Functional Quantum Programming

Thorsten Altenkirch

University of Nottingham

based on joint work with Jonathan Grattage

and discussions with V.P. Belavkin

Background

Background

- Simulation of quantum systems is expensive: PSPACE complexity for polynomial circuits.

Background

- Simulation of quantum systems is expensive: PSPACE complexity for polynomial circuits.
- Feynman: *Can we exploit this fact to perform computations more efficiently?*

Background

- Simulation of quantum systems is expensive: PSPACE complexity for polynomial circuits.
- Feynman: *Can we exploit this fact to perform computations more efficiently?*
- Shor: Factorisation in quantum polynomial time.

Background

- Simulation of quantum systems is expensive: PSPACE complexity for polynomial circuits.
- Feynman: *Can we exploit this fact to perform computations more efficiently?*
- Shor: Factorisation in quantum polynomial time.
- Grover: Blind search in $O(n/\sqrt{2})$

Background

- Simulation of quantum systems is expensive: PSPACE complexity for polynomial circuits.
- Feynman: *Can we exploit this fact to perform computations more efficiently?*
- Shor: Factorisation in quantum polynomial time.
- Grover: Blind search in $O(n/\sqrt{2})$
- Can we build a quantum computer?

Background

- Simulation of quantum systems is expensive: PSPACE complexity for polynomial circuits.
- Feynman: *Can we exploit this fact to perform computations more efficiently?*
- Shor: Factorisation in quantum polynomial time.
- Grover: Blind search in $O(n/\sqrt{2})$
- Can we build a quantum computer?

  yes We can run quantum algorithms.

Background

- Simulation of quantum systems is expensive: PSPACE complexity for polynomial circuits.
- Feynman: *Can we exploit this fact to perform computations more efficiently?*
- Shor: Factorisation in quantum polynomial time.
- Grover: Blind search in $O(n/\sqrt{2})$
- Can we build a quantum computer?

yes We can run quantum algorithms.

no Nature is classical after all!

Background

- Simulation of quantum systems is expensive: PSPACE complexity for polynomial circuits.
- Feynman: *Can we exploit this fact to perform computations more efficiently?*
- Shor: Factorisation in quantum polynomial time.
- Grover: Blind search in $O(n/\sqrt{2})$
- Can we build a quantum computer?

yes We can run quantum algorithms.

no Nature is classical after all!

Assumption: Nature is fair. . .

The quantum software crisis

The quantum software crisis

- Quantum algorithms are usually presented using the circuit model.

The quantum software crisis

- Quantum algorithms are usually presented using the circuit model.
- Nielsen and Chuang, p.7, *Coming up with good quantum algorithms is hard.*

The quantum software crisis

- Quantum algorithms are usually presented using the circuit model.
- Nielsen and Chuang, p.7, *Coming up with good quantum algorithms is hard.*
- Richard Josza, QPL 2004: *We need to develop quantum thinking!*

QML

QML

- QML: a functional language for quantum computations on finite types.

QML

- QML: a functional language for quantum computations on finite types.
- Quantum control **and** quantum data.

QML

- QML: a functional language for quantum computations on finite types.
- Quantum control **and** quantum data.
- Design guided by denotational semantics

QML

- QML: a functional language for quantum computations on finite types.
- Quantum control **and** quantum data.
- Design guided by denotational semantics
- Analogy with classical computation

  FCC Finite classical computations

  FQC Finite quantum computations

QML

- QML: a functional language for quantum computations on finite types.
- Quantum control **and** quantum data.
- Design guided by denotational semantics
- Analogy with classical computation

FCC	Finite classical computations
FQC	Finite quantum computations

- Important issue: **control of decoherence**

QML

- QML: a functional language for quantum computations on finite types.
- Quantum control **and** quantum data.
- Design guided by denotational semantics
- Analogy with classical computation

FCC	Finite classical computations
FQC	Finite quantum computations

- Important issue: **control of decoherence**
- Draft paper available (Google:Thorsten,functional,quantum)

QML

- QML: a functional language for quantum computations on finite types.
- Quantum control **and** quantum data.
- Design guided by denotational semantics
- Analogy with classical computation

$\mathbf{FCC}$	Finite classical computations
$\mathbf{FQC}$	Finite quantum computations

- Important issue: **control of decoherence**
- Draft paper available (Google:Thorsten,functional,quantum)
- Compiler under construction (Jonathan)

Example: Hadamard operation

Example: Hadamard operation

Matrix

$$\mathrm{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Example: Hadamard operation

Matrix

$$\mathrm{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

QML

$$\begin{aligned} \mathsf{H}\, x : \mathcal{Q}_2 = \mathtt{if}^{\circ}\ x\ & \mathtt{then}\ \{\mathtt{qfalse} \mid (-1)\mathtt{qtrue}\} \\ & \mathtt{else}\ \{\mathtt{qfalse} \mid \mathtt{qtrue}\} \end{aligned}$$

Related Work

- P. Zuliani, 2001, *Quantum Programming*
- S. Abramsky and B. Coecke, 2004, *A Categorical Semantics of Quantum Protocols*
- S-C. Mu and R. S. Bird, 2001, *Quantum functional programming*
- A. Sabry, 2003, *Modeling quantum computing in Haskell*
- J. Karczmarczuk, 2003, *Structure and interpretation of quantum mechanics: a functional framework*
- P. Selinger, 2002, *Towards a Quantum Programming Language*
- A. van Tonder, 2003, *A Lambda Calculus for Quantum Computation*

Something we know well . . .

Something we know well . . .

- Start with classical computations on finite types.

Something we know well . . .

- Start with classical computations on finite types.
- Quantum mechanics is time-reversible. . .

Something we know well . . .

- Start with classical computations on finite types.
- Quantum mechanics is time-reversible. . .
- . . . hence quantum computation is based on reversible operations.

Something we know well . . .

- Start with classical computations on finite types.
- Quantum mechanics is time-reversible. . .
- . . . hence quantum computation is based on reversible operations.
- **However:** Newtonian mechanics, Maxwellian electrodynamics is also time-reversible. . .

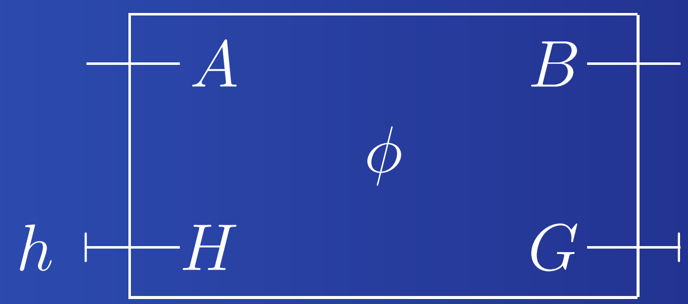
Something we know well . . .

- Start with classical computations on finite types.
- Quantum mechanics is time-reversible. . .
- . . . hence quantum computation is based on reversible operations.
- **However:** Newtonian mechanics, Maxwellian electrodynamics is also time-reversible. . .
- . . . hence classical computation **should be** based on reversible operations.

