QML: A functional quantum programming language

quantum control and orthogonality

Jonathan Grattage
with Thorsten Altenkirch & Alex Green

sneezy.cs.nott.ac.uk/qml
School of Computer Science & IT,
The University of Nottingham

What is QML?

- A high–level quantum programming language with a structure familiar to functional programmers, which supports reasoning and and algorithm design

What is QML?

- A high–level quantum programming language with a structure familiar to functional programmers, which supports reasoning and and algorithm design
- Simplifing the design of quantum programs by:
  Allowing formal reasoning principles for quantum programs
  Giving a more intuitive understanding of quantum algorithms

What is QML?

- A high–level quantum programming language with a structure familiar to functional programmers, which supports reasoning and and algorithm design
- Simplifing the design of quantum programs by:
  Allowing formal reasoning principles for quantum programs
  Giving a more intuitive understanding of quantum algorithms
- *A functional quantum programming language*, LICS 2005
  with Thorsten Altenkirch

What is QML?

- A high–level quantum programming language with a structure familiar to functional programmers, which supports reasoning and and algorithm design
- Simplifing the design of quantum programs by:
  Allowing formal reasoning principles for quantum programs
  Giving a more intuitive understanding of quantum algorithms
- *A functional quantum programming language*, LICS 2005
  with Thorsten Altenkirch
- *An Algebra of Pure Quantum Programming*, QPL 2005
  with Thorsten Altenkirch, Juliana K. Vizzotto, & Amr Sabry

What is QML?

- A high–level quantum programming language with a structure familiar to functional programmers, which supports reasoning and and algorithm design
- Simplifing the design of quantum programs by:
  Allowing formal reasoning principles for quantum programs
  Giving a more intuitive understanding of quantum algorithms
- *A functional quantum programming language*, LICS 2005
  with Thorsten Altenkirch
- *An Algebra of Pure Quantum Programming*, QPL 2005
  with Thorsten Altenkirch, Juliana K. Vizzotto, & Amr Sabry
- Project Site: QML@CS.Nott – sneezy.cs.nott.ac.uk/qml

Quantum Languages

- P. Zuliani, PhD 2001, *Quantum Programming* (qGCL)
- P. Selinger, MSCS 2003, *Towards a Quantum Programming Language* (QPL)
- A. van Tonder, SIAM 2003, *A Lambda Calculus for Quantum Computation*
- A. Sabry, Haskell 2003, *Modeling quantum computing in Haskell*
- P. Selinger and B. Valiron, TLCA 2005, *A lambda calculus for quantum computation with classical control*
- . . .

Quantum Languages

- P. Zuliani, PhD 2001, *Quantum Programming* (qGCL)
- P. Selinger, MSCS 2003, *Towards a Quantum Programming Language* (QPL)
- A. van Tonder, SIAM 2003, *A Lambda Calculus for Quantum Computation*
- A. Sabry, Haskell 2003, *Modeling quantum computing in Haskell*
- P. Selinger and B. Valiron, TLCA 2005, *A lambda calculus for quantum computation with classical control*
- ...
- *Quantum* data, *Classical* control

QML Overview

- A first-order functional language for quantum computations on finite types

QML Overview

- A first-order functional language for quantum computations on finite types
- “Quantum Data *and* Control”

QML Overview

- A first-order functional language for quantum computations on finite types
- "Quantum Data *and* Control"
- Based on strict linear logic - controlled, explicit, weakening

QML Overview

The University of Nottingham

- A first-order functional language for quantum computations on finite types
- "Quantum Data *and* Control"
- Based on strict linear logic - controlled, explicit, weakening
- Operational Semantics: Quantum Circuit Model

QML Overview

- A first-order functional language for quantum computations on finite types
- “Quantum Data *and* Control”
- Based on strict linear logic - controlled, explicit, weakening
- Operational Semantics: Quantum Circuit Model
- Denotational Semantics: Superoperators

QML Overview

- A first-order functional language for quantum computations on finite types
- "Quantum Data *and* Control"
- Based on strict linear logic - controlled, explicit, weakening
- Operational Semantics: Quantum Circuit Model
- Denotational Semantics: Superoperators

- Notion of Finite Quantum Computations (**FQC**) developed by analogy with Finite Classical Computations (**FCC**)

Classical v. Quantum

Classical v. Quantum

Classical Case (**FCC**)	Quantum Case (**FQC**)

