Functional Quantum Programming

Thorsten Altenkirch
University of Nottingham
based on joint work with Jonathan Grattage
and discussions with V.P. Belavkin
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 $O\left(\sqrt{n}\right)$.

Can we build a quantum computer? yes We can run quantum algorithms. no Nature is classical after all!

Assumption: Nature is fair. . .
Simulation of quantum systems is expensive: PSPACE complexity for polynomial circuits.
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 $O(n/\sqrt{2})$
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.
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!
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

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!
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!

FCC: Finite classical computations
FQC: Finite quantum computations

Important issue: control of decoherence.

Draft paper available (Google: Thorsten, functional, quantum).
Compiler under construction (Jonathan).
QML

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

Quantum control and quantum data.
QML

- Quantum control and quantum data.
- Design guided by denotational semantics
QML

- 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)
Compiler under construction (Jonathan)
QML

- 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: 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: 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)

Compiler under construction (Jonathan)
Example: Hadamard operation
Example: Hadamard operation

Matrix

\[
H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}
\]
Example: Hadamard operation

Matrix

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

QML

$$H x : Q_2 = \text{if}^\circ x \text{ then } \{\text{qfalse} | (-1)\text{qtrue}\} \text{ else } \{\text{qfalse} | \text{qtrue}\}$$
Related Work

- P. Zuliani, 2001, *Quantum Programming*
- S-C. Mu and R. S. Bird, 2001, *Quantum functional programming*
- A. Sabry, 2003, *Modeling quantum computing in Haskell*
- P. Selinger, 2002, *Towards a Quantum Programming Language*
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):

- A finite set of initial heaps $H$
- An initial heap $h \in H$
- A finite set of garbage states $G$
- A bijection $\phi : A \to H \to B \to G$

$\phi$
Classical computations (FCC)

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

- a finite set of initial heaps $H$,

\[
\begin{array}{c}
A \\
\downarrow h \\
H \\
\phi \\
G \\
B
\end{array}
\]
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

Exercise: Define $I$. 

$\phi_\beta \circ \phi_\alpha$
Composing classical computations

Exercise: Define $I$. 
Extensional equality
Extensional equality

Every computation $\alpha$ gives rise to a function $U_{\text{FCC}} \alpha \in A \to B$
Extensional equality

Every computation $\alpha$ gives rise to a function $\text{U}_{\text{FCC}} \alpha \in A \rightarrow B$

$$
\begin{align*}
A \times H & \xrightarrow{\phi} B \times G \\
A & \xrightarrow{\text{U}_{\text{FCC}} \alpha} B
\end{align*}
$$

$\alpha =_{\text{ext}} \beta$, if $\text{U}_{\text{FCC}} \alpha = \text{U}_{\text{FCC}} \beta$
Extensional equality

Every computation $\alpha$ gives rise to a function $U_{\text{FCC}} \alpha \in A \rightarrow B$

$$A \times H \xrightarrow{\phi} B \times G$$

$$\begin{array}{c}
A \downarrow \phi \downarrow (\cdot, h) \downarrow \pi_1 \\
\downarrow U_{\text{FCC}} \alpha \downarrow B
\end{array}$$

$$\alpha =_{\text{ext}} \beta, \text{ if } U_{\text{FCC}} \alpha = U_{\text{FCC}} \beta$$

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

$U_{\text{FCC}} (\beta \circ \alpha) = (U_{\text{FCC}} \beta) \circ (U_{\text{FCC}} \alpha)$
$U_{FCC} I = I$

$U_{FCC} (\beta \circ \alpha) = (U_{FCC} \beta) \circ (U_{FCC} \alpha)$

$U_{FCC}$ is a functor $U_{FCC} : FCC \rightarrow \text{FinSet}$. 
\( U_{\text{FCC}} \)

\[
\begin{align*}
U_{\text{FCC}} I &= I \\
U_{\text{FCC}} (\beta \circ \alpha) &= (U_{\text{FCC}} \beta) \circ (U_{\text{FCC}} \alpha)
\end{align*}
\]