Classical computations (FCC)

Classical computations (FCC)

Given finite sets A (input) and B (output):



Classical computations (FCC)

Given finite sets A (input) and B (output):

- a finite set of initial heaps H,

Classical computations (FCC)

Given finite sets A (input) and B (output):

- a finite set of initial heaps H,
- an initial heap $h \in H$,

Classical computations (FCC)

Given finite sets A (input) and B (output):

- a finite set of initial heaps H,
- an initial heap $h \in H$,
- a finite set of garbage states G,

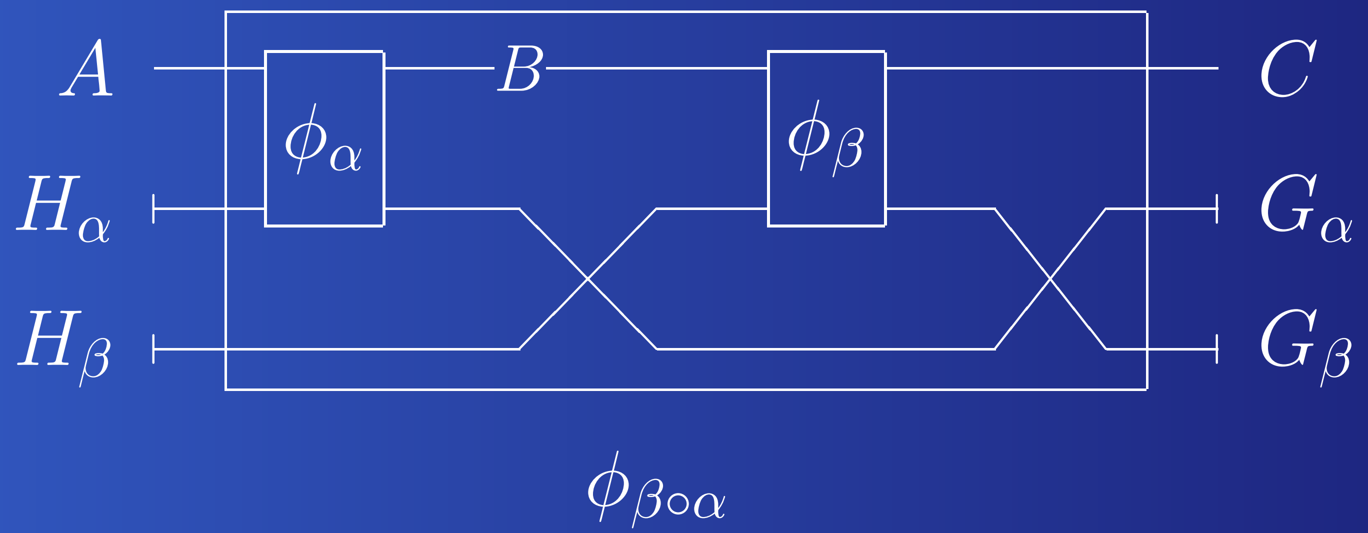
Classical computations (FCC)

Given finite sets A (input) and B (output):

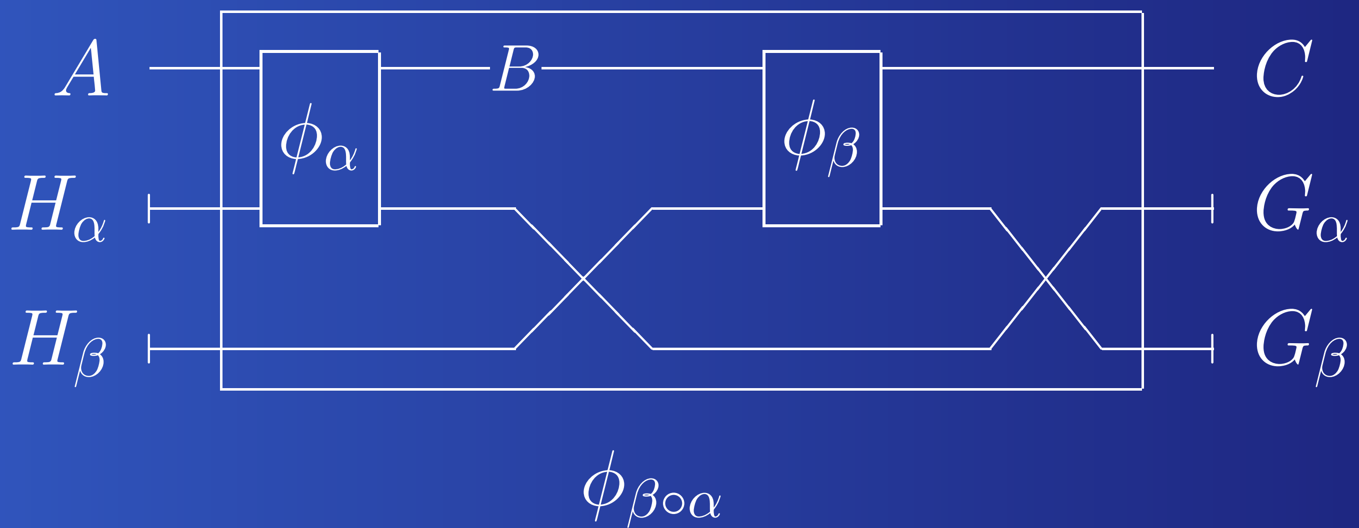
- a finite set of initial heaps H,
- an initial heap $h \in H$,
- a finite set of garbage states G,
- a bijection $\phi \in A \times H \simeq B \times G$,

Composing classical computations

Composing classical computations



Composing classical computations



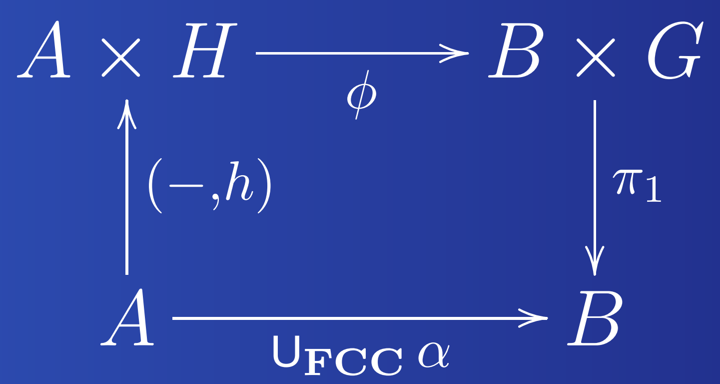
Exercise: Define I.

Extensional equality

Extensional equality

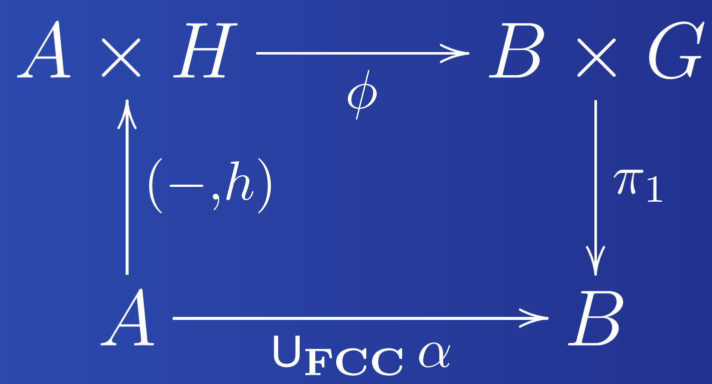
Every computation α gives rise to a function

$\mathsf{U}_{\mathbf{FCC}}\,\alpha \in A \to B$



Extensional equality

Every computation α gives rise to a function $\mathsf{U}_{\mathbf{FCC}}\,\alpha \in A \to B$



$$\alpha =_{\mathsf{ext}} \beta, \text{ if } \mathsf{U}_{\mathbf{FCC}}\,\alpha = \mathsf{U}_{\mathbf{FCC}}\,\beta$$

Extensional equality

Every computation α gives rise to a function

$\mathsf{U}_{\mathbf{FCC}}\,\alpha \in A \to B$

$$
\begin{array}{ccc}
A \times H & \xrightarrow[\phi]{} & B \times G \\
\big\uparrow{\scriptstyle (-,h)} & & \big\downarrow{\scriptstyle \pi_1} \\
A & \xrightarrow[\mathsf{U}_{\mathbf{FCC}}\,\alpha]{} & B
\end{array}
$$

$$\alpha =_{\mathsf{ext}} \beta, \text{ if } \mathsf{U}_{\mathbf{FCC}}\,\alpha = \mathsf{U}_{\mathbf{FCC}}\,\beta$$

$\mathbf{FCC}$:

Objects fi nite sets

Morphisms computations $/ =_{\mathsf{ext}}$.

$\mathsf{U}_{\mathbf{FCC}}$

$$
\begin{aligned}
\mathsf{U}_{\mathbf{FCC}}\, I &= I \\
\mathsf{U}_{\mathbf{FCC}}\, (\beta \circ \alpha) &= (\mathsf{U}_{\mathbf{FCC}}\, \beta) \circ (\mathsf{U}_{\mathbf{FCC}}\, \alpha)
\end{aligned}
$$

$\mathsf{U}_{\mathbf{FCC}}$

$$
\begin{aligned}
\mathsf{U}_{\mathbf{FCC}}\, I &= I \\
\mathsf{U}_{\mathbf{FCC}}\, (\beta \circ \alpha) &= (\mathsf{U}_{\mathbf{FCC}}\, \beta) \circ (\mathsf{U}_{\mathbf{FCC}}\, \alpha)
\end{aligned}
$$

- $\mathsf{U}_{\mathbf{FCC}}$ is a functor $\mathsf{U}_{\mathbf{FCC}} : \mathbf{FCC} \longrightarrow \mathbf{FinSet}$.