Classical v. Quantum

Classical Case (**FCC**)	Quantum Case (**FQC**)
Finite sets	

Classical v. Quantum

Classical Case (**FCC**)	Quantum Case (**FQC**)
Finite sets	Finite dimensional Hilbert spaces

Classical v. Quantum

Classical Case (**FCC**)	Quantum Case (**FQC**)
Finite sets	Finite dimensional Hilbert spaces
Cartesian product ($\times$)	

Classical v. Quantum

Classical Case (**FCC**)	Quantum Case (**FQC**)
Finite sets	Finite dimensional Hilbert spaces
Cartesian product ($\times$)	Tensor product ($\otimes$)

Classical v. Quantum

Classical Case (**FCC**)	Quantum Case (**FQC**)
Finite sets	Finite dimensional Hilbert spaces
Cartesian product ($\times$)	Tensor product ($\otimes$)
Bijections	

Classical v. Quantum

Classical Case (**FCC**)	Quantum Case (**FQC**)
Finite sets	Finite dimensional Hilbert spaces
Cartesian product ($\times$)	Tensor product ($\otimes$)
Bijections	Unitary operators

Classical v. Quantum

Classical Case (**FCC**)	Quantum Case (**FQC**)
Finite sets	Finite dimensional Hilbert spaces
Cartesian product ($\times$)	Tensor product ($\otimes$)
Bijections	Unitary operators
Functions	

Classical v. Quantum

Classical Case (**FCC**)	Quantum Case (**FQC**)
Finite sets	Finite dimensional Hilbert spaces
Cartesian product ($\times$)	Tensor product ($\otimes$)
Bijections	Unitary operators
Functions	Superoperators

Classical v. Quantum

Classical Case (**FCC**)	Quantum Case (**FQC**)
Finite sets	Finite dimensional Hilbert spaces
Cartesian product ($\times$)	Tensor product ($\otimes$)
Bijections	Unitary operators
Functions	Superoperators
Injective functions	

Classical v. Quantum

Classical Case (**FCC**)	Quantum Case (**FQC**)
Finite sets	Finite dimensional Hilbert spaces
Cartesian product ($\times$)	Tensor product ($\otimes$)
Bijections	Unitary operators
Functions	Superoperators
Injective functions	Isometries

Classical v. Quantum

Classical Case (**FCC**)	Quantum Case (**FQC**)
Finite sets	Finite dimensional Hilbert spaces
Cartesian product ($\times$)	Tensor product ($\otimes$)
Bijections	Unitary operators
Functions	Superoperators
Injective functions	Isometries
Projections	

Classical v. Quantum

The University of Nottingham

Classical Case (**FCC**)	Quantum Case (**FQC**)
Finite sets	Finite dimensional Hilbert spaces
Cartesian product ($\times$)	Tensor product ($\otimes$)
Bijections	Unitary operators
Functions	Superoperators
Injective functions	Isometries
Projections	Partial trace

QML Syntax

- Types:

$$\sigma = \mathcal{Q}_{\mathbf{1}} \mid \mathcal{Q}_{\mathbf{2}} \mid \sigma \otimes \tau$$

QML Syntax

- Types:

$$\sigma = \mathcal{Q}_{\mathbf{1}} \mid \mathcal{Q}_{\mathbf{2}} \mid \sigma \otimes \tau$$

- Syntax:

$$\begin{array}{lll}
(\mathit{Variables}) & x, y, \ldots \in & \mathit{Vars} \\
(\mathit{Prob.amplitudes}) & \kappa, \iota, \ldots \in & \mathbb{C} \\
(\mathit{Patterns}) & p, q & ::= x \mid (x, y) \\
(\mathit{Terms}) & t, u, e & ::= x \mid x^{\vec{y}} \mid () \mid (t, u) \\
 & & \mid \quad \mathbf{let}\ p = t\ \mathbf{in}\ u \\
 & & \mid \quad \mathbf{if}\ \ t\ \mathbf{then}\ u\ \mathbf{else}\ u' \\
 & & \mid \quad \mathbf{if}^{\circ}\ t\ \mathbf{then}\ u\ \mathbf{else}\ u' \\
 & & \mid \quad \mathit{false} \mid \mathit{true} \mid \overrightarrow{0} \mid \kappa \times t \mid t + u
\end{array}$$