- \( U_{\text{FCC}} \) is a functor \( U_{\text{FCC}} : \text{FCC} \to \text{FinSet} \).
- \( U_{\text{FCC}} \) is faithful (trivially).
$U_{FCC} I = I$

$U_{FCC} (\beta \circ \alpha) = (U_{FCC} \beta) \circ (U_{FCC} \alpha)$

- $U_{FCC}$ is a functor $U_{FCC} : FCC \to \text{FinSet}$.
- $U_{FCC}$ is faithful (trivially).
- **Exercise:** $U_{FCC}$ is full!
Coming next: Quantum computations

Develop FQC analogously to FCC...
Given a finite set \( A \) (the base), \( C^A \) is a Hilbert space.

Linear operators: \( f : A \to B \to C \) induces \( ^f : C^A \to C^B \).

We write \( f : A(B) \).

**Norm of a vector:** \( \|v\| = a^2 \in A(v(a^2)) \).

Unitary operators: A unitary operator is a linear isomorphism that preserves the norm.
Linear algebra revision

Given a finite set $A$ (the base) $\mathbb{C}A = A \rightarrow \mathbb{C}$ is a **Hilbert space**.
Linear algebra revision

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

**Linear operators:**

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

We write $f \in A \rightsquigarrow B$.
Linear algebra revision

Given a finite set $A$ (the base)

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

**Linear operators:**

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

we write $f \in A \rightarrow B$

**Norm of a vector:**

$\|v\| = \sum_{a \in A} (va)^* (va) \in \mathbb{R}^+$,
Given a finite set $A$ (the base)
$\mathbb{C}A = A \rightarrow \mathbb{C}$ is a **Hilbert space**.

**Linear operators:**

$f \in A \rightarrow B \rightarrow \mathbb{C}$ induces $\hat{f} \in \mathbb{C}A \rightarrow \mathbb{C}B$.
we write $f \in A \rightarrow B$

**Norm of a vector:**

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

**Unitary operators:**

A unitary operator $\phi \in A \rightarrow_{\text{unitary}} B$ is a linear isomorphism that preserves the norm.
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 $2^A$ (unitary $B$).
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 $\nu \in \mathbb{C} A$ with unit norm $\|\nu\| = 1$.

- A **reversible computation** is given by a unitary operator $\phi \in A \xrightarrow{\text{unitary}} B$. 
Quantum computations (FQC)

Given finite sets $A$ (input) and $B$ (output), the base of the space of initial heaps, a heap initialisation vector $h \in H \subset C$, a finite set $G$ (output), the base of the space of garbage states, a unitary operator $2^A \otimes 2^B$ (unitary $B \otimes G$).
Quantum computations (FQC)

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

\[
\begin{array}{c}
A & \phi & B \\
\hline
h & H & G
\end{array}
\]
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 $h \in \mathbb{C}H$, 

\[ \begin{array}{c}
A \\
\phi \\
H \\
B \\
\end{array} \]

\[ h \]

\[ G \]
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 \( \phi \in A \otimes H \rightarrow_{\text{unitary}} B \otimes G \).
Composing quantum computations
Composing quantum computations

\[ \phi_{\alpha} \circ \phi_{\beta} \]

A \quad B \quad C

\[ H_{\alpha} \quad \phi_{\alpha} \quad \phi_{\beta} \quad G_{\alpha} \]

\[ H_{\beta} \quad G_{\beta} \]

\[ \phi_{\beta \circ \alpha} \]
Extensional equality...
Extensional equality... is a bit more subtle.
Extensional equality…

… is a bit more subtle.

There is no sensible operator replacing $\pi_1$ on vector spaces:

$$A \otimes H \xrightarrow{\phi} B \otimes G$$

Indeed:

Forgetting part of a pure state results in a mixed state.
Extensional equality…

…is a bit more subtle.