$\mathsf{U}_{\mathbf{FCC}}$

$$\begin{aligned} \mathsf{U}_{\mathbf{FCC}}\, I &= I \\ \mathsf{U}_{\mathbf{FCC}}\, (\beta \circ \alpha) &= (\mathsf{U}_{\mathbf{FCC}}\, \beta) \circ (\mathsf{U}_{\mathbf{FCC}}\, \alpha) \end{aligned}$$

- $\mathsf{U}_{\mathbf{FCC}}$ is a functor $\mathsf{U}_{\mathbf{FCC}} : \mathbf{FCC} \longrightarrow \mathbf{FinSet}$.
- $\mathsf{U}_{\mathbf{FCC}}$ is faithful (trivially).

$\mathsf{U}_{\mathbf{FCC}}$

$$\begin{aligned} \mathsf{U}_{\mathbf{FCC}}\, I &= I \\ \mathsf{U}_{\mathbf{FCC}}\, (\beta \circ \alpha) &= (\mathsf{U}_{\mathbf{FCC}}\, \beta) \circ (\mathsf{U}_{\mathbf{FCC}}\, \alpha) \end{aligned}$$

- $\mathsf{U}_{\mathbf{FCC}}$ is a functor $\mathsf{U}_{\mathbf{FCC}} : \mathbf{FCC} \longrightarrow \mathbf{FinSet}$.
- $\mathsf{U}_{\mathbf{FCC}}$ is faithful (trivially).
- **Exercise:** $\mathsf{U}_{\mathbf{FCC}}$ is full!

Coming next: Quantum computations

Develop FQC analogously to FCC. . .

Linear algebra revision

Linear algebra revision

Given a finite set A (the base)
$\mathbb{C}\,A = A \to \mathbb{C}$ is a **Hilbert space**.

Linear algebra revision

Given a finite set A (the base)

$\mathbb{C}\,A = A \to \mathbb{C}$ is a **Hilbert space**.

Linear operators:

$f \in A \to B \to \mathbb{C}$ induces $\hat{f} \in \mathbb{C}\,A \to \mathbb{C}\,B$.

we write $f \in A \multimap B$

Linear algebra revision

Given a finite set A (the base)

$\mathbb{C}\,A = A \to \mathbb{C}$ is a **Hilbert space**.

Linear operators:

$f \in A \to B \to \mathbb{C}$ induces $\hat{f} \in \mathbb{C}\,A \to \mathbb{C}\,B$.

we write $f \in A \multimap B$

Norm of a vector:

$||v|| = \Sigma_{a \in A}(va)^*(va) \in \mathbb{R}^+$,

Linear algebra revision

Given a finite set A (the base)

$\mathbb{C}\,A = A \to \mathbb{C}$ is a **Hilbert space**.

Linear operators:

$f \in A \to B \to \mathbb{C}$ induces $\hat{f} \in \mathbb{C}\,A \to \mathbb{C}\,B$.

we write $f \in A \multimap B$

Norm of a vector:

$\|v\| = \Sigma_{a \in A}(v\,a)^*(v\,a) \in \mathbb{R}^+$,

Unitary operators:

A unitary operator $\phi \in A \multimap_{\text{unitary}} B$ is a linear isomorphism that preserves the norm.

Basics of quantum computation

Basics of quantum computation

- A **pure state** over A is a vector $v \in \mathbb{C}\, A$ with unit norm $\|v\| = 1$.

Basics of quantum computation

- A **pure state** over A is a vector $v \in \mathbb{C}\ A$ with unit norm $\|v\| = 1$.
- A **reversible computation** is given by a unitary operator $\phi \in A \multimap_{\text{unitary}} B$.

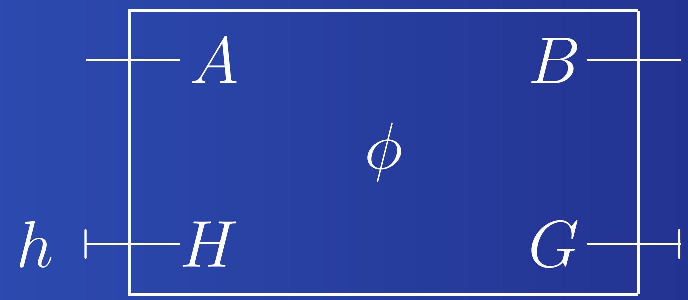
Quantum computations (FQC)

Quantum computations (FQC)

Given finite sets A (input) and B (output):

Quantum computations (FQC)

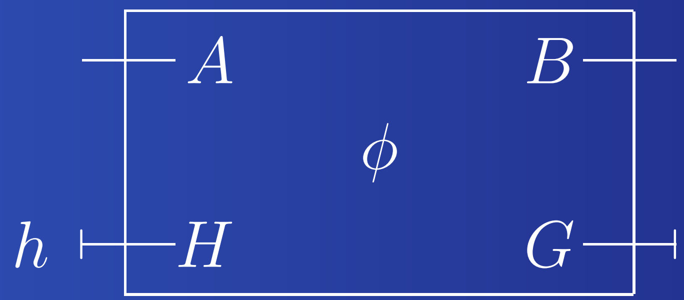
Given finite sets A (input) and B (output):



- a finite set H, the base of the space of initial heaps,

Quantum computations (FQC)

Given finite sets A (input) and B (output):



- a finite set H, the base of the space of initial heaps,
- a heap initialisation vector $h \in \mathbb{C}\, H$,

Quantum computations (FQC)

Given finite sets A (input) and B (output):

- a finite set H, the base of the space of initial heaps,
- a heap initialisation vector $h \in \mathbb{C}\,H$,
- a finite set G, the base of the space of garbage states,

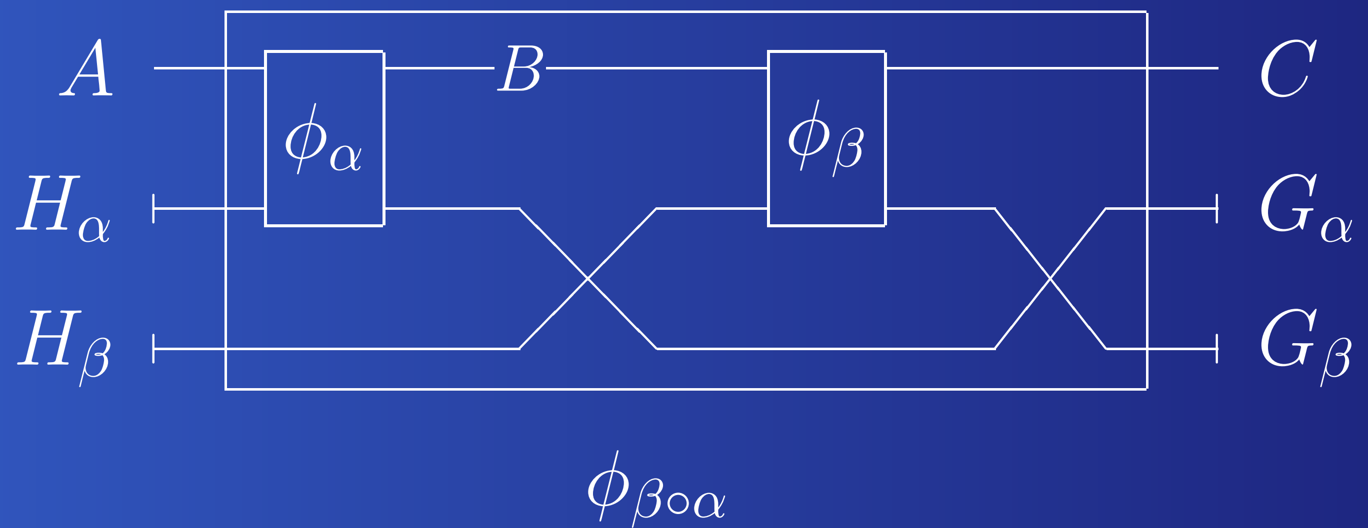
Quantum computations (FQC)

Given finite sets A (input) and B (output):

- a finite set H, the base of the space of initial heaps,
- a heap initialisation vector $h \in \mathbb{C}\, H$,
- a finite set G, the base of the space of garbage states,
- a unitary operator $\phi \in A \otimes H \multimap_{\text{unitary}} B \otimes G$.

