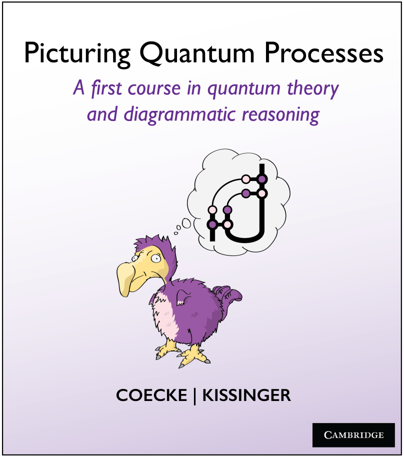
Quantum Picturalism

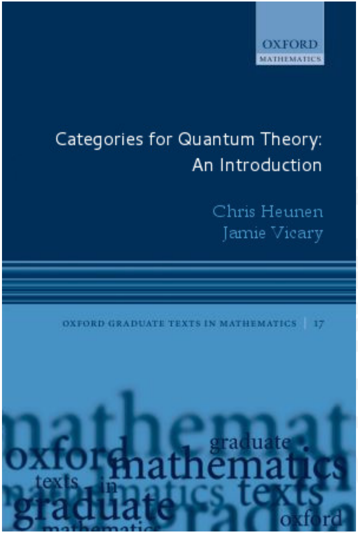
Bob Coecke[1], Chris Heunen[2], and **Aleks Kissinger**[3]

[1]University of Oxford
[2]University of Edinburgh
[3]**Radboud University Nijmegen**

Foundations 2016, LSE



CUP 2016

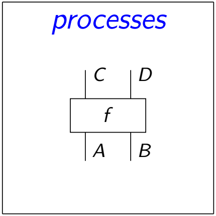


OUP 2016

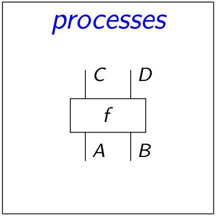
Quantum Picturalism

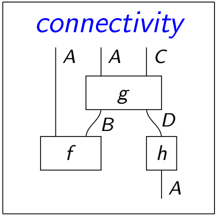
The idea: Describe quantum theory entirely in terms of:

The idea: Describe quantum theory entirely in terms of:



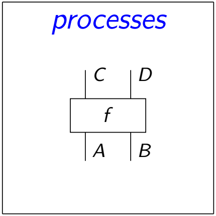
The idea: Describe quantum theory entirely in terms of:

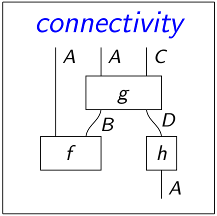


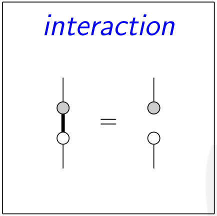


Quantum Picturalism

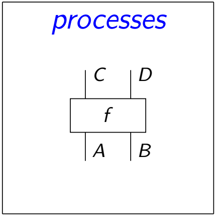
The idea: Describe quantum theory entirely in terms of:

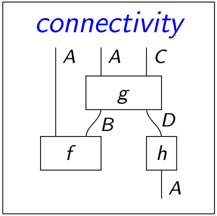


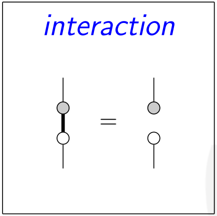




The idea: Describe quantum theory entirely in terms of:







Not in terms of:

- Hilbert space
- self-adjoint operators, unitary transformations
- calculations with matrices/complex numbers
- ....

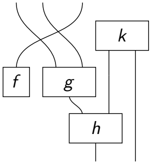
(though some may be emergent notions)

Why?

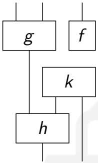
- Simpler!

$$(1 \otimes \sigma \otimes k) \circ (\sigma \otimes 1 \otimes 1 \otimes 1) \circ (f \otimes g \otimes 1 \otimes 1) \circ (h \otimes 1) = (g \otimes f) \circ (1 \otimes k) \circ (h \otimes 1)$$

vs.



$=$

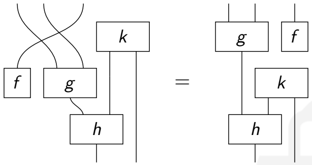


Why?

- Simpler!

$$(1 \otimes \sigma \otimes k) \circ (\sigma \otimes 1 \otimes 1 \otimes 1) \circ (f \otimes g \otimes 1 \otimes 1) \circ (h \otimes 1) = (g \otimes f) \circ (1 \otimes k) \circ (h \otimes 1)$$

vs.



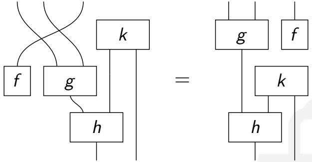
- New perspective = new insights

Why?

- Simpler!

$$(1 \otimes \sigma \otimes k) \circ (\sigma \otimes 1 \otimes 1 \otimes 1) \circ (f \otimes g \otimes 1 \otimes 1) \circ (h \otimes 1) = (g \otimes f) \circ (1 \otimes k) \circ (h \otimes 1)$$

vs.



- New perspective = new insights
- Reconstruction ⇐ 'diagrammatic backbone' + extra assms

e.g. **Pavia 2010** and **Hardy 2011**

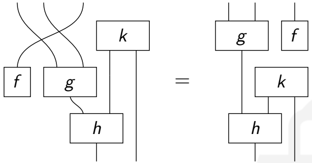
Hardy (2010): "we join the quantum picturalism revolution"

Why?