There is no sensible operator replacing $\pi_1$ 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**

$$\rho \in A \rightarrow A$$

such that all eigenvalues are positive reals

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

and has a unit trace

$$\sum a \in A. \nu a = 1$$
A superoperator $A$ is a linear operator on density operators which is completely positive. A unitary operator $B$ gives rise to a superoperator $y$. Partial trace: $\text{tr}_A ; G = G$.
A superoperator $f \in A \rightarrow_{\text{super}} B$ is a linear operator on density operators which is completely positive.
Superoperators

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

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

- A unitary operator $\phi \in A \rightarrow_{\text{unitary}} B$ gives rise to a superoperator $\phi^\dagger \in A \rightarrow_{\text{super}} B$.

- Partial trace:

  $$\text{tr}_{A,G} \in A \otimes G \rightarrow_{\text{super}} A$$
Extensional equality
Extensional equality

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

\[
\begin{array}{ccc}
A \otimes H & \xrightarrow{\phi} & B \otimes G \\
\downarrow{\tilde{h}} & & \downarrow{\text{tr}_G} \\
A & \xrightarrow{U_{\text{FQC}} \alpha} & B
\end{array}
\]
Extensional equality

Every computation \( \alpha \) gives rise to a superoperator \( U \alpha \in A \overset{\text{super}}{\rightarrow} B \)

\[
\begin{array}{c}
A \otimes H \overset{\phi}{\longrightarrow} B \otimes G \\
\downarrow \approx h \quad \downarrow \text{tr}_G \\
A \overset{\mathcal{U}_{\text{FQC}} \alpha}{\longrightarrow} B
\end{array}
\]

\[\alpha = \text{ext} \beta, \text{ if } \mathcal{U}_{\text{FQC}} \alpha = \mathcal{U}_{\text{FQC}} \beta\]
Extensional equality

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

\[
\begin{array}{ccc}
A \otimes H & \xrightarrow{\phi} & B \otimes G \\
\downarrow -\otimes h & & \downarrow \text{tr}_G \\
A & \xrightarrow{U_{\text{FQC}} \alpha} & B
\end{array}
\]

$\alpha =_{\text{ext}} \beta$, if $U_{\text{FQC}} \alpha = U_{\text{FQC}} \beta$

**Objects** finite sets

**FCC:**

**Morphisms** computations $/ =_{\text{ext}}$
U_{\text{FQC}}(\mathcal{F}) = (U_{\text{FQC}}(\mathcal{F}))^!_{\text{Super}}.

U_{\text{FQC}} is faithful (trivially).

U_{\text{FQC}} is full!
\[ U_{\text{FQC}} I = I \]
\[ U_{\text{FQC}} (\beta \circ \alpha) = (U_{\text{FQC}} \beta) \circ (U_{\text{FQC}} \alpha) \]
$\mathcal{U}_{\text{FQC}} I = I$

$\mathcal{U}_{\text{FQC}} (\beta \circ \alpha) = (\mathcal{U}_{\text{FQC}} \beta) \circ (\mathcal{U}_{\text{FQC}} \alpha)$

$\mathcal{U}_{\text{FQC}}$ is a functor $\mathcal{U}_{\text{FQC}} : \text{FQC} \to \text{Super}$. 
\( U_{\text{FQC}} I = I \)

\( U_{\text{FQC}} (\beta \circ \alpha) = (U_{\text{FQC}} \beta) \circ (U_{\text{FQC}} \alpha) \)

- \( U_{\text{FQC}} \) is a functor \( U_{\text{FQC}} : \text{FQC} \to \text{Super} \).
- \( U_{\text{FQC}} \) is faithful (trivially).
$U_{FQC} I = I$