Composing quantum computations

Composing quantum computations



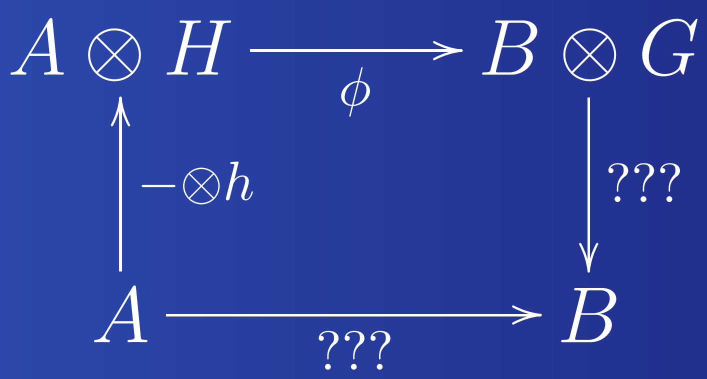
Extensional equality. . .

Extensional equality...

- ...is a bit more subtle.

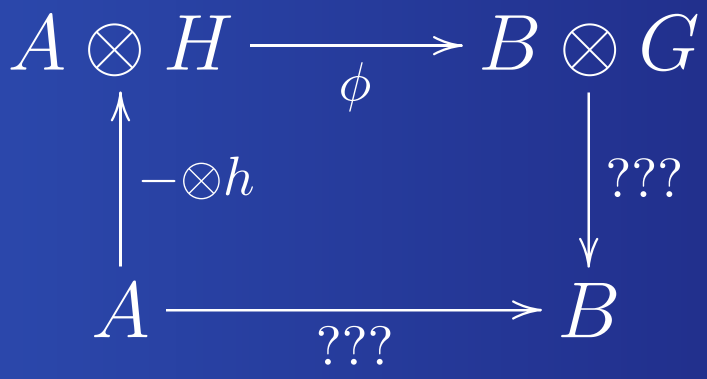
Extensional equality…

- …is a bit more subtle.
- There is no sensible operator replacing π_1 on vector spaces:



Extensional equality...

- ...is a bit more subtle.
- There is no sensible operator replacing π_1 on vector spaces:



- **Indeed:** Forgetting part of a **pure state** results in a **mixed state**.

Density Operators

A mixed state on A is given by a **density operator**

$$\rho \in A \multimap A$$

such that all eigenvalues are positive reals

$$\hat{\rho}\, v = \lambda v \implies \lambda \in \mathbb{R}^{+}$$

and has a unit trace

$$\Sigma a \in A . v\, a = 1$$

Superoperators

Superoperators

- A superoperator $f \in A \multimap_{\text{super}} B$ is a linear operator on density operators which is completely positive.

Superoperators

- A superoperator $f \in A \multimap_{\text{super}} B$ is a linear operator on density operators which is completely positive.
- A unitary operator $\phi \in A \multimap_{\text{unitary}} B$ gives rise to a superoperator $\phi^{\dagger} \in A \multimap_{\text{super}} B$.

Superoperators

- A superoperator $f \in A \multimap_{\text{super}} B$ is a linear operator on density operators which is completely positive.
- A unitary operator $\phi \in A \multimap_{\text{unitary}} B$ gives rise to a superoperator $\phi^{\dagger} \in A \multimap_{\text{super}} B$.
- Partial trace:

$$\mathrm{tr}_{A,G} \in A \otimes G \multimap_{\text{super}} A$$

Extensional equality

Extensional equality

Every computation α gives rise to a superoperator $\mathsf{U}\,\alpha \in A \multimap_{\mathsf{super}} B$

$$
\begin{array}{ccc}
A \otimes H & \xrightarrow{\quad \widehat{\phi} \quad} & B \otimes G \\
\uparrow {\scriptstyle -\otimes \widetilde{h}} & & \downarrow {\scriptstyle \mathrm{tr}_G} \\
A & \xrightarrow{\quad \mathsf{U}_{\mathbf{FQC}}\,\alpha \quad} & B
\end{array}
$$

Extensional equality

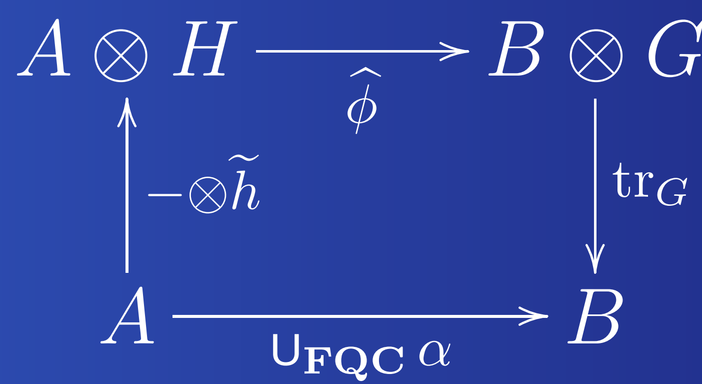
Every computation α gives rise to a superoperator $\mathsf{U}\,\alpha \in A \multimap_{\mathsf{super}} B$

$$
\begin{array}{ccc}
A \otimes H & \xrightarrow[\widehat{\phi}]{} & B \otimes G \\
\Big\uparrow {\scriptstyle -\otimes\widetilde{h}} & & \Big\downarrow {\scriptstyle \mathrm{tr}_G} \\
A & \xrightarrow[\mathsf{U}_{\mathbf{FQC}}\,\alpha]{} & B
\end{array}
$$

$$\alpha =_{\mathsf{ext}} \beta, \text{ if } \mathsf{U}_{\mathbf{FQC}}\,\alpha = \mathsf{U}_{\mathbf{FQC}}\,\beta$$

Extensional equality

Every computation α gives rise to a superoperator U $\alpha \in A \multimap_{\text{super}} B$



$$\alpha =_{\text{ext}} \beta, \text{if } \mathsf{U}_{\mathbf{FQC}}\,\alpha = \mathsf{U}_{\mathbf{FQC}}\,\beta$$

FCC:
- **Objects** finite sets
- **Morphisms** computations / $=_{\text{ext}}$.

UFQC

$\mathsf{U}_{\mathbf{FQC}}$

$$\begin{aligned}
\mathsf{U}_{\mathbf{FQC}}\, I &= I \\
\mathsf{U}_{\mathbf{FQC}}\, (\beta \circ \alpha) &= (\mathsf{U}_{\mathbf{FQC}}\, \beta) \circ (\mathsf{U}_{\mathbf{FQC}}\, \alpha)
\end{aligned}$$

$\mathsf{U}_{\mathbf{FQC}}$

$$
\begin{aligned}
\mathsf{U}_{\mathbf{FQC}}\, I &= I \\
\mathsf{U}_{\mathbf{FQC}}\, (\beta \circ \alpha) &= (\mathsf{U}_{\mathbf{FQC}}\, \beta) \circ (\mathsf{U}_{\mathbf{FQC}}\, \alpha)
\end{aligned}
$$

- $\mathsf{U}_{\mathbf{FQC}}$ is a functor $\mathsf{U}_{\mathbf{FQC}} : \mathbf{FQC} \rightarrow \mathbf{Super}$.

$\mathsf{U}_{\mathbf{FQC}}$

$$\begin{aligned} \mathsf{U}_{\mathbf{FQC}}\, I &= I \\ \mathsf{U}_{\mathbf{FQC}}\, (\beta \circ \alpha) &= (\mathsf{U}_{\mathbf{FQC}}\, \beta) \circ (\mathsf{U}_{\mathbf{FQC}}\, \alpha) \end{aligned}$$

- $\mathsf{U}_{\mathbf{FQC}}$ is a functor $\mathsf{U}_{\mathbf{FQC}} : \mathbf{FQC} \to \mathbf{Super}$.
- $\mathsf{U}_{\mathbf{FQC}}$ is faithful (trivially).