- Simpler!

$$(1 \otimes \sigma \otimes k) \circ (\sigma \otimes 1 \otimes 1 \otimes 1) \circ (f \otimes g \otimes 1 \otimes 1) \circ (h \otimes 1) = (g \otimes f) \circ (1 \otimes k) \circ (h \otimes 1)$$

vs.



- New perspective = new insights
- Reconstruction ⇐ 'diagrammatic backbone' + extra assms

e.g. **Pavia 2010** and **Hardy 2011**

Hardy (2010): "we join the quantum picturalism revolution"

- A 'theory playground'

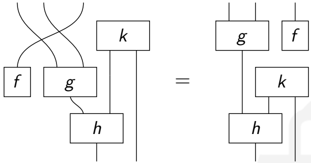
e.g. QT vs. real/boolean-valued/modal QT, stabiliser QT vs. Spekken's toy theory, OPTs, ...

Why?

- Simpler!

$$(1 \otimes \sigma \otimes k) \circ (\sigma \otimes 1 \otimes 1 \otimes 1) \circ (f \otimes g \otimes 1 \otimes 1) \circ (h \otimes 1) = (g \otimes f) \circ (1 \otimes k) \circ (h \otimes 1)$$

vs.



- New perspective = new insights
- Reconstruction ⇐ 'diagrammatic backbone' + extra assms

  e.g. **Pavia 2010** and **Hardy 2011**

  Hardy (2010): "we join the quantum picturalism revolution"

- A 'theory playground'

  e.g. QT vs. real/boolean-valued/modal QT, stabiliser QT vs. Spekken's toy theory, OPTs, ...

- New calculational tools, applications in quantum info/computation

Processes

- A process is anything with zero or more *inputs* and zero or more *outputs*

Processes

- A process is anything with zero or more *inputs* and zero or more *outputs*
- For example, this function:

$$f(x, y) = x^2 + y$$

Processes

- A process is anything with zero or more *inputs* and zero or more *outputs*
- For example, this function:

$$f(x, y) = x^2 + y$$

...is a process when takes two real numbers as input, and produces a real number as output.

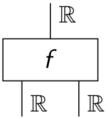
Processes

- A process is anything with zero or more *inputs* and zero or more *outputs*
- For example, this function:

$$f(x, y) = x^2 + y$$

...is a process when takes two real numbers as input, and produces a real number as output.

- We could also write it like this:



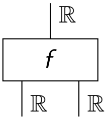
Processes

- A process is anything with zero or more *inputs* and zero or more *outputs*
- For example, this function:

$$f(x, y) = x^2 + y$$

...is a process when takes two real numbers as input, and produces a real number as output.

- We could also write it like this:



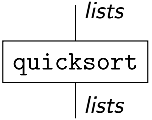
- The labels on wires are called system-types or just types

More processes

- Similarly, computer programs are processes

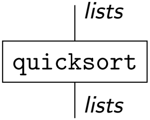
More processes

- Similarly, computer programs are processes
- For example, a program that sorts lists might look like this:

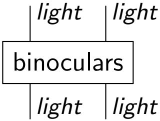


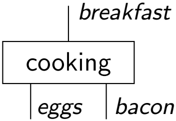
More processes

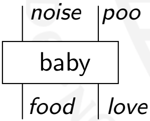
- Similarly, computer programs are processes
- For example, a program that sorts lists might look like this:



- These are also perfectly good processes:

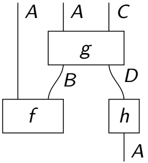






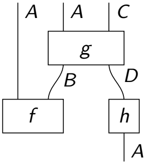
Diagrams

- We can combine simple processes to make more complicted ones, described by diagrams:

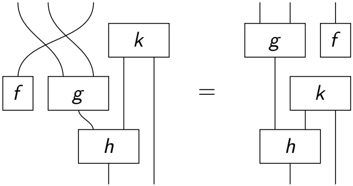


Diagrams

- We can combine simple processes to make more complicted ones, described by diagrams:

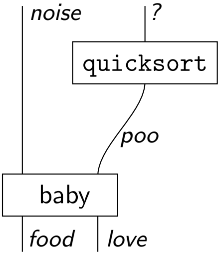


- The golden rule: only connectivity matters!

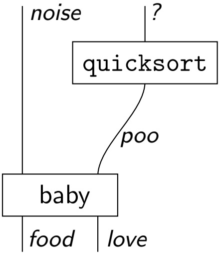


Types and Process Theories

- Connections are only allowed where the types match
- Ill-typed diagrams are undefined:



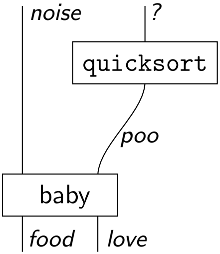
- Connections are only allowed where the types match
- Ill-typed diagrams are undefined:



- In fact, these processes don't ever make sense to plug together

Types and Process Theories

- Connections are only allowed where the types match
- Ill-typed diagrams are undefined:



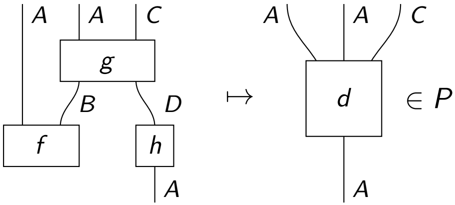
- In fact, these processes don't ever make sense to plug together
- A family of processes which do make sense together is called a process theory

A *process theory* consists of:

- a set T of *system-types*,
- a set P of *processes*

which are:

- closed under forming diagrams:



- Processes with no inputs are called states:

Special processes: states and effects

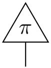
- Processes with no inputs are called states:

Interpret as: preparing a system in a particular configuration, where we don't care what came before.

- Processes with no inputs are called states:

Interpret as: preparing a system in a particular configuration, where we don't care what came before.