$U_{FQC} (\beta \circ \alpha) = (U_{FQC} \beta) \circ (U_{FQC} \alpha)$

- $U_{FQC}$ is a functor $U_{FQC} : FQC \rightarrow Super$.
- $U_{FQC}$ is faithful (trivially).
- $U_{FQC}$ is full!
Classical vs quantum
### Classical vs quantum

<table>
<thead>
<tr>
<th>classical</th>
<th>quantum</th>
</tr>
</thead>
</table>

- finite sets
- finite dimensional Hilbert spaces
- bijections
- unitary operators
- cartesian product
- tensor product
- functions
- superoperators
- projections
- partial trace
Classical vs quantum

<table>
<thead>
<tr>
<th>classical</th>
<th>quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>finite sets</td>
<td></td>
</tr>
</tbody>
</table>
### Classical vs Quantum

<table>
<thead>
<tr>
<th>Classical</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>finite sets</td>
<td>finite dimensional Hilbert spaces</td>
</tr>
</tbody>
</table>
### Classical vs Quantum

<table>
<thead>
<tr>
<th>Classical</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>finite sets</td>
<td>finite dimensional Hilbert spaces</td>
</tr>
<tr>
<td>bijections</td>
<td></td>
</tr>
<tr>
<td>classical</td>
<td>quantum</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>finite sets</td>
<td>finite dimensional Hilbert spaces</td>
</tr>
<tr>
<td>bijections</td>
<td>unitary operators</td>
</tr>
</tbody>
</table>
### Classical vs quantum

<table>
<thead>
<tr>
<th>classical</th>
<th>quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>finite sets</td>
<td>finite dimensional Hilbert spaces</td>
</tr>
<tr>
<td>bijections</td>
<td>unitary operators</td>
</tr>
<tr>
<td>cartesian product ($\times$)</td>
<td></td>
</tr>
</tbody>
</table>
# Classical vs quantum

<table>
<thead>
<tr>
<th>classical</th>
<th>quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>finite sets</td>
<td>finite dimensional Hilbert spaces</td>
</tr>
<tr>
<td>bijections</td>
<td>unitary operators</td>
</tr>
<tr>
<td>cartesian product (×)</td>
<td>tensor product (⊗)</td>
</tr>
</tbody>
</table>

- Classical vs quantum
- Finite sets vs finite dimensional Hilbert spaces
- Bijections vs unitary operators
- Cartesian product (×) vs tensor product (⊗)
### Classical vs quantum

<table>
<thead>
<tr>
<th>classical</th>
<th>quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>finite sets</td>
<td>finite dimensional Hilbert spaces</td>
</tr>
<tr>
<td>bijections</td>
<td>unitary operators</td>
</tr>
<tr>
<td>cartesian product ($\times$)</td>
<td>tensor product ($\otimes$)</td>
</tr>
<tr>
<td>functions</td>
<td></td>
</tr>
<tr>
<td>Classical</td>
<td>Quantum</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>finite sets</td>
<td>finite dimensional Hilbert spaces</td>
</tr>
<tr>
<td>bijections</td>
<td>unitary operators</td>
</tr>
<tr>
<td>cartesian product</td>
<td>tensor product</td>
</tr>
<tr>
<td>((\times))</td>
<td>((\otimes))</td>
</tr>
<tr>
<td>functions</td>
<td>superoperators</td>
</tr>
</tbody>
</table>
## Classical vs quantum

<table>
<thead>
<tr>
<th>classical</th>
<th>quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>finite sets</td>
<td>finite dimensional Hilbert spaces</td>
</tr>
<tr>
<td>bijections</td>
<td>unitary operators</td>
</tr>
<tr>
<td>cartesian product (×)</td>
<td>tensor product (⊗)</td>
</tr>
<tr>
<td>functions</td>
<td>superoperators</td>
</tr>
<tr>
<td>projections</td>
<td></td>
</tr>
</tbody>
</table>
### Classical vs quantum

