
  if x then M else N : $2 \otimes A$

So, $1 \oplus 1 \neq 2$.

Idea: Closure and timing

$$\text{let } (x', f) = \quad \text{if } x \text{ then } (\text{not}) \text{ else } (\text{id})$$

Create function closure
that captures x



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

Solution: Causality relations as Types

New Type

$$A, B : \text{Type} \quad \Rightarrow \quad A \lhd B : \text{Type}$$

= (A before B)

BV-logic (extension of MLL)

Typing rule

$$\frac{\vdash x : 2 \qquad \vdash M : A \qquad \vdash N : A}{\vdash \text{if } x \text{ then } M \text{ else } N \ : \ 2 \rhd A}$$

Solution: Causality relations as Types

Meaning of Types

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

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

What you can do?

- ✓ $A \otimes B \multimap A \lhd B$ ✗ $A \lhd B \multimap A \otimes B$

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

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

Solution: Causality relations as Types

The problematic program

$$\text{let } (x', f) : 2 \triangleright (2 \multimap 2) \;=$$

$$\text{if } x \text{ then } (\text{not}) \text{ else } (\text{id})$$

$$\text{in } \boxed{f\ x'}$$

= app(x', f)

does not type check!

Summary so far

Problem: $1 \oplus 1 = 2$ raises a causality problem.

Solution: Introduce "before" type $A \lhd B$

- New typing rule

$$\vdash \text{ if } x \text{ then } M \text{ else } N \ : \ 2 \rhd A$$

- f cannot be applied to x, when (x, f): $2 \rhd (2 \multimap 2)$

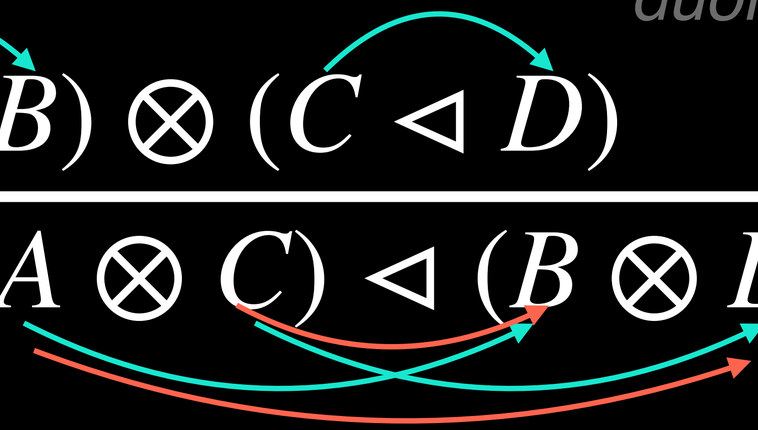
Remaining topic: More on language and types

More structures: interchange & first order types

Structure 1. Interchange law

duoidal category

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



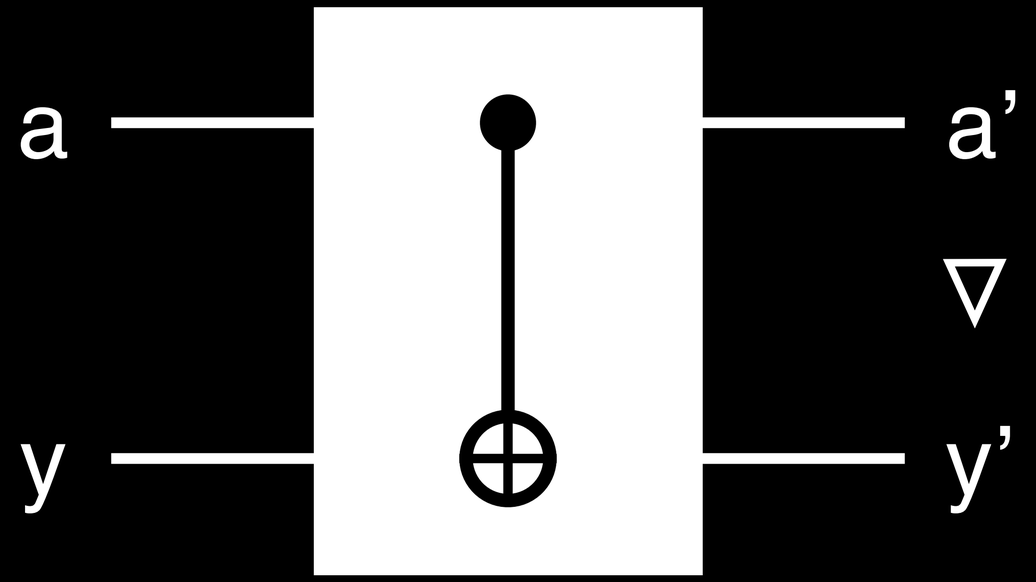
Structure 2. First order types (= w/o $\multimap$)