$\mathsf{U}_{\mathbf{FQC}}$

$$\begin{aligned} \mathsf{U}_{\mathbf{FQC}}\, I &= I \\ \mathsf{U}_{\mathbf{FQC}}\,(\beta \circ \alpha) &= (\mathsf{U}_{\mathbf{FQC}}\,\beta) \circ (\mathsf{U}_{\mathbf{FQC}}\,\alpha) \end{aligned}$$

- $\mathsf{U}_{\mathbf{FQC}}$ is a functor $\mathsf{U}_{\mathbf{FQC}} : \mathbf{FQC} \to \mathbf{Super}$.
- $\mathsf{U}_{\mathbf{FQC}}$ is faithful (trivially).
- $\mathsf{U}_{\mathbf{FQC}}$ is full!

Classical vs quantum

Classical vs quantum

classical	quantum

Classical vs quantum

classical	quantum
finite sets	

Classical vs quantum

classical	quantum
finite sets	finite dimensional Hilbert spaces

Classical vs quantum

classical	quantum
finite sets	finite dimensional Hilbert spaces
bijections	

Classical vs quantum

classical	quantum
finite sets	finite dimensional Hilbert spaces
bijections	unitary operators

Classical vs quantum

classical	quantum
finite sets	finite dimensional Hilbert spaces
bijections	unitary operators
cartesian product ($\times$)	

Classical vs quantum

classical	quantum
finite sets	finite dimensional Hilbert spaces
bijections	unitary operators
cartesian product ($\times$)	tensor product ($\otimes$)

Classical vs quantum

classical	quantum
finite sets	finite dimensional Hilbert spaces
bijections	unitary operators
cartesian product ($\times$)	tensor product ($\otimes$)
functions	

Classical vs quantum

classical	quantum
finite sets	finite dimensional Hilbert spaces
bijections	unitary operators
cartesian product ($\times$)	tensor product ($\otimes$)
functions	superoperators

Classical vs quantum

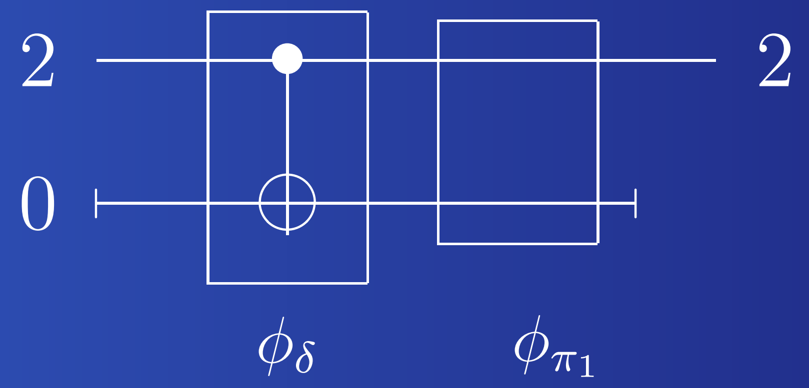
classical	quantum
finite sets	finite dimensional Hilbert spaces
bijections	unitary operators
cartesian product ($\times$)	tensor product ($\otimes$)
functions	superoperators
projections	

Classical vs quantum

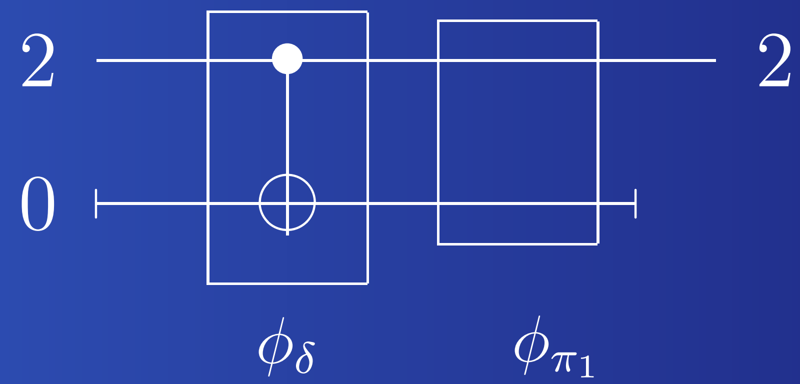
classical	quantum
finite sets	finite dimensional Hilbert spaces
bijections	unitary operators
cartesian product ($\times$)	tensor product ($\otimes$)
functions	superoperators
projections	partial trace

Decoherence

Decoherence



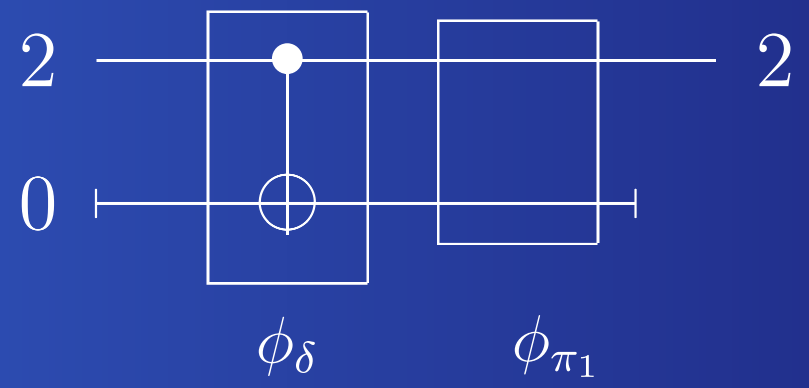
Decoherence



Classically

$$\pi_1 \circ \delta = \mathrm{I}$$

Decoherence

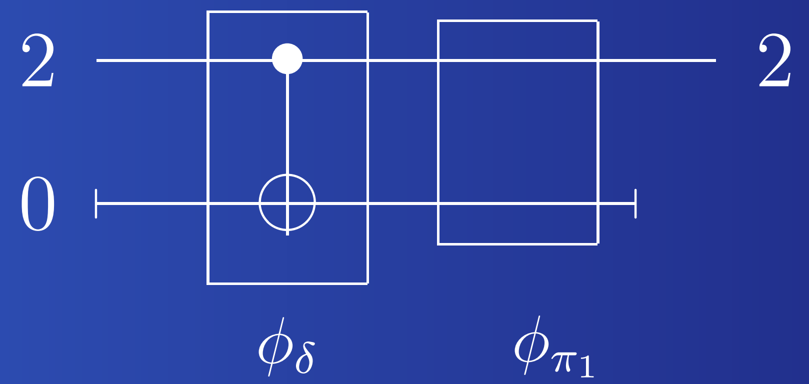


Classically

$$\pi_1 \circ \delta = \mathrm{I}$$

Quantum

Decoherence



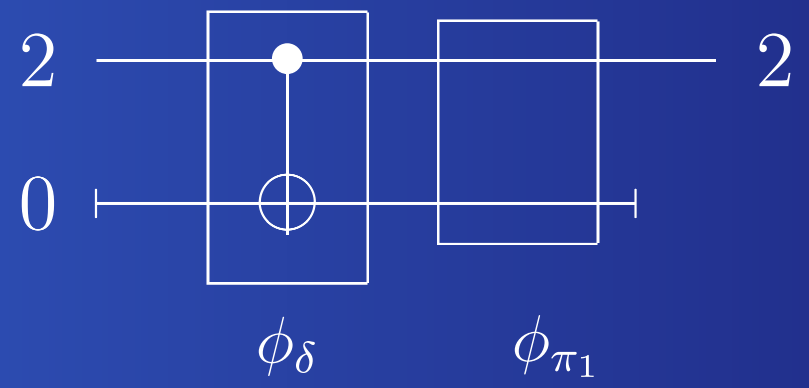
Classically

$$\pi_1 \circ \delta = \mathrm{I}$$

Quantum

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

Decoherence



Classically

$$\pi_1 \circ \delta = \mathrm{I}$$

Quantum

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

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

QML basics

QML basics

- $$\frac{\Gamma \vdash t : \sigma}{[\![t]\!] \in \mathbf{FQC}\,[\![\Gamma]\!]\,[\![\tau]\!]}$$

QML basics

- $$\frac{\Gamma \vdash t : \sigma}{[\![t]\!] \in \mathbf{FQC}\,[\![\Gamma]\!]\,[\![\tau]\!]}$$
- QML is based on strict linear logic
  no weakening but contraction.