<table>
<thead>
<tr>
<th>classical</th>
<th>quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>finite sets</td>
<td>finite dimensional Hilbert spaces</td>
</tr>
<tr>
<td>bijections</td>
<td>unitary operators</td>
</tr>
<tr>
<td>cartesian product ($\times$)</td>
<td>tensor product ($\otimes$)</td>
</tr>
<tr>
<td>functions</td>
<td>superoperators</td>
</tr>
<tr>
<td>projections</td>
<td>partial trace</td>
</tr>
</tbody>
</table>
Decoherence
Decoherence

Classically $\mathcal{I} = I$

Quantum input: $f_1 p_2 j_0 i_1 + 1 p_2 j_0 i_g$

output: $1 2 f j_0 i_g + 1 2 f j_1 i_g$

Functional Quantum Programming – p. 23/44
Decoherence

Classically

$\pi_1 \circ \delta = I$
Decoherence

Classically

Quantum

\[ \pi_1 \circ \delta = I \]
Decoherence

Classically

Quantum

input: \[ \left\{ \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |0\rangle \right\} \]

\[ \pi_1 \circ \delta = I \]
Decoherence

Classically

\[ \pi_1 \circ \delta = I \]

Quantum

input: \[ \left\{ \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |0\rangle \right\} \]

output: \[ \frac{1}{2} \left\{ |0\rangle \right\} + \frac{1}{2} \left\{ |1\rangle \right\} \]
QML basics
QML basics

\[ \Gamma \vdash t : \sigma \]

\[ [t] \in \text{FQC} [\Gamma] [\tau] \]
QML basics

\[ \Gamma \vdash t : \sigma \quad \Rightarrow \quad [t] \in \text{FQC} \left[ \Gamma \right]\left[ \tau \right] \]

QML is based on strict linear logic
no weakening but contraction.
QML basics

\[ \Gamma \vdash t : \sigma \]

\[ [t] \in \text{FQC} \left[ \Gamma \right] \left[ \tau \right] \]

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

\[ |1| = 0 \]

\[ |\sigma \sqcup \tau| = \max \{ |\sigma|, |\tau| \} \]

\[ |\sigma \oplus \tau| = |\sigma \sqcup \tau| + 1 \]

\[ |\sigma \otimes \tau| = |\sigma| + |\tau| \]
Interpretation of types

\[
\begin{align*}
|1| &= 0 \\
|\sigma \uplus \tau| &= \max \{ |\sigma|, |\tau| \} \\
|\sigma \oplus \tau| &= |\sigma \uplus \tau| + 1 \\
|\sigma \otimes \tau| &= |\sigma| + |\tau| \\
[\sigma] &= 2^{|\sigma|}
\end{align*}
\]
on contexts
on contexts

\[ \Gamma, x : \sigma \otimes \Delta, x : \sigma = (\Gamma \otimes \Delta), x : \sigma \]
\[ \Gamma, x : \sigma \otimes \Delta = (\Gamma \otimes \Delta), x : \sigma \quad \text{if } x \notin \text{dom } \Delta \]
\[ \bullet \otimes \Delta = \Delta \]
on contexts

\[ \Gamma, x : \sigma \otimes \Delta, x : \sigma = (\Gamma \otimes \Delta), x : \sigma \]
\[ \Gamma, x : \sigma \otimes \Delta = (\Gamma \otimes \Delta), x : \sigma \quad \text{if} \ x \not\in \text{dom} \Delta \]
\[ \otimes \Delta = \Delta \]

\[ \phi_{C_{\Gamma,\Delta}} \]

\[ H_{\Gamma,\Delta} \]

\[ \Gamma \otimes \Delta \]

\[ \Gamma \]

\[ \Delta \]
The let-rule
The let-rule

\[
\Gamma \vdash t : \sigma \\
\Delta, x : \sigma \vdash u : \tau \\
\Gamma \otimes \Delta \vdash \text{let } x = t \text{ in } u : \tau
\]
The let-rule