QML Syntax

- Types:

$$\sigma = \mathcal{Q}_{\mathbf{1}} \mid \mathcal{Q}_{\mathbf{2}} \mid \sigma \otimes \tau$$

- Syntax:

$$\begin{array}{llll}
(\mathit{Variables}) & x, y, \ldots & \in & \mathit{Vars} \\
(\mathit{Prob.amplitudes}) & \kappa, \iota, \ldots & \in & \mathbb{C} \\
(\mathit{Patterns}) & p, q & ::= & x \mid (x, y) \\
(\mathit{Terms}) & t, u, e & ::= & x \mid x^{\vec{y}} \mid () \mid (t, u) \\
 & & \mid & \mathbf{let}\ p = t\ \mathbf{in}\ u \\
 & & \mid & \mathbf{if}\ \ t\ \mathbf{then}\ u\ \mathbf{else}\ u' \\
 & & \mid & \mathbf{if}^{\circ}\ t\ \mathbf{then}\ u\ \mathbf{else}\ u' \\
 & & \mid & \mathit{false} \mid \mathit{true} \mid \overrightarrow{0} \mid \kappa \times t \mid t + u
\end{array}$$

- EPR State: $\frac{1}{\sqrt{2}} \times (\mathit{false}, \mathit{false}) + \frac{1}{\sqrt{2}} \times (\mathit{true}, \mathit{true})$

Control of Decoherence

- Projection Function

$$\pi_1 \in (\mathcal{Q}_{\mathbf{2}}, \mathcal{Q}_{\mathbf{2}}) \rightarrow \mathbf{C}_{\mathbf{2}}$$

$$\pi_1\ (x, y) = x$$

$$Q_2 \longrightarrow Q_2$$

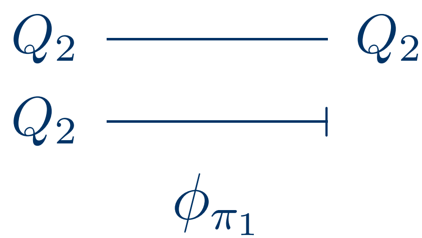
$$Q_2 \longmapsto$$

$$\phi_{\pi_1}$$

Control of Decoherence

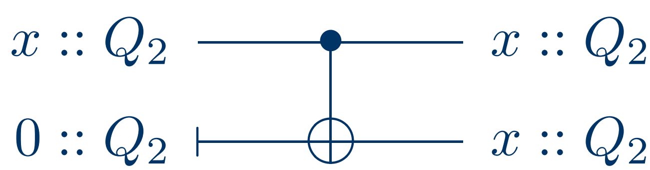
- Projection Function

$$\pi_1 \in (\mathcal{Q}_{\mathbf{2}}, \mathcal{Q}_{\mathbf{2}}) \longrightarrow \mathbf{C_2}$$
$$\pi_1\ (x, y) = x$$



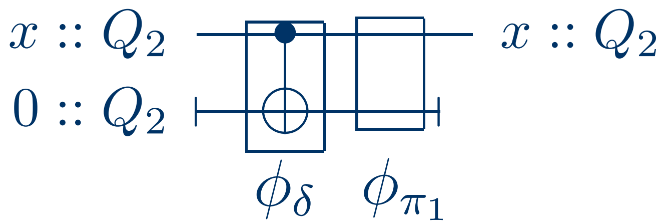
- Diagonal Function

$$\delta \in \mathcal{Q}_{\mathbf{2}} \longrightarrow (\mathcal{Q}_{\mathbf{2}}, \mathcal{Q}_{\mathbf{2}})$$
$$\delta\ x = (x, x)$$



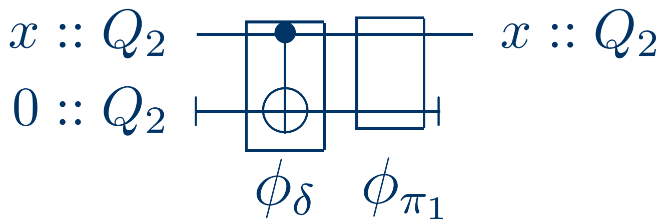
Control of Decoherence

- $\pi_1.\delta \in \mathcal{Q}_{\mathbf{2}} \to \mathcal{Q}_{\mathbf{2}}$



Control of Decoherence

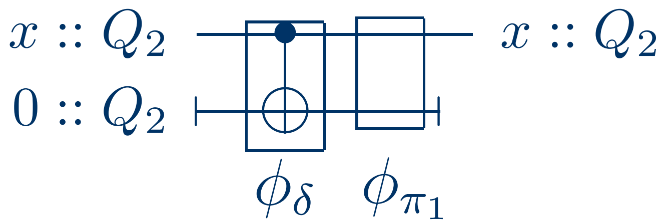
- $\pi_1.\delta \in \mathcal{Q}_{\mathbf{2}} \to \mathcal{Q}_{\mathbf{2}}$