- Processes with no outputs are called effects:

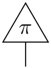


Special processes: states and effects

- Processes with no inputs are called states:

Interpret as: preparing a system in a particular configuration, where we don't care what came before.

- Processes with no outputs are called effects:



Interpret as: testing for a property π, where we don't care what happens after.

Numbers

- A number is a process with no inputs or outputs, written as:

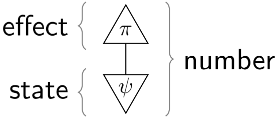
$$\diamond\lambda \qquad \text{or just:} \qquad \lambda$$

Numbers

- A number is a process with no inputs or outputs, written as:

or just: λ

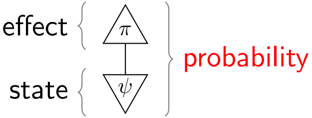
Interpret as: what happens when a state meets an effect



- A number is a process with no inputs or outputs, written as:

or just: λ

Interpret as: what happens when a state meets an effect

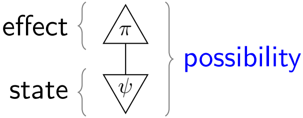


Numbers

- A number is a process with no inputs or outputs, written as:

or just: λ

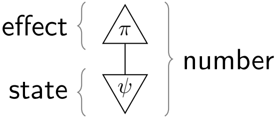
Interpret as: what happens when a state meets an effect



- A number is a process with no inputs or outputs, written as:

or just: λ

Interpret as: what happens when a state meets an effect



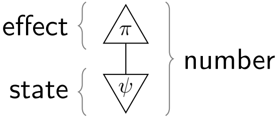
This is called the (generalised) Born rule

Numbers

- A number is a process with no inputs or outputs, written as:

or just: λ

Interpret as: what happens when a state meets an effect



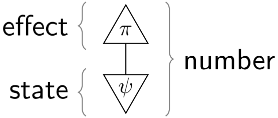
This is called the (generalised) Born rule

- From properties of diagrams, we get:

- A number is a process with no inputs or outputs, written as:

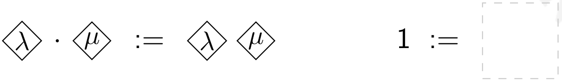
or just: λ

Interpret as: what happens when a state meets an effect



This is called the (generalised) Born rule

- From properties of diagrams, we get:



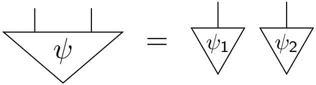
Q: What kinds of behaviour can we study using just diagrams, and nothing else?

Q: What kinds of behaviour can we study using just diagrams, and nothing else?

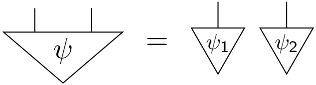
A: (Non-)separability

Separability for states

- Separable:



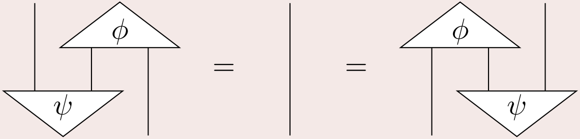
- Separable:



- vs. 'completely non-separable':

Definition

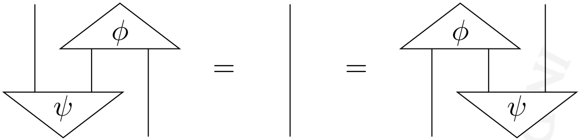
A state ψ is called *cup-state* if there exists an effect ϕ, called a *cap-effect*, such that:



- By introducing some clever notation:

- By introducing some clever notation:

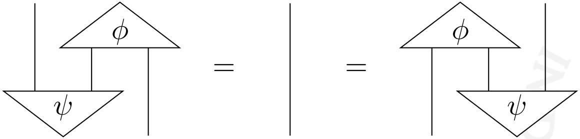
- Then these equations:



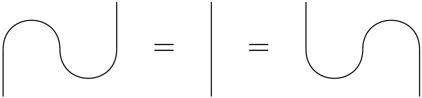
Cup-states

- By introducing some clever notation:

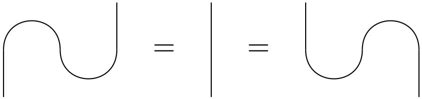
- Then these equations:



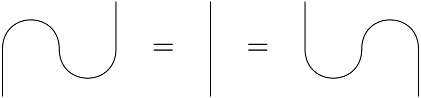
- ...look like this:



Yank the wire!



Yank the wire!



A no-go theorem for separability

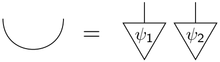
Theorem

If a process theory (i) has cup-states for every type and (ii) every state separates, then it has trivial dynamics.

Theorem

If a process theory (i) has cup-states for every type and (ii) every state separates, then it has trivial dynamics.

Proof. Suppose a cup-state separates:

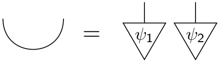


Then for any f:

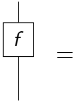
Theorem

If a process theory (i) has cup-states for every type and (ii) every state separates, then it has trivial dynamics.

Proof. Suppose a cup-state separates:



Then for any f:

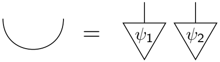


□

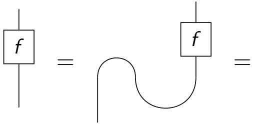
Theorem

If a process theory (i) has cup-states for every type and (ii) every state separates, then it has trivial dynamics.

Proof. Suppose a cup-state separates:



Then for any f:

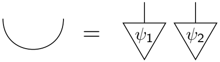


A no-go theorem for separability

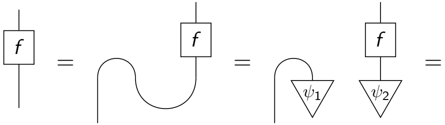
Theorem

If a process theory (i) has cup-states for every type and (ii) every state separates, then it has trivial dynamics.