\[
\frac{\Gamma \vdash t : \sigma}{\Delta, x : \sigma \vdash u : \tau} \quad \text{let}
\]

\[
\frac{\Gamma \otimes \Delta \vdash \text{let } x = t \text{ in } u : \tau}{
}

\[
\Gamma \otimes \Delta \\
H_{\Gamma,\Delta} \\
H_t \\
H_u
\]

\[
\phi_{C_{\Gamma,\Delta}} \\
\phi_t \\
\phi_u
\]

\[
B \\
G_t \\
G_u
\]
The var-rule
The var-rule

\[
\Gamma, x : \sigma \vdash x^{\text{dom}\Gamma} : \sigma \quad \text{var}
\]
The var-rule

\[ \Gamma, x : \sigma \vdash x^{\text{dom}\Gamma} : \sigma \]

\[ \Gamma \quad \sigma \]

\[ \sigma \]
Example

\( y : Q_2 \vdash \text{let } x = y \text{ in } x \{ \} : Q_2 \)
Example

\[ y : Q_2 \vdash \text{let } x = y \text{ in } x \{ \} : Q_2 \]

\[ y : Q_2 \vdash \text{let } x = y \text{ in } x \{ y \} : Q_2 \]
\( \frac{\Gamma \vdash t : \sigma \quad \Delta \vdash u : \tau}{\Gamma \otimes \Delta \vdash (t, u) : \sigma \otimes \tau} \quad \otimes \text{intro} \)
\[ \Gamma \vdash t : \sigma \quad \Delta \vdash u : \tau \]
\[ \Gamma \otimes \Delta \vdash (t, u) : \sigma \otimes \tau \]

\( \otimes \text{intro} \)
\( -\text{elim} \)
\(\otimes\)-elim

\[
\begin{align*}
\Gamma \vdash t : \sigma \otimes \tau \\
\Delta, x : \sigma, y : \tau \vdash u : C
\end{align*}
\]

\[
\frac{}{\Gamma \otimes \Delta \vdash \text{let} \ (x, y) = t \ \text{in} \ u : C} \otimes \text{elim}
\]
\(\bigotimes\)-elim

\[
\Gamma \vdash t : \sigma \otimes \tau \\
\Delta, x : \sigma, y : \tau \vdash u : C \\
\Gamma \otimes \Delta \vdash \text{let } (x, y) = t \text{ in } u : C
\]

\(\bigotimes\) elim
Example

\[ p : Q_2 \otimes Q_2 \vdash \text{let} \ (x, y) = p \text{ in } (y, x) : Q_2 \otimes Q_2 \]
Example

\[ p : Q_2 \otimes Q_2 \vdash \text{let } (x, y) = p \text{ in } (y^{}, x^{}) : Q_2 \otimes Q_2 \]

\[ p : Q_2 \otimes Q_2 \vdash \text{let } (x, y) = p \text{ in } (y^p, x^p) : Q_2 \otimes Q_2 \]
$$\Gamma \vdash t : A$$

$$\Gamma \vdash \text{inl } t : A \oplus B$$
\[
\Gamma \vdash t : A \\
\Gamma \vdash \text{inl} t : A \oplus B
\]
-elim
\(\bigcirc\text{-elim}\)

\[
\begin{align*}
\Gamma \vdash c : \sigma \bigcirc \tau \\
\Delta, x : \sigma \vdash t : \rho \\
\Delta, y : \tau \vdash u : \rho \\
\hline
\Gamma \otimes \Delta \vdash \text{case } c \text{ of } \{\text{inl } x \Rightarrow t \mid \text{inr } y \Rightarrow u\} : \rho
\end{align*}
\]
$\oplus$-elim