- Classical Case:

Control of Decoherence

- $\pi_1 . \delta \in \mathcal{Q}_{\mathbf{2}} \to \mathcal{Q}_{\mathbf{2}}$



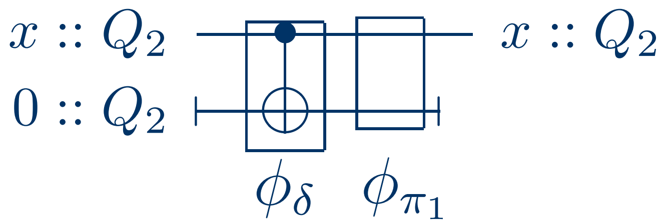
- Classical Case:

$$Q_2 \longrightarrow Q_2$$

- Quantum Case:
  Input $= \frac{1}{\sqrt{2}} \times \mathit{false} + \frac{1}{\sqrt{2}} \times \mathit{true}$ (equal superposition)

Control of Decoherence

- $\pi_1.\delta \in \mathcal{Q}_{\mathbf{2}} \to \mathcal{Q}_{\mathbf{2}}$

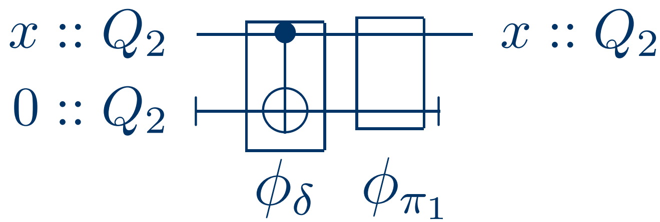


- Classical Case:

- Quantum Case:
  Input $= \frac{1}{\sqrt{2}} \times false + \frac{1}{\sqrt{2}} \times true$ (equal superposition)
  Output $= \{\frac{1}{2}\}false + \{\frac{1}{2}\}true$ (probability distribution)

Control of Decoherence

- $\pi_1 . \delta \in \mathcal{Q}_\mathbf{2} \to \mathcal{Q}_\mathbf{2}$



- Classical Case:

$$Q_2 \longrightarrow Q_2$$

- Quantum Case:
  Input $= \frac{1}{\sqrt{2}} \times \mathit{false} + \frac{1}{\sqrt{2}} \times \mathit{true}$ (equal superposition)
  Output $= \{\frac{1}{2}\}\mathit{false} + \{\frac{1}{2}\}\mathit{true}$ (probability distribution)

  Decoherence! Not the identity function

More Decoherence

More Decoherence

- *forget* mentions x

$$forget \in \mathcal{Q}_{\mathbf{2}} \multimap \mathcal{Q}_{\mathbf{2}}$$

$$forget\ x = \mathbf{if}\ x\ \mathbf{then}\ true\ \mathbf{else}\ true$$

More Decoherence

- *forget* mentions x

$$forget \in \mathcal{Q}_\mathbf{2} \multimap \mathcal{Q}_\mathbf{2}$$

$$forget\ x = \mathbf{if}\ x\ \mathbf{then}\ true\ \mathbf{else}\ true$$

- but doesn't use it.

More Decoherence

- forget mentions x

$$\mathit{forget} \in \mathcal{Q}_{\mathbf{2}} \multimap \mathcal{Q}_{\mathbf{2}}$$

$$\mathit{forget}\ x = \mathbf{if}\ x\ \mathbf{then}\ \mathit{true}\ \mathbf{else}\ \mathit{true}$$

- but doesn't use it.
- Hence, it **has** to measure it!