Proof. Suppose a cup-state separates:



Then for any f:



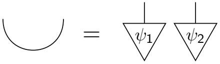
□

A no-go theorem for separability

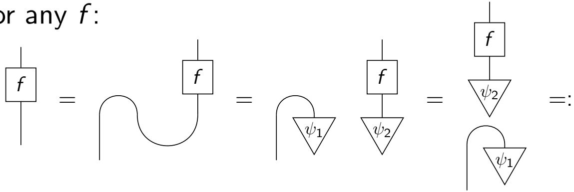
Theorem

If a process theory (i) has cup-states for every type and (ii) every state separates, then it has trivial dynamics.

Proof. Suppose a cup-state separates:



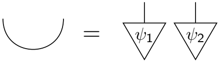
Then for any f:



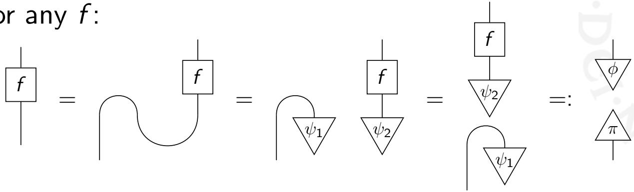
Theorem

If a process theory (i) has cup-states for every type and (ii) every state separates, then it has trivial dynamics.

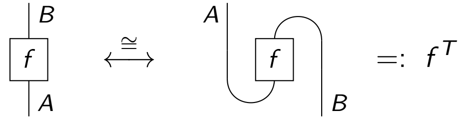
Proof. Suppose a cup-state separates:

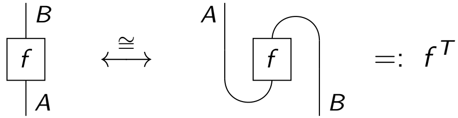


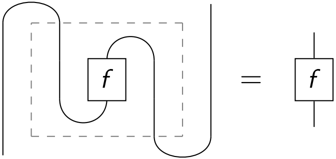
Then for any f:



Transpose





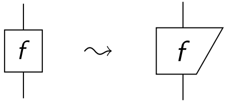


i.e.

$$(f^T)^T = f$$

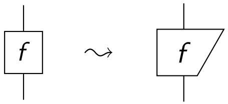
Tranpose = rotation

A bit of a deformation:

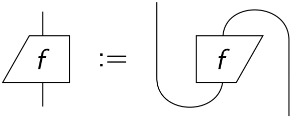


Tranpose = rotation

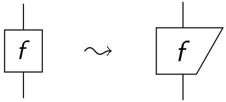
A bit of a deformation:



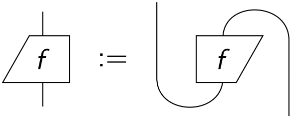
allows some clever notation:

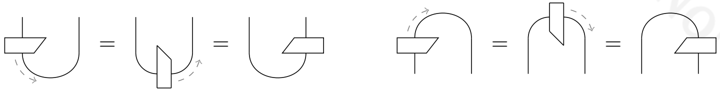


A bit of a deformation:

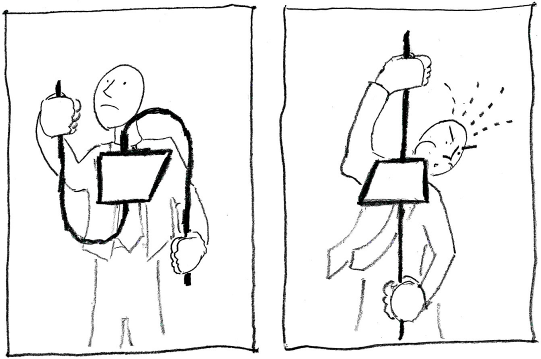


allows some clever notation:



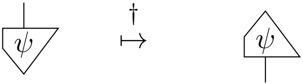


Transpose = rotation



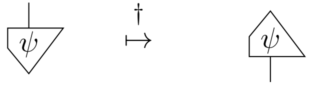
Adjoint = reflection

Adjoint = reflection



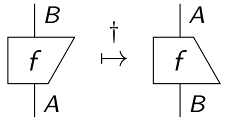
state ψ **testing for** ψ

Adjoint = reflection

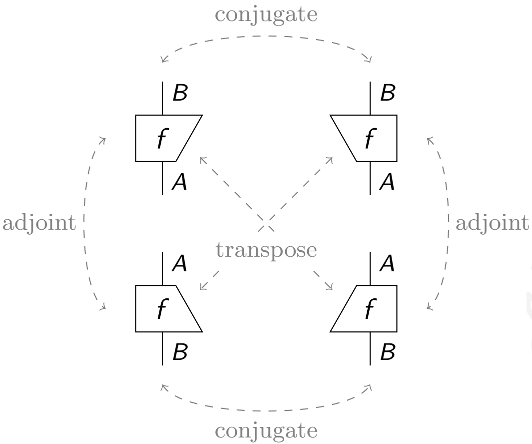


state ψ **testing for** ψ

Extends from states/effects to all processes:



4 kinds of box

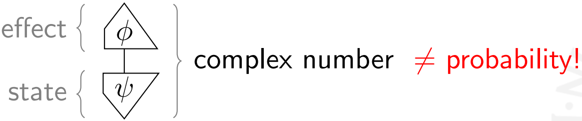


Doubling

Doubling

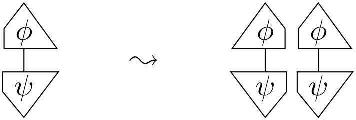
If the ‘numbers’ of our process theory are complex numbers (e.g. as in **linear maps**), then we have a problem:

If the 'numbers' of our process theory are complex numbers (e.g. as in **linear maps**), then we have a problem:

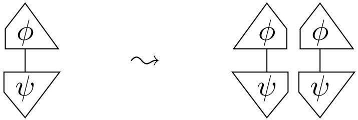


Doubling

Solution: multiply by the conjugate:

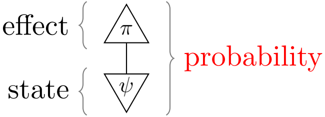


Solution: multiply by the conjugate:



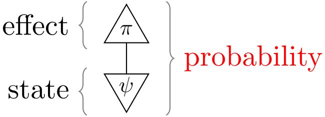
(i.e. use the 'plain old' Born rule: $\overline{\langle\phi|\psi\rangle}\langle\phi|\psi\rangle = |\langle\phi|\psi\rangle|^2$)

New problem: We lost this:



Doubling

New problem: We lost this:

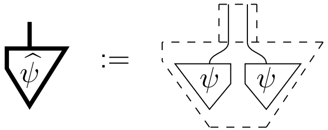


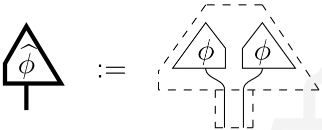
...which was the basis of our interpretation for states, effects, and numbers.

Doubling

Solution: Make a new process theory with doubling 'baked in':

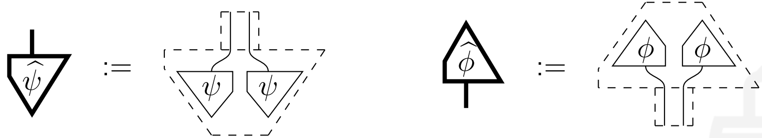
Solution: Make a new process theory with doubling 'baked in':



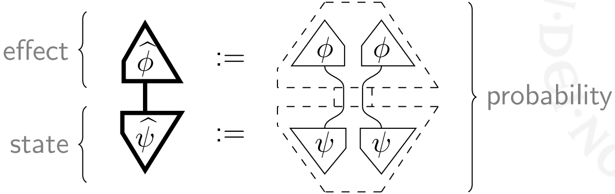


Doubling

Solution: Make a new process theory with doubling 'baked in':

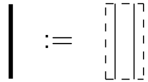


Then:



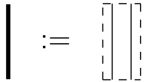
Doubling

The new process theory has doubled systems $\widehat{H} := H \otimes H$:

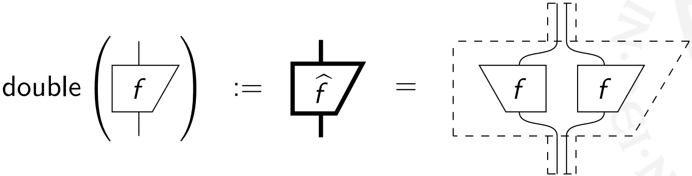


Doubling

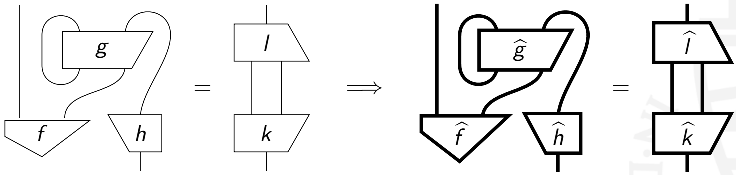
The new process theory has doubled systems $\widehat{H} := H \otimes H$:

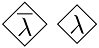


and processes:



Doubling preserves diagrams

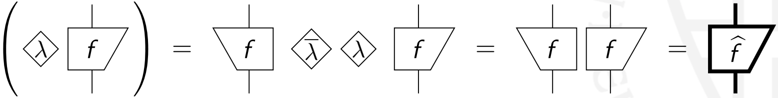




$= 1$ (i.e. $\lambda = e^{i\alpha}$)

$\Longrightarrow$

double



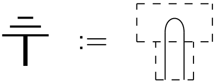
Discarding

Doubling also lets us do something we couldn't do before:

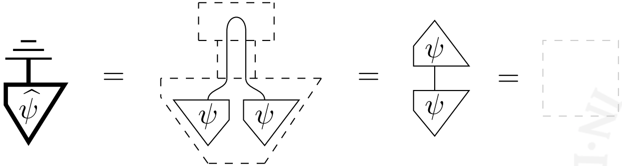
Doubling also lets us do something we couldn't do before: throw stuff away!

Doubling also lets us do something we couldn't do before: throw stuff away!

How? Like this:

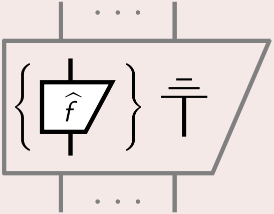


For normalised ψ, the two copies annihilate:



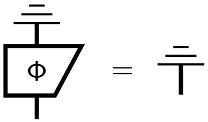
Definition

The process theory of **quantum maps** has as types (doubled) Hilbert spaces $\widehat{H}$ and as processes:



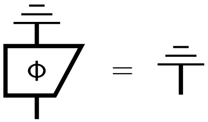
Causality

A quantum map is called *causal* if:



Causality

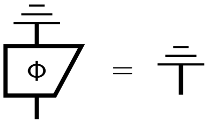
A quantum map is called *causal* if:



If we discard the output of a process,
it doesn't matter which process happened.