$$\frac{\Gamma \vdash M : A \lhd B \quad A\text{: FO}}{\Gamma \vdash \mathrm{await}(M) : A \otimes B}$$

$$\frac{A\text{: FO} \qquad \Gamma \vdash M : B \multimap (C \lhd A)}{\Gamma \vdash \mathrm{expose}(M) : (B \multimap C) \lhd A}$$

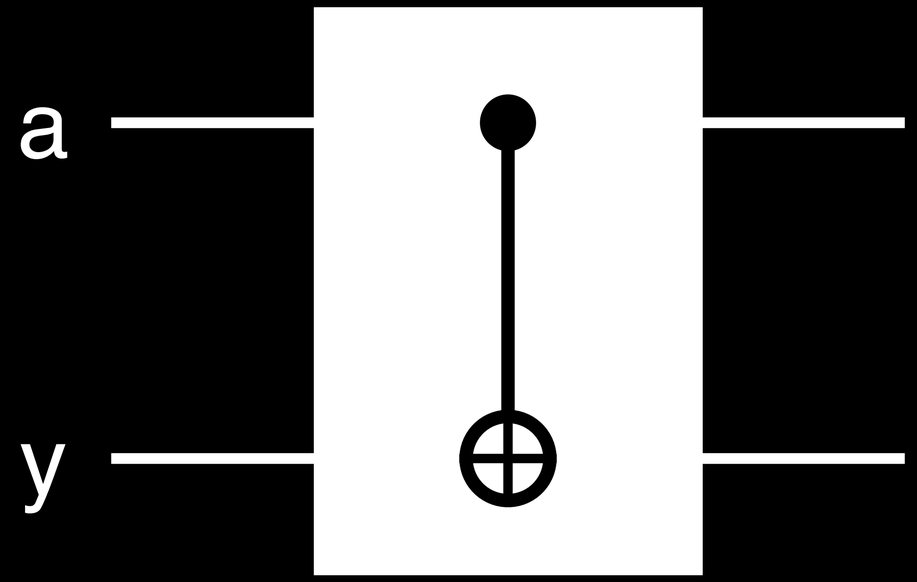
Example: await

let (a’, y’): $2 \triangleright 2$ = if a then (not y) else y



Example: await

let (a', y'): $2 \otimes 2$ = await(if a then (not y) else y)