More Decoherence

- *forget* mentions x

$$forget \in \mathcal{Q}_{\mathbf{2}} \multimap \mathcal{Q}_{\mathbf{2}}$$

$$forget\ x = \mathbf{if}\ x\ \mathbf{then}\ true\ \mathbf{else}\ true$$

- but doesn't use it.
- Hence, it **has** to measure it!
- **if** always measures the conditional

More Decoherence

- forget mentions x

$$\mathit{forget} \in \mathcal{Q}_{\mathbf{2}} \multimap \mathcal{Q}_{\mathbf{2}}$$
$$\mathit{forget}\ x = \mathbf{if}\ x\ \mathbf{then}\ \mathit{true}\ \mathbf{else}\ \mathit{true}$$

- but doesn't use it.
- Hence, it **has** to measure it!
- **if** always measures the conditional
- Not, using **if**

$$\mathit{not} \in \mathcal{Q}_{\mathbf{2}} \multimap \mathcal{Q}_{\mathbf{2}}$$
$$\mathit{not}\ x = \mathbf{if}\ x\ \mathbf{then}\ \mathit{false}\ \mathbf{else}\ \mathit{true}$$

$\mathbf{if}^{\circ}$ – Quantum control

$\mathbf{if}^\circ$ – Quantum control

- $$\begin{aligned} &\mathit{forget}' \in \mathcal{Q}_{\mathbf{2}} \multimap \mathcal{Q}_{\mathbf{2}} \\ &\mathit{forget}'\ x = \mathbf{if}^\circ\ x\ \mathbf{then}\ \mathit{true}\ \mathbf{else}\ \mathit{true} \end{aligned}$$

$\mathbf{if}^{\circ}$ – Quantum control

- $$forget' \in \mathcal{Q}_{\mathbf{2}} \multimap \mathcal{Q}_{\mathbf{2}}$$
  $$forget'\ x = \mathbf{if}^{\circ}\ x\ \mathbf{then}\ true\ \mathbf{else}\ true$$
- This program has a type error, because $\mathrm{true} \not\perp \mathrm{true}$.

$\mathbf{if}^\circ$ – Quantum control

- $$\begin{aligned} &forget' \in Q_{\mathbf{2}} \multimap Q_{\mathbf{2}} \\ &forget'\ x = \mathbf{if}^\circ\ x\ \mathbf{then}\ true\ \mathbf{else}\ true \end{aligned}$$
- This program has a type error, because $\mathrm{true} \not\perp \mathrm{true}$.
- $$\begin{aligned} &qnot \in Q_{\mathbf{2}} \multimap Q_{\mathbf{2}} \\ &qnot\ x = \mathbf{if}^\circ\ x\ \mathbf{then}\ false\ \mathbf{else}\ true \end{aligned}$$

$\mathbf{if}^\circ$ – Quantum control

- $$forget' \in \mathcal{Q}_\mathbf{2} \multimap \mathcal{Q}_\mathbf{2}$$
  $$forget'\ x = \mathbf{if}^\circ\ x\ \mathbf{then}\ true\ \mathbf{else}\ true$$
- This program has a type error, because $\mathrm{true} \not\perp \mathrm{true}$.
- $$qnot \in \mathcal{Q}_\mathbf{2} \multimap \mathcal{Q}_\mathbf{2}$$
  $$qnot\ x = \mathbf{if}^\circ\ x\ \mathbf{then}\ false\ \mathbf{else}\ true$$
- This program typechecks, because $\mathrm{false} \perp \mathrm{true}$.

$\mathbf{if}^\circ$ – Quantum control

- $$forget' \in \mathcal{Q}_\mathbf{2} \multimap \mathcal{Q}_\mathbf{2}$$
  $$forget'\ x = \mathbf{if}^\circ\ x\ \mathbf{then}\ true\ \mathbf{else}\ true$$
- This program has a type error, because $\mathrm{true} \not\perp \mathrm{true}$.
- $$qnot \in \mathcal{Q}_\mathbf{2} \multimap \mathcal{Q}_\mathbf{2}$$
  $$qnot\ x = \mathbf{if}^\circ\ x\ \mathbf{then}\ false\ \mathbf{else}\ true$$
- This program typechecks, because $\mathrm{false} \perp \mathrm{true}$.
- $$cnot\ c\ x = \mathbf{if}^\circ\ c\ \mathbf{then}\ (true, qnot\ x)\ \mathbf{else}\ (false, x)$$