Causality

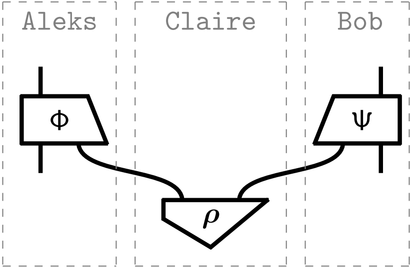
A quantum map is called *causal* if:



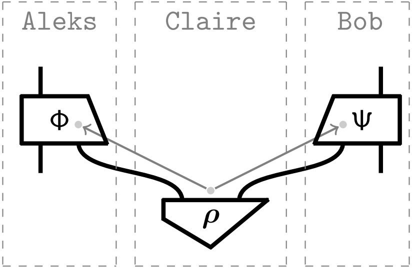
If we discard the output of a process,
it doesn't matter which process happened.

causal $\iff$ *deterministically physically realisable*

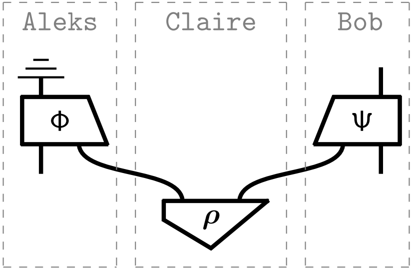
Consequence: no signalling



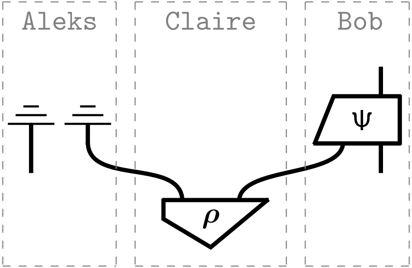
Consequence: no signalling



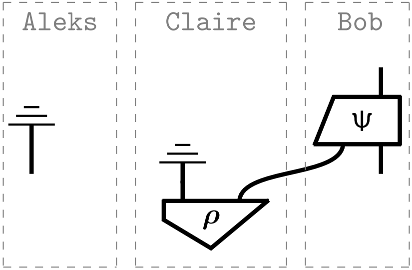
Consequence: no signalling



Consequence: no signalling

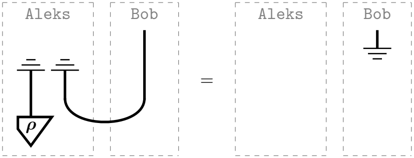


Consequence: no signalling

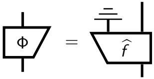


Consequences of doubling + causality

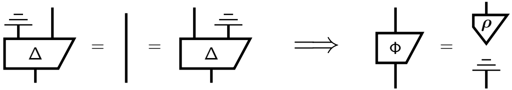
- Impossibility of deterministic teleporation:



- Purification/Stinespring dilation



- Quantum no-broadcasting theorem



Classical and quantum interaction

Classical and quantum interaction

$$\left(\text{quantum} \quad := \quad \Big| \right)$$

Classical and quantum interaction

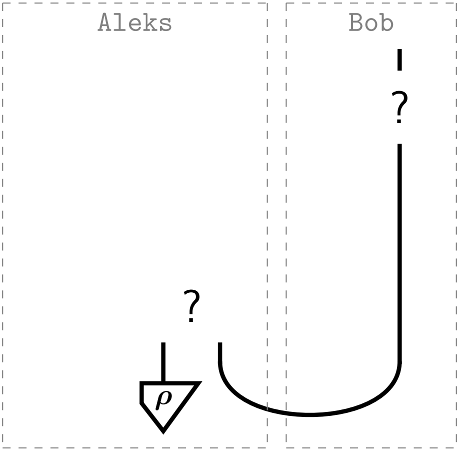
$$\left(\text{quantum} \;\; := \;\; \rule{3pt}{2em} \right) \;\neq\; \left(\text{classical} \;\; := \;\; \Big| \right)$$