QML basics

- $$\frac{\Gamma \vdash t : \sigma}{[\![t]\!] \in \mathbf{FQC}\ [\![\Gamma]\!]\ [\![\tau]\!]}$$
- QML is based on strict linear logic
  no weakening but contraction.
- QML types: $1, \sigma \otimes \tau, \sigma \oplus \tau$

Interpretation of types

Interpretation of types

$$
\begin{aligned}
|1| &= 0 \\
|\sigma \sqcup \tau| &= \max\{|\sigma|, |\tau|\} \\
|\sigma \oplus \tau| &= |\sigma \sqcup \tau| + 1 \\
|\sigma \otimes \tau| &= |\sigma| + |\tau|
\end{aligned}
$$

Interpretation of types

$$\begin{aligned} |1| &= 0 \\ |\sigma \sqcup \tau| &= \max\{|\sigma|, |\tau|\} \\ |\sigma \oplus \tau| &= |\sigma \sqcup \tau| + 1 \\ |\sigma \otimes \tau| &= |\sigma| + |\tau| \end{aligned}$$

$$[\![\sigma]\!] = 2^{|\sigma|}$$

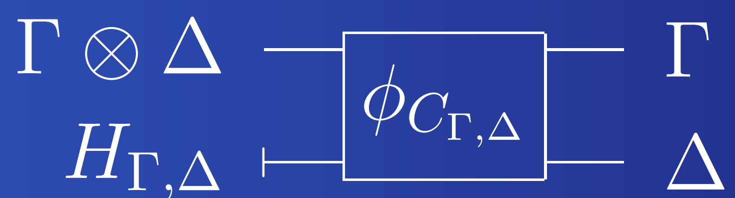
⊗ on contexts

$\otimes$ on contexts

$$
\begin{aligned}
\Gamma, x:\sigma \otimes \Delta, x:\sigma &= (\Gamma \otimes \Delta), x:\sigma \\
\Gamma, x:\sigma \otimes \Delta &= (\Gamma \otimes \Delta), x:\sigma \quad \text{if } x \notin \operatorname{dom} \Delta \\
\bullet \otimes \Delta &= \Delta
\end{aligned}
$$

$\otimes$ on contexts

$$
\begin{aligned}
\Gamma, x : \sigma \otimes \Delta, x : \sigma &= (\Gamma \otimes \Delta), x : \sigma \\
\Gamma, x : \sigma \otimes \Delta &= (\Gamma \otimes \Delta), x : \sigma \quad \text{if } x \notin \text{dom}\,\Delta \\
\bullet \otimes \Delta &= \Delta
\end{aligned}
$$



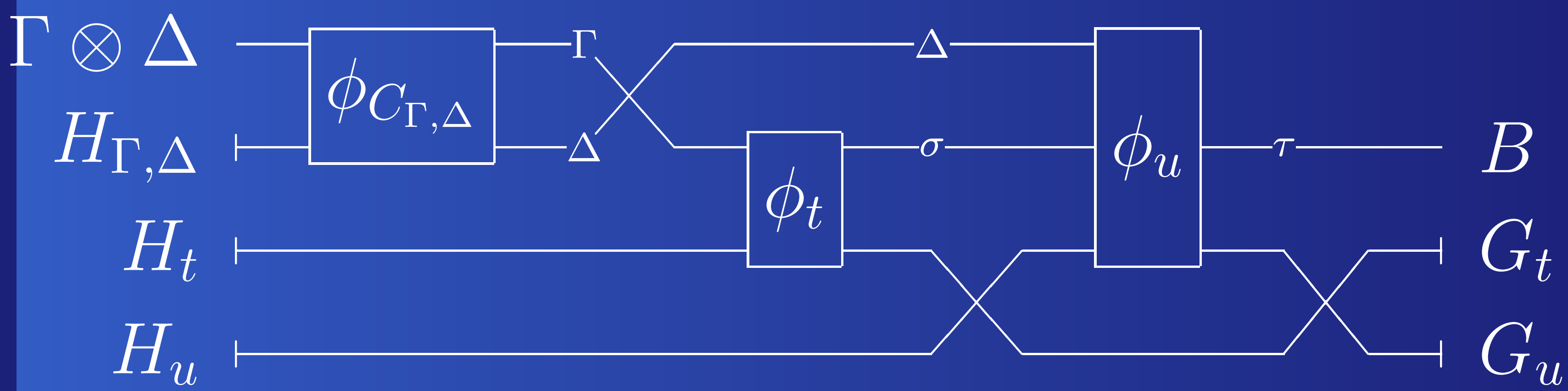
The let-rule

The let-rule

$$\frac{\Gamma \vdash t : \sigma \qquad \Delta, x : \sigma \vdash u : \tau}{\Gamma \otimes \Delta \vdash \texttt{let}\ x = t\ \texttt{in}\ u : \tau}\ \text{let}$$

The let-rule

$$\frac{\Gamma \vdash t : \sigma \qquad \Delta, x : \sigma \vdash u : \tau}{\Gamma \otimes \Delta \vdash \texttt{let}\ x = t\ \texttt{in}\ u : \tau}\ \text{let}$$



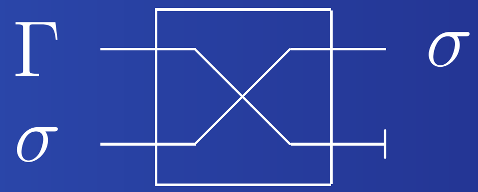
The var-rule

The var-rule

$$\frac{}{\Gamma, x : \sigma \vdash x^{\mathsf{dom}\,\Gamma} : \sigma}\ \text{var}$$

The var-rule

$$\frac{}{\Gamma, x : \sigma \vdash x^{\mathsf{dom}\,\Gamma} : \sigma}\ \mathrm{var}$$



Example

$$y : \mathcal{Q}_2 \vdash \texttt{let}\ x = y\ \texttt{in}\ x^{\{\}} : \mathcal{Q}_2$$

Example

$$y : \mathcal{Q}_2 \vdash \texttt{let}\ x = y\ \texttt{in}\ x^{\{\}} : \mathcal{Q}_2$$

$$y : \mathcal{Q}_2 \vdash \texttt{let}\ x = y\ \texttt{in}\ x^{\{y\}} : \mathcal{Q}_2$$

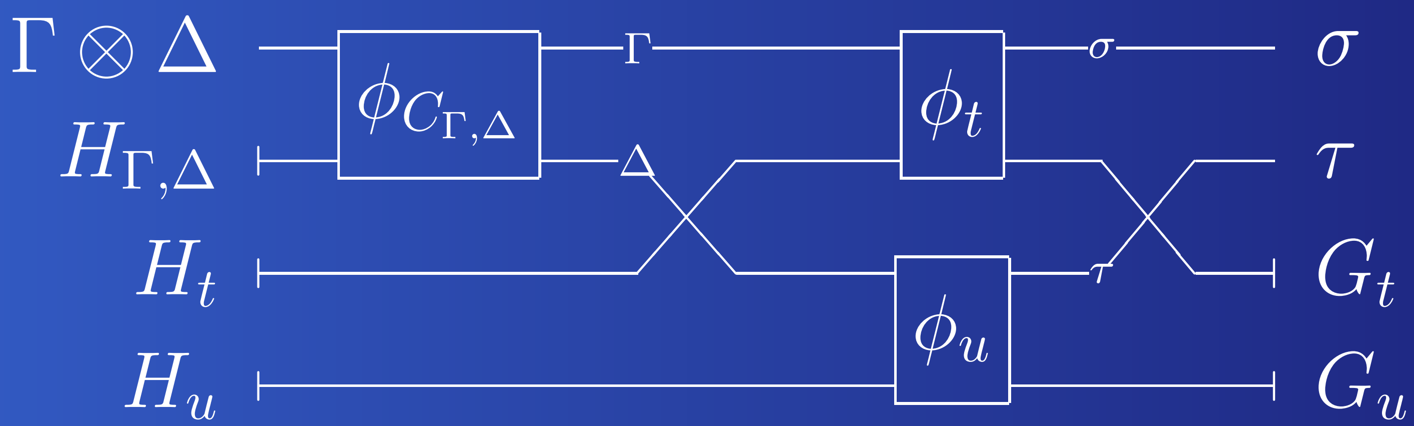
$\otimes$-intro

$\otimes$-intro

$$\frac{\Gamma \vdash t : \sigma \quad \Delta \vdash u : \tau}{\Gamma \otimes \Delta \vdash (t, u) : \sigma \otimes \tau} \otimes \mathrm{intro}$$