$\mathbf{if}^{\circ}$ – Quantum control

-
$$\begin{aligned} &\mathit{forget}' \in \mathcal{Q}_{\mathbf{2}} \multimap \mathcal{Q}_{\mathbf{2}} \\ &\mathit{forget}'\ x = \mathbf{if}^{\circ}\ x\ \mathbf{then}\ \mathit{true}\ \mathbf{else}\ \mathit{true} \end{aligned}$$
- This program has a type error, because $\mathrm{true} \not\perp \mathrm{true}$.
-
$$\begin{aligned} &\mathit{qnot} \in \mathcal{Q}_{\mathbf{2}} \multimap \mathcal{Q}_{\mathbf{2}} \\ &\mathit{qnot}\ x = \mathbf{if}^{\circ}\ x\ \mathbf{then}\ \mathit{false}\ \mathbf{else}\ \mathit{true} \end{aligned}$$
- This program typechecks, because $\mathrm{false} \perp \mathrm{true}$.
-
$$\mathit{cnot}\ c\ x = \mathbf{if}^{\circ}\ c\ \mathbf{then}\ (\mathit{true}, \mathit{qnot}\ x)\ \mathbf{else}\ (\mathit{false}, x)$$
- Deutsch-Joza Algorithm, Quantum Teleport Algorithm, ...

Quantum Teleport Algorithm

$$pZed \in \mathcal{Q}_\mathbf{2} \multimap \mathcal{Q}_\mathbf{2}$$

$$pZed\ x = \mathbf{if}^\circ\ x\ \mathbf{then}\ (-1) \times \mathit{true}\ \mathbf{else}\ \mathit{false}$$

$$had \in \mathcal{Q}_\mathbf{2} \multimap \mathcal{Q}_\mathbf{2}$$

$$had\ x = \mathbf{if}^\circ\ x\ \mathbf{then}\ (-1) \times \mathit{true} + \mathit{false}\ \mathbf{else}\ \mathit{true} + \mathit{false}$$

$$tel \in \mathcal{Q}_\mathbf{2} \multimap \mathcal{Q}_\mathbf{2}$$

$$\begin{aligned} tel\ x = \ \mathbf{let}\ (a, b) &= (\mathit{false}, \mathit{false}) + (\mathit{true}, \mathit{true}) \\ (a', x') &= cnot\ a\ x \\ b' &= \mathbf{if}\ a'\ \mathbf{then}\ qnot\ b\ \mathbf{else}\ b \\ b'' &= \mathbf{if}^\circ\ had\ x'\ \mathbf{then}\ pZed\ b'\ \mathbf{else}\ b' \\ \mathbf{in}\ b'' & \end{aligned}$$

Inner Product & $\perp$

- We define the inner product of terms, which to any pair of terms $\Gamma \vdash t, u : \sigma$ assigns $\langle t|u\rangle \in \mathbb{C} \cup \{?\}$.

Inner Product & $\perp$

- We define the inner product of terms, which to any pair of terms $\Gamma \vdash t, u : \sigma$ assigns $\langle t|u\rangle \in \mathbb{C} \cup \{?\}$.
- $t \perp u$ holds if $\langle t|u\rangle = 0$.

Inner Product & $\perp$

- We define the inner product of terms, which to any pair of terms $\Gamma \vdash t, u : \sigma$ assigns $\langle t|u\rangle \in \mathbb{C} \cup \{?\}$.
- $t \perp u$ holds if $\langle t|u\rangle = 0$.
- $\langle \mathrm{t}|\mathrm{t}\rangle = 1, \quad \langle \mathrm{false}|\mathrm{true}\rangle = 0, \quad \langle \mathrm{true}|\mathrm{false}\rangle = 0$

Inner Product & $\perp$

- We define the inner product of terms, which to any pair of terms $\Gamma \vdash t, u : \sigma$ assigns $\langle t|u\rangle \in \mathbb{C} \cup \{?\}$.
- $t \perp u$ holds if $\langle t|u\rangle = 0$.
- $\langle \mathrm{t}|\mathrm{t}\rangle = 1, \quad \langle \mathrm{false}|\mathrm{true}\rangle = 0, \quad \langle \mathrm{true}|\mathrm{false}\rangle = 0$
- $\langle \overrightarrow{0}|\mathrm{true}\rangle = 0 = \langle \mathrm{true}|\overrightarrow{0}\rangle, \quad \langle \overrightarrow{0}|\mathrm{false}\rangle = 0 = \langle \mathrm{false}|\overrightarrow{0}\rangle,$
  $\langle \overrightarrow{0}|\mathrm{x}\rangle = 0 = \langle \mathrm{x}|\overrightarrow{0}\rangle$