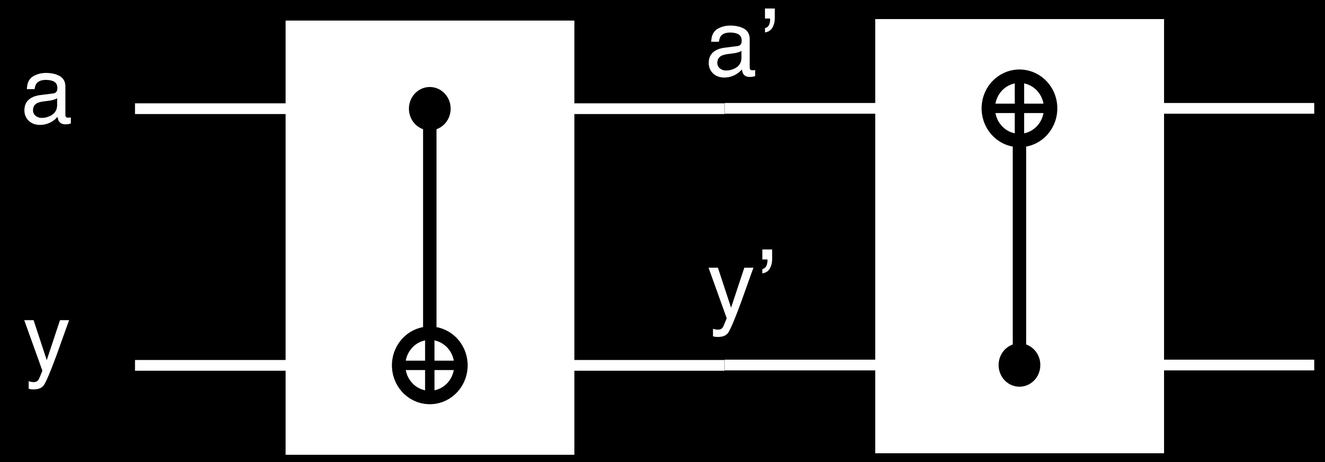


(If everything is first-order, we don't need ◁)

Example: await

let (a', y'): $2 \otimes 2$ = await(if a then (not y) else y)

in await(if y' then (not a') else a')



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

Summary ~~so far~~

Problem: $1 \oplus 1 = 2$ raises a causality problem.

Solution: Introduce "before" type $A \lhd B$

- New typing rule

$$\vdash \text{ if } x \text{ then } M \text{ else } N \ : \ 2 \rhd A$$

- f cannot be applied to x, when (x, f): $2 \rhd (2 \multimap 2)$

Remaining topic: More on language and types

What is $1 \oplus 1$ then?

2 = qubit

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

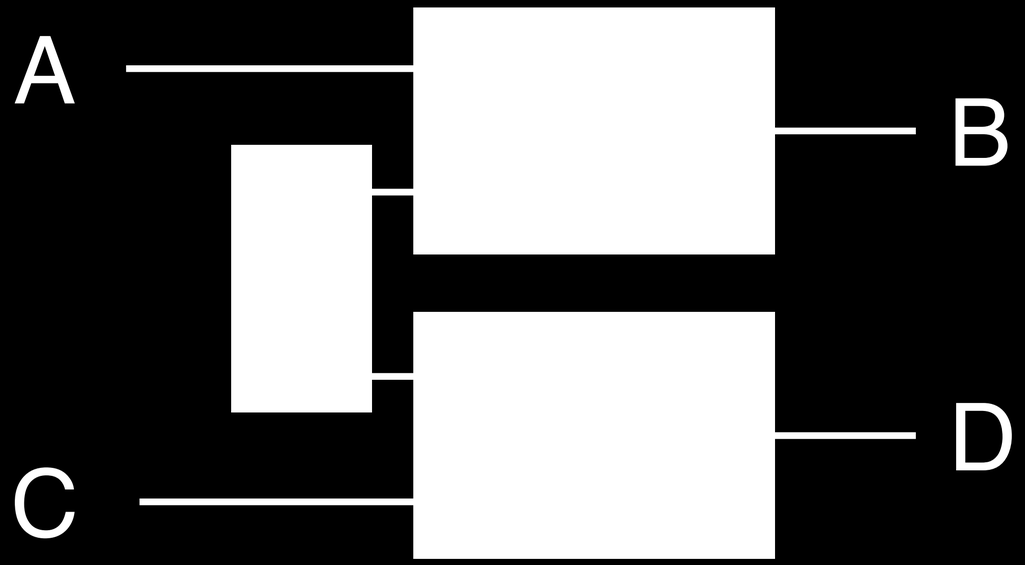
$1 \oplus 1$

$$e^{i\theta}|0\rangle \text{ or } e^{i\theta}|1\rangle$$

categorically.