Classical and quantum interaction

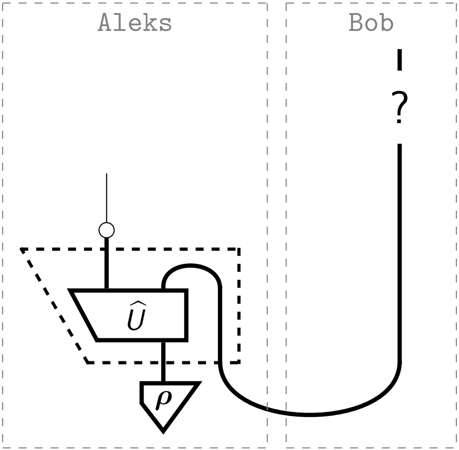
encode $:=$ 

measure $:=$ 

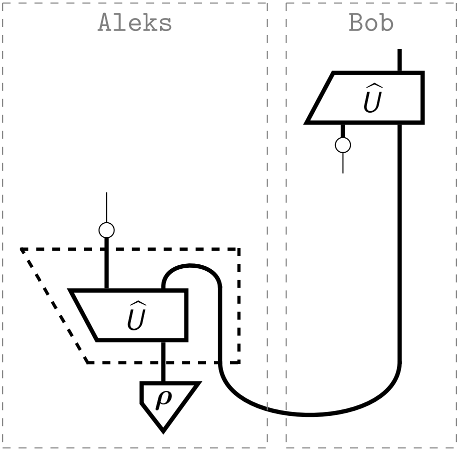
Quantum teleportation: take 2

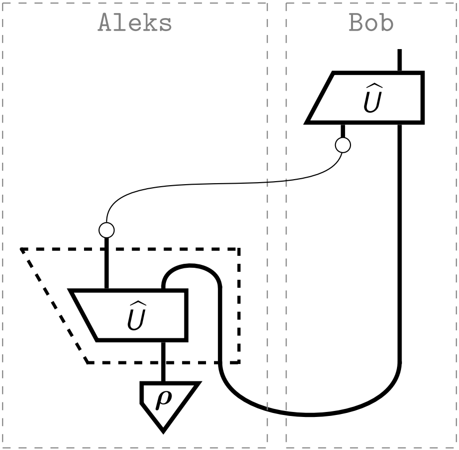


Quantum teleportation: take 2

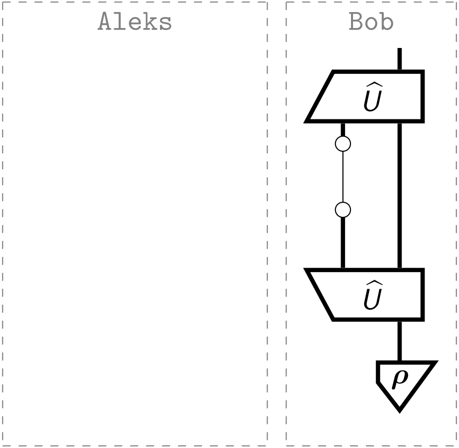


Quantum teleportation: take 2

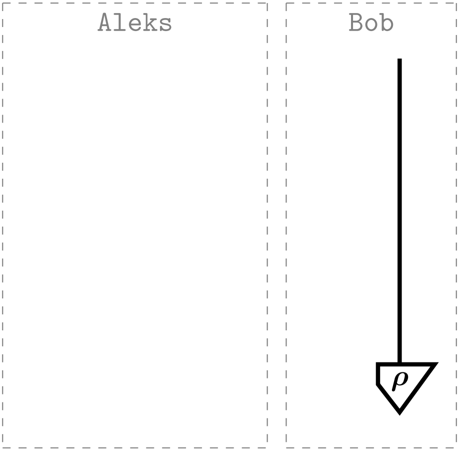



Aleks
Bob
$\widehat{U}$
$\widehat{U}$
ρ

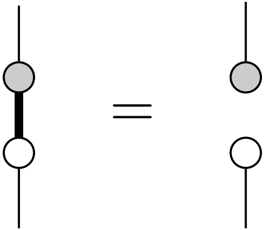
Quantum teleportation: take 2

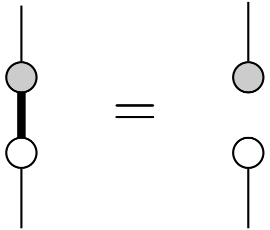


Quantum teleportation: take 2



Complementarity

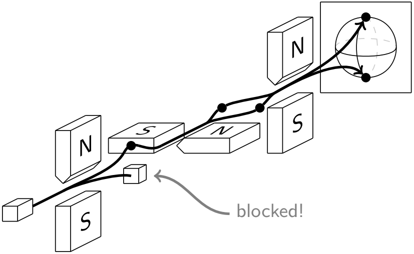


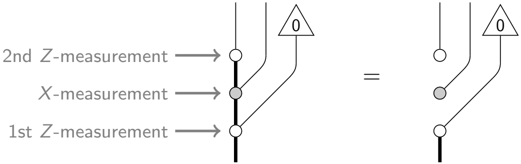


Interpretation:

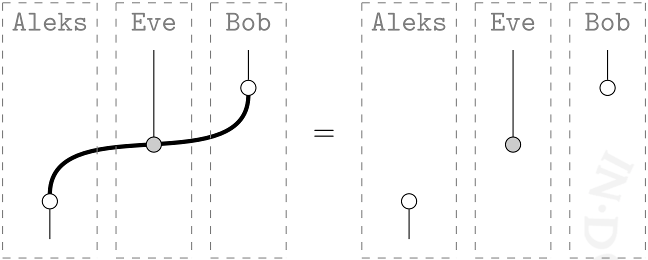
(encode in ○) THEN (measure in ◉) = (no data flow)

e.g. Stern-Gerlach



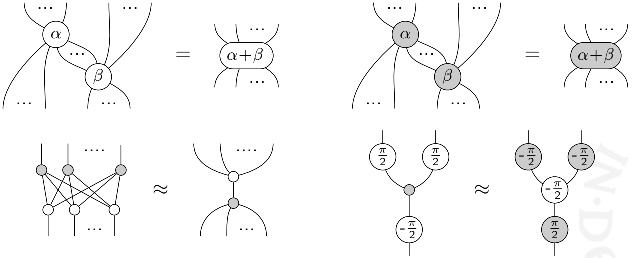


e.g. Quantum Key Distribution



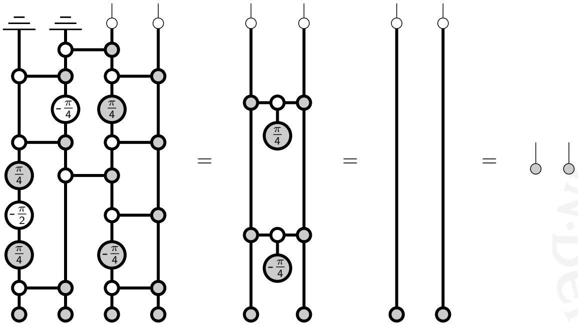
Graphical calculus

Complementarity + group structure = **ZX-calculus**:

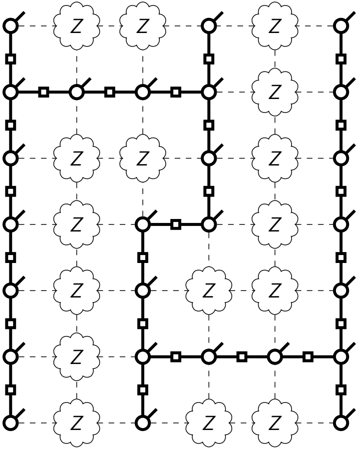


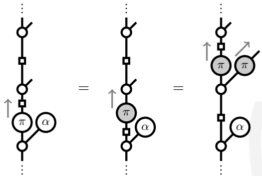
A **sound and complete** equational theory for stabilizer quantum mechanics.