Inner Product & $\perp$

- We define the inner product of terms, which to any pair of terms $\Gamma \vdash t, u : \sigma$ assigns $\langle t|u\rangle \in \mathbb{C} \cup \{?\}$.
- $t \perp u$ holds if $\langle t|u\rangle = 0$.
- $\langle \mathrm{t}|\mathrm{t}\rangle = 1, \quad \langle \mathrm{false}|\mathrm{true}\rangle = 0, \quad \langle \mathrm{true}|\mathrm{false}\rangle = 0$
- $\langle \overrightarrow{0}|\mathrm{true}\rangle = 0 = \langle \mathrm{true}|\overrightarrow{0}\rangle, \quad \langle \overrightarrow{0}|\mathrm{false}\rangle = 0 = \langle \mathrm{false}|\overrightarrow{0}\rangle,$
  $\langle \overrightarrow{0}|\mathrm{x}\rangle = 0 = \langle \mathrm{x}|\overrightarrow{0}\rangle$
- $\langle (t, t') \mid (u, u')\rangle = \langle t|u\rangle * \langle t'|u'\rangle$

Inner Product & $\perp$

- We define the inner product of terms, which to any pair of terms $\Gamma \vdash t, u : \sigma$ assigns $\langle t|u \rangle \in \mathbb{C} \cup \{?\}$.
- $t \perp u$ holds if $\langle t|u \rangle = 0$.
- $\langle \mathrm{t}|\mathrm{t} \rangle = 1, \quad \langle \mathrm{false}|\mathrm{true} \rangle = 0, \quad \langle \mathrm{true}|\mathrm{false} \rangle = 0$
- $\langle \overrightarrow{0}|\mathrm{true} \rangle = 0 = \langle \mathrm{true}|\overrightarrow{0} \rangle, \quad \langle \overrightarrow{0}|\mathrm{false} \rangle = 0 = \langle \mathrm{false}|\overrightarrow{0} \rangle,$
  $\langle \overrightarrow{0}|\mathrm{x} \rangle = 0 = \langle \mathrm{x}|\overrightarrow{0} \rangle$
- $\langle (t, t') \mid (u, u') \rangle = \langle t|u \rangle * \langle t'|u' \rangle$
- $\langle \lambda * t + \lambda' * t' \mid u \rangle = \lambda^* * \langle t|u \rangle + \lambda'^* * \langle t'|u \rangle,$
  $\langle t \mid \kappa * u + \kappa' * u' \rangle = \kappa * \langle t|u \rangle + \kappa' * \langle t|u' \rangle$

Inner Product & $\perp$

- We define the inner product of terms, which to any pair of terms $\Gamma \vdash t, u : \sigma$ assigns $\langle t|u\rangle \in \mathbb{C} \cup \{?\}$.
- $t \perp u$ holds if $\langle t|u\rangle = 0$.
- $\langle \mathrm{t}|\mathrm{t}\rangle = 1, \quad \langle \mathrm{false}|\mathrm{true}\rangle = 0, \quad \langle \mathrm{true}|\mathrm{false}\rangle = 0$
- $\langle \overrightarrow{0}|\mathrm{true}\rangle = 0 = \langle \mathrm{true}|\overrightarrow{0}\rangle, \quad \langle \overrightarrow{0}|\mathrm{false}\rangle = 0 = \langle \mathrm{false}|\overrightarrow{0}\rangle,$
  $\langle \overrightarrow{0}|\mathrm{x}\rangle = 0 = \langle \mathrm{x}|\overrightarrow{0}\rangle$
- $\langle (t, t') \mid (u, u')\rangle = \langle t|u\rangle * \langle t'|u'\rangle$
- $\langle \lambda * t + \lambda' * t' \mid u\rangle = \lambda^* * \langle t|u\rangle + \lambda'^* * \langle t'|u\rangle,$
  $\langle t \mid \kappa * u + \kappa' * u'\rangle = \kappa * \langle t|u\rangle + \kappa' * \langle t|u'\rangle$
- $\ldots$