Communication and causality

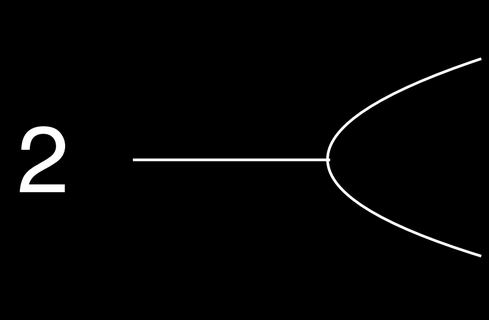
$$(A \multimap B) \otimes (C \multimap D)$$



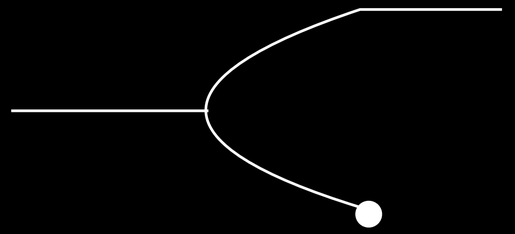
$$(A \multimap B) \triangleleft (C \multimap D)$$

Coproduct does not solve the problem

$$2 = (1 \oplus 1) \xrightarrow{\text{inl} \oplus \text{inr}} (1 \oplus 1) \oplus (1 \oplus 1) = 4$$

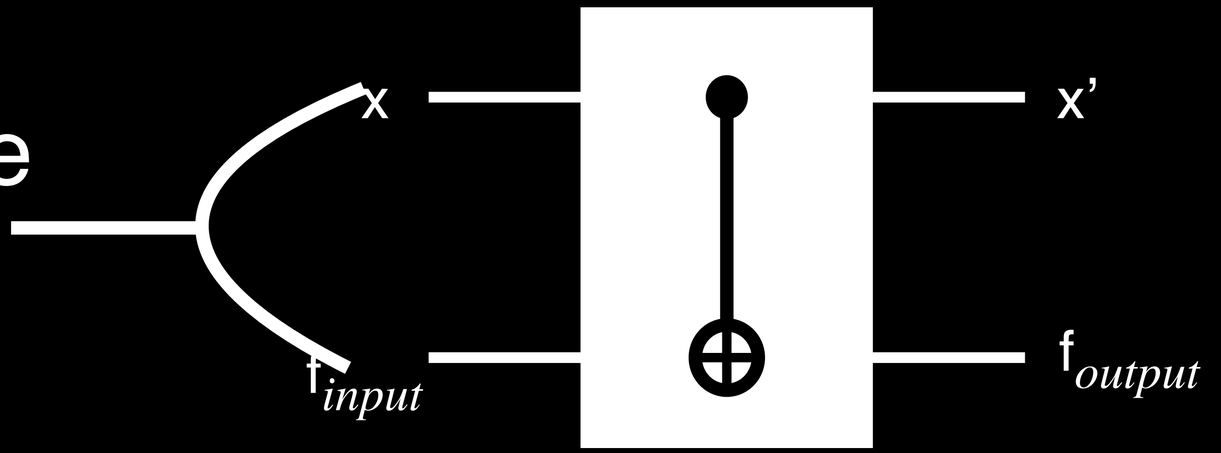


but



$\neq$

We can do like



but it doesn't provide a compositional semantics

Exercise

Make a term M (B: FO)

$$\text{x: } (A \multimap B) \lhd (B \multimap C) \vdash M : A \multimap C$$

Exercise (B: FO)

Answer: Let $\Gamma =$ x: $(A \multimap B) \triangleleft (B \multimap C)$, a : A

$$\Gamma \vdash x \otimes (a \triangleleft ()) \quad : ((A \multimap B) \triangleleft (B \multimap C)) \otimes (A \triangleleft 1)$$

$$\Gamma \vdash \text{interch}(x \otimes (a \triangleleft ())) \quad : ((A \multimap B) \otimes A) \triangleleft (1 \otimes (B \multimap C))$$

$$\Gamma \vdash \text{let } y \triangleleft z = \text{interch}(x \otimes (a \triangleleft ())) \text{ in}$$

$$\text{app}(y) \triangleleft (\text{let\_} \otimes \text{g} = w \text{ in } g) \quad : B \triangleleft (B \multimap C)$$

$$\Gamma \vdash \text{app}(\text{await}(\text{let } y \triangleleft z = \text{interch}(x \otimes (a \triangleleft ())) \text{ in}$$

$$\text{app}(y) \triangleleft (\text{let\_} \otimes \text{g} = w \text{ in } g))) \quad : C$$