$\Gamma \vdash c : \sigma \oplus \tau$

$\Delta, x : \sigma \vdash t : \rho$

$\Delta, y : \tau \vdash u : \rho$

$\Gamma \otimes \Delta \vdash \text{case } c \text{ of } \{ \text{inl } x \Rightarrow t | \text{inr } y \Rightarrow u \} : \rho$

$\Gamma \otimes \Delta \rightarrow C_{\Gamma,\Delta}$

$H_{\Gamma,\Delta}$

$H_b$

$H_{t-u}$

$\phi_b$

$\phi_{[t|u]}$

$\Delta$

$\sigma \sqcup \tau$

$Q_2$

$G$

$G_b$

$\rho$
-elim decoherence-free
⊕-elim decoherence-free

\[ \Gamma \vdash c : \sigma \oplus \tau \]
\[ \Delta, \ x : \sigma \vdash t : \rho \]
\[ \Delta, \ y : \tau \vdash u : \rho, \ t \perp u \]
\[ \Gamma \otimes \Delta \vdash \text{case}^\circ \ b \ \text{of} \ \{ \text{inl} \ x \Rightarrow t | \text{inr} \ y \Rightarrow u \} : \rho \]

\[ + \text{elim}^\circ \]
-elim decoherence-free

\[ \Gamma \vdash c : \sigma \oplus \tau \]
\[ \Delta, x : \sigma \vdash t : \rho \]
\[ \Delta, y : \tau \vdash u : \rho, \quad t \perp u \]

\[ \Gamma \otimes \Delta \vdash \text{case}^\circ \; b \; \text{of} \; \{ \text{inl} \; x \Rightarrow t | \text{inr} \; y \Rightarrow u \} : \rho \quad + \text{elim}^\circ \]

\[ \phi_{C_{\Gamma,\Delta}} \]
\[ \phi_{b} \]
\[ \phi_{[f|g]} \]
\[ \phi_{\bot} \]
\[ H_{\Gamma,\Delta} \]
\[ H_{b} \]
\[ H_{t-u} \]
\[ G_{b} \]
\[ G \]
Orthogonality

\[ \text{inl } t \perp \text{inr } u \quad \text{inl } t \perp \text{inl } u \quad \text{inr } t \perp \text{inr } u \]

\[ (t, v) \perp (u, w) \quad (v, t) \perp (w, u) \]
Semantics of \( \bot \)

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

- \( S \) finite set.
- \( \phi \in Q_2 \otimes S \xrightarrow{\text{unitary}} [\sigma] \)
- \( f \in \text{FQC} [\Gamma] S \)
- \( g \in \text{FQC} [\Gamma] S \)

\[
[t] = \phi \circ (\text{true} \otimes -) \circ f,
[u] = \phi \circ (\text{false} \otimes -) \circ g
\]
Superpositions

\[ \Gamma \vdash t, u : \sigma \quad t \perp u \]
\[ ||\lambda||^2 + ||\lambda'||^2 = 1 \quad \lambda, \lambda' \neq 0 \]

\[ \Gamma \vdash \{ (\lambda)t | (\lambda')u \} : \sigma \]
\[ \equiv \text{if}^\circ \{ (\lambda)\text{qtrue} | (\lambda')\text{qfalse} \} \text{ then } t \text{ else } u \]
Example: Deutsch’s algorithm

\[
\text{Eq } a : Q_2, b : Q_2 = \text{let } (x, y) = \text{if } \{ \text{qfalse } | (-1)\text{qtrue}\} \\
\text{then } (\text{qtrue, if } a \\
\text{then } \{ \text{qfalse } | (-1)\text{qtrue}\} \\
\text{else } \{ \text{qfalse } | \text{qtrue}\} ) \\
\text{else } \{ \text{qfalse, if } b \\
\text{then } \{ \text{qfalse } | (-1)\text{qtrue}\} \\
\text{else } \{ \text{qfalse } | \text{qtrue}\} \\
\text{in } x \\
: Q_2
\]
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
- Infinite or indexed?