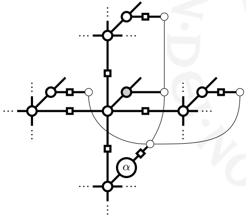
Quantum circuit simplification



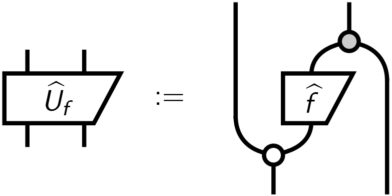
Measurement-based quantum computation





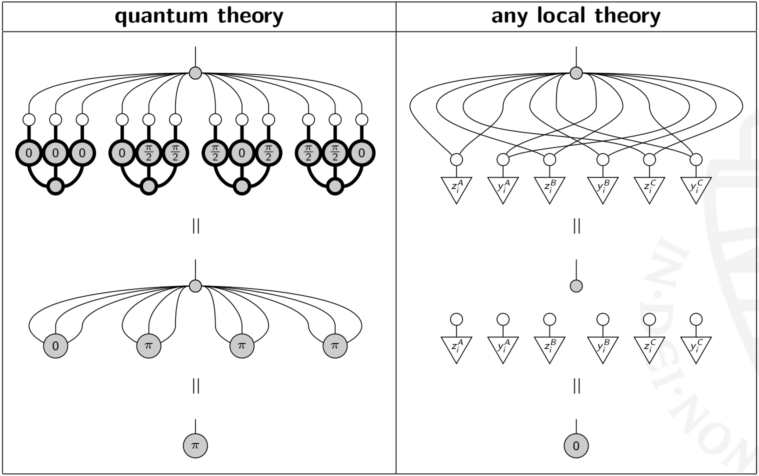


Quantum algorithms

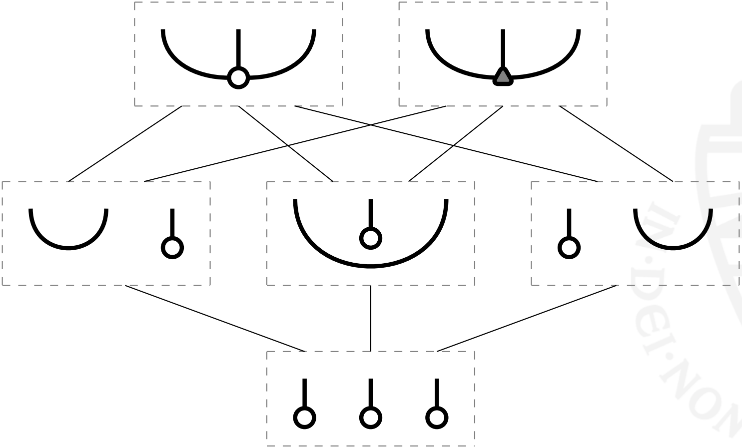


⇒ simple derivations of **Deutsch-Jozsa**, **quantum seach**, and **hidden subgroup** algorithms.

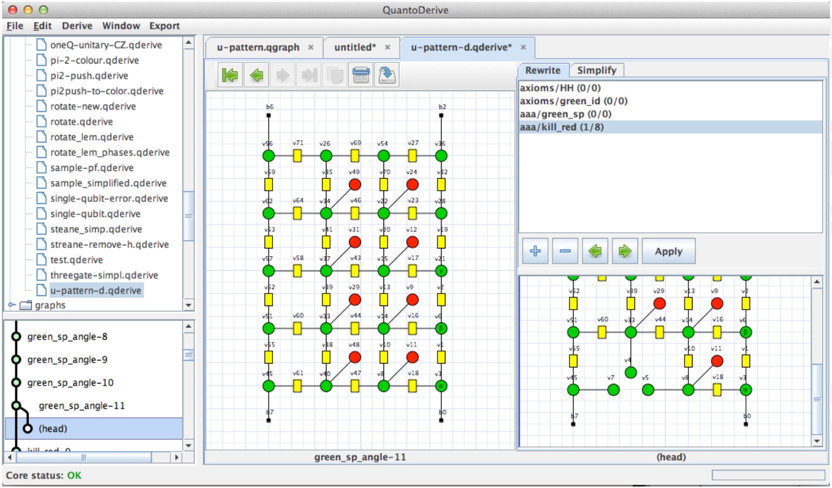
GHZ/Mermin non-locality

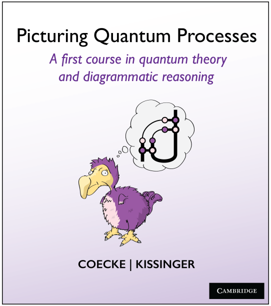


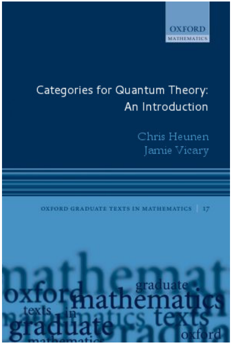
SLOCC-classification of 3 qubits:



Quantomatic:







- Categorical Quantum Mechanics I: Causal Quantum Processes. Coecke and Kissinger. arXiv:1510.05468
- Categorical Quantum Mechanics II: Classical-Quantum Interaction. Coecke and Kissinger. arXiv:1605.08617
- Categories of Quantum and Classical Channels. Coecke, Kissinger, Heunen. arXiv:1305.3821

Thanks! Joint work with:

...

Abramsky, Backens, Coecke, Duncan, Edwards, Gogioso, Hadzihasanovic, Heunen, Lal, Merry, Pavlovic, Paquette, Perdrix, Quick, Selinger, Vicary, Wang, Zamdzhiev, ...and many more!

http://quantomatic.github.io