Inner Product & $\perp$

- We define the inner product of terms, which to any pair of terms $\Gamma \vdash t, u : \sigma$ assigns $\langle t|u\rangle \in \mathbb{C} \cup \{?\}$.
- $t \perp u$ holds if $\langle t|u\rangle = 0$.
- $\langle \mathrm{t}|\mathrm{t}\rangle = 1, \quad \langle \mathrm{false}|\mathrm{true}\rangle = 0, \quad \langle \mathrm{true}|\mathrm{false}\rangle = 0$
- $\langle \overrightarrow{0}|\mathrm{true}\rangle = 0 = \langle \mathrm{true}|\overrightarrow{0}\rangle, \quad \langle \overrightarrow{0}|\mathrm{false}\rangle = 0 = \langle \mathrm{false}|\overrightarrow{0}\rangle,$
  $\langle \overrightarrow{0}|\mathrm{x}\rangle = 0 = \langle \mathrm{x}|\overrightarrow{0}\rangle$
- $\langle (t, t') \mid (u, u')\rangle = \langle t|u\rangle * \langle t'|u'\rangle$
- $\langle \lambda * t + \lambda' * t' \mid u\rangle = \lambda^* * \langle t|u\rangle + \lambda'^* * \langle t'|u\rangle,$
  $\langle t \mid \kappa * u + \kappa' * u'\rangle = \kappa * \langle t|u\rangle + \kappa' * \langle t|u'\rangle$
- $\ldots$
- $\langle t|u\rangle = ? \qquad$ otherwise

Summary

- Our semantic ideas proved useful when designing a quantum programming language, analogous concepts are modelled by the same syntactic constructs

Summary

- Our semantic ideas proved useful when designing a quantum programming language, analogous concepts are modelled by the same syntactic constructs
- Our analysis also highlights the differences between classical and quantum programming

Summary

- Our semantic ideas proved useful when designing a quantum programming language, analogous concepts are modelled by the same syntactic constructs
- Our analysis also highlights the differences between classical and quantum programming
- Quantum programming introduces the problem of *control of decoherence*, which we address by making the 'forgetting' of variables explicit, and by having two **if** constructs and the orthogonality condition

Summary

- Our semantic ideas proved useful when designing a quantum programming language, analogous concepts are modelled by the same syntactic constructs
- Our analysis also highlights the differences between classical and quantum programming
- Quantum programming introduces the problem of *control of decoherence*, which we address by making the ‘forgetting’ of variables explicit, and by having two **if** constructs and the orthogonality condition
- Other work:

Summary

- Our semantic ideas proved useful when designing a quantum programming language, analogous concepts are modelled by the same syntactic constructs
- Our analysis also highlights the differences between classical and quantum programming
- Quantum programming introduces the problem of *control of decoherence*, which we address by making the 'forgetting' of variables explicit, and by having two **if** constructs and the orthogonality condition
- Other work:
- A compiler for QML, in Haskell, that produces (typed) circuits

Summary

- Our semantic ideas proved useful when designing a quantum programming language, analogous concepts are modelled by the same syntactic constructs
- Our analysis also highlights the differences between classical and quantum programming
- Quantum programming introduces the problem of *control of decoherence*, which we address by making the 'forgetting' of variables explicit, and by having two **if** constructs and the orthogonality condition
- Other work:
- A compiler for QML, in Haskell, that produces (typed) circuits
- Extend algebra and normal form beyond pure subset of QML

Summary

- Our semantic ideas proved useful when designing a quantum programming language, analogous concepts are modelled by the same syntactic constructs
- Our analysis also highlights the differences between classical and quantum programming
- Quantum programming introduces the problem of *control of decoherence*, which we address by making the 'forgetting' of variables explicit, and by having two **if** constructs and the orthogonality condition
- Other work:
- A compiler for QML, in Haskell, that produces (typed) circuits
- Extend algebra and normal form beyond pure subset of QML
- Higher-order types, composionality proofs, implement using measurement calculus ...

Thanks for listening

- Papers on QML can be found at:
- sneezy.cs.nott.ac.uk/qml
- There is also an interactive research diary:
- sneezy.cs.nott.ac.uk/qml/weblog

- Jonathan Grattage (www.cs.nott.ac.uk/∼ jjg)
  Thorsten Altenkirch (www.cs.nott.ac.uk/∼ txa)