$\otimes$-intro

$$\frac{\Gamma \vdash t : \sigma \quad \Delta \vdash u : \tau}{\Gamma \otimes \Delta \vdash (t, u) : \sigma \otimes \tau} \otimes \text{intro}$$



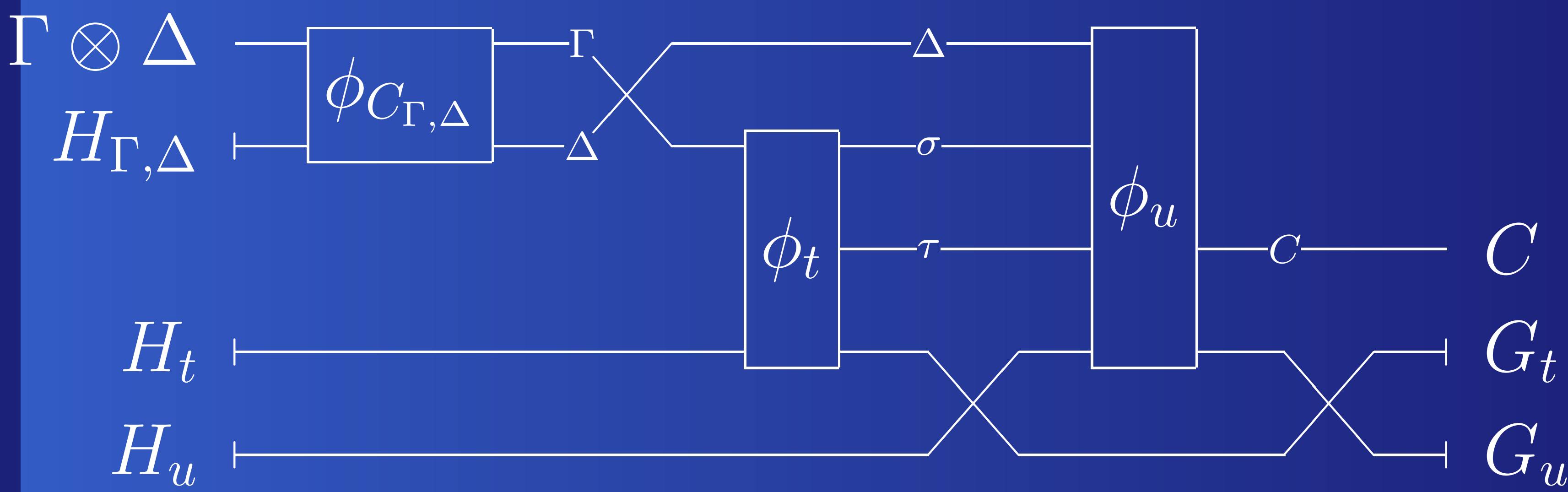
$\otimes$-elim

$\otimes$-elim

$$\frac{\Gamma \vdash t : \sigma \otimes \tau \qquad \Delta, x : \sigma, y : \tau \vdash u : C}{\Gamma \otimes \Delta \vdash \texttt{let}\ (x, y) = t\ \texttt{in}\ u : C} \otimes \mathrm{elim}$$

⊗-elim

$$\frac{\Gamma \vdash t : \sigma \otimes \tau \qquad \Delta, x : \sigma, y : \tau \vdash u : C}{\Gamma \otimes \Delta \vdash \texttt{let}\ (x, y) = t\ \texttt{in}\ u : C} \otimes \text{elim}$$



Example

$$p : \mathcal{Q}_2 \otimes \mathcal{Q}_2 \vdash \texttt{let}\ (x, y) = p\ \texttt{in}\ (y^{\{\}}, x^{\{\}}) : \mathcal{Q}_2 \otimes \mathcal{Q}_2$$

Example

$$p : \mathcal{Q}_2 \otimes \mathcal{Q}_2 \vdash \texttt{let}\ (x, y) = p\ \texttt{in}\ (y^{\{\}}, x^{\{\}}) : \mathcal{Q}_2 \otimes \mathcal{Q}_2$$

$$p : \mathcal{Q}_2 \otimes \mathcal{Q}_2 \vdash \texttt{let}\ (x, y) = p\ \texttt{in}\ (y^{\{p\}}, x^{\{p\}}) : \mathcal{Q}_2 \otimes \mathcal{Q}_2$$

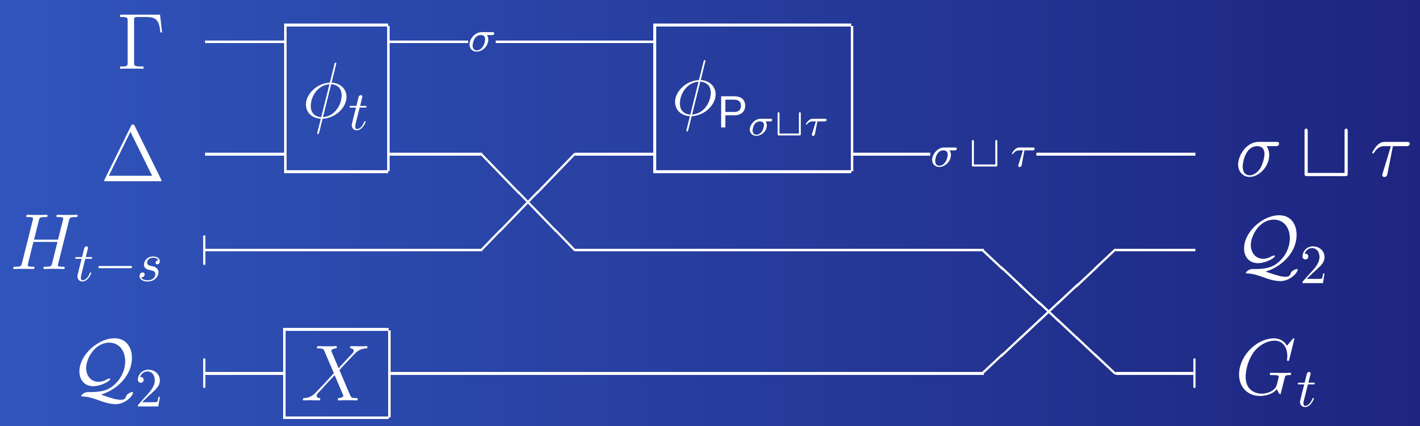
⊕-intro

$\oplus$-intro

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathtt{inl}\, t : A \oplus B}$$

$\oplus$-intro

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \texttt{inl}\, t : A \oplus B}$$



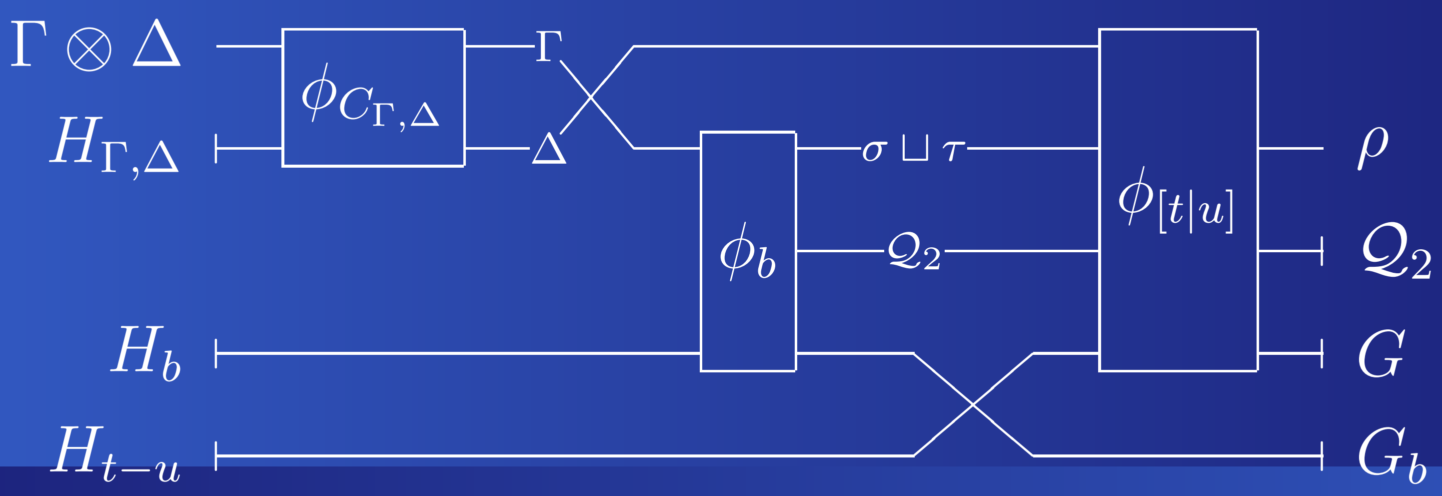
$\oplus$-elim

$\oplus$-elim

$$\frac{\Gamma \vdash c : \sigma \oplus \tau \qquad \Delta, x : \sigma \vdash t : \rho \qquad \Delta, y : \tau \vdash u : \rho}{\Gamma \otimes \Delta \vdash \texttt{case}\ c\ \texttt{of}\ \{\texttt{inl}\ x \Rightarrow t \mid \texttt{inr}\ y \Rightarrow u\} : \rho}\ +\mathrm{elim}$$

$\oplus$-elim

$$\frac{\Gamma \vdash c : \sigma \oplus \tau \qquad \Delta, x : \sigma \vdash t : \rho \qquad \Delta, y : \tau \vdash u : \rho}{\Gamma \otimes \Delta \vdash \texttt{case}\ c\ \texttt{of}\ \{\texttt{inl}\, x \Rightarrow t \mid \texttt{inr}\, y \Rightarrow u\} : \rho} + \mathrm{elim}$$



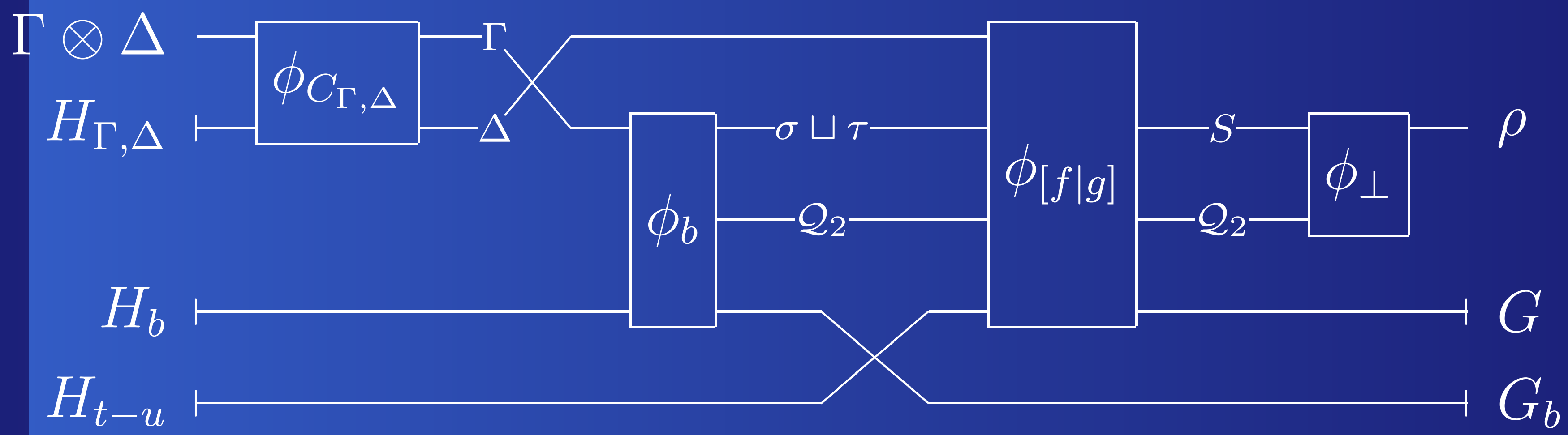
⊕-elim decoherence-free

$\oplus$-elim decoherence-free

$$\frac{\Gamma \vdash c : \sigma \oplus \tau \qquad \Delta, x : \sigma \vdash t : \rho \qquad \Delta, y : \tau \vdash u : \rho, \quad t \perp u}{\Gamma \otimes \Delta \vdash \texttt{case}^{\circ}\ b\ \texttt{of}\ \{\texttt{inl}\, x \Rightarrow t \,|\, \texttt{inr}\, y \Rightarrow u\} : \rho} + \mathrm{elim}^{\circ}$$

$\oplus$-elim decoherence-free

$$\frac{\Gamma \vdash c : \sigma \oplus \tau \qquad \Delta, x : \sigma \vdash t : \rho \qquad \Delta, y : \tau \vdash u : \rho, \quad t \perp u}{\Gamma \otimes \Delta \vdash \texttt{case}^{\circ}\ b\ \texttt{of}\ \{\texttt{inl}\ x \Rightarrow t \mid \texttt{inr}\ y \Rightarrow u\} : \rho} + \mathrm{elim}^{\circ}$$



Orthogonality

$$\frac{}{\texttt{inl}\ t \perp \texttt{inr}\ u} \qquad \frac{t \perp u}{\texttt{inl}\ t \perp \texttt{inl}\ u \quad \texttt{inr}\ t \perp \texttt{inr}\ u}$$

$$\frac{t \perp u}{(t, v) \perp (u, w) \quad (v, t) \perp (w, u)}$$

Semantics of $\perp$

$$[\![t \perp u]\!] = (S, \phi, f, g)$$

- S finite set.
- $\phi \in \mathcal{Q}_2 \otimes S \multimap_{\text{unitary}} [\![\sigma]\!]$
- $f \in \mathbf{FQC}\,[\![\Gamma]\!]\,S$
  $g \in \mathbf{FQC}\,[\![\Gamma]\!]\,S$
- $[\![t]\!] = \phi \circ (\mathrm{true} \otimes -) \circ f,$
  $[\![u]\!] = \phi \circ (\mathrm{false} \otimes -) \circ g$

Superpositions

$$\frac{\Gamma \vdash t, u : \sigma \qquad t \perp u \qquad ||\lambda||^2 + ||\lambda'||^2 = 1 \quad \lambda, \lambda' \neq 0}{\Gamma \vdash \{(\lambda)t \,|\, (\lambda')u\} : \sigma}$$

$$\equiv \ \texttt{if}^\circ \ \{(\lambda)\texttt{qtrue} \,|\, (\lambda')\texttt{qfalse}\} \ \texttt{then} \ t \ \texttt{else} \ u$$

Example: Deutsch's algorithm

$$
\begin{aligned}
\mathrm{Eq}\, a : \mathcal{Q}_2, b : \mathcal{Q}_2 = \texttt{let}\,(x,y) = \;& \texttt{if}^\circ\ \{\texttt{qfalse} \mid (-1)\texttt{qtrue}\} \\
& \texttt{then}\ \ (\texttt{qtrue},\ \texttt{if}\ \ a \\
& \qquad\qquad\qquad \texttt{then}\ \{\texttt{qfalse} \mid (-1)\texttt{qtrue}\} \\
& \qquad\qquad\qquad \texttt{else}\ \{\texttt{qfalse} \mid \texttt{qtrue}\}) \\
& \texttt{else}\ \ (\texttt{qfalse},\texttt{if}\ \ b \\
& \qquad\qquad\qquad \texttt{then}\ \{\texttt{qfalse} \mid (-1)\texttt{qtrue}\} \\
& \qquad\qquad\qquad \texttt{else}\ \{\texttt{qfalse} \mid \texttt{qtrue}\}) \\
\texttt{in}\ x & \\
: \ \mathcal{Q}_2 &
\end{aligned}
$$

Future work

Future work

- Higher order

Future work

- Higher order
- High level reasoning principles for QML programs

Future work

- Higher order
- High level reasoning principles for QML programs
- Categorical analysis

Future work

- Higher order
- High level reasoning principles for QML programs
- Categorical analysis
- Infi nite or indexed?