Quantum Effects

Juliana K. Vizzotto[1] **Thorsten Altenkirch**[2]

Amr Sabry[3]

[1] Federal University of Rio Grande do Sul

[2] The University of Nottingham

[3] Indiana University

20 January 2005

Quantum mechanics

Real Black Magic Calculus — Albert Einstein

If quantum mechanics hasn't profoundly shocked you, you haven't understood it yet — Niels Bohr

I think I can safely say that nobody today understands quantum mechanics — Richard Feynman

I can't possibly know what I am talking about — Amr Sabry

Models of (quantum) computation

Abstract (Compositional) — Values and functions

Circuits — Vectors and matrices

Physics — Particle spins and electromagnetic fields

Abstract models of (quantum) computation

Semantic foundation for functional quantum programming language:

- Category theory — categorical models of quantum computation [Abramsky, Selinger, van Tonder]
- λ-calculus — quantum λ-calculus [Van Tonder]
- Domain theory, logic, etc — [Birkhoff, von Neumann]
- Haskell (not perfect but rich executable language)

Plan

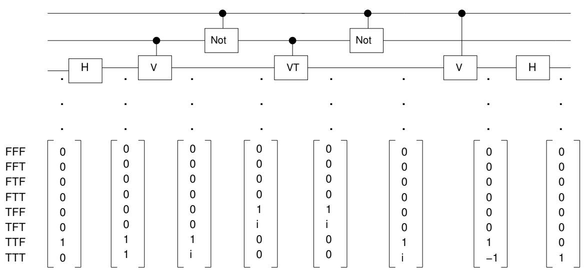
- Quantum computation (I): unitary operations on state vectors
- Embedding in Haskell using monads
- What to do with measurement?

- Quantum computation (II): superoperators on density matrices
- Arrows
- Embedding in Haskell using arrows

- QML [Altenkirch and Grattage]
- Open problems; related work; conclusions

Quantum Computing (I)

Example: Toffoli circuit



- In this example, input is $|TTF\rangle$
- After first step, state vector is a superposition $|TTF\rangle + |TTT\rangle$
- Result: Negate the last bit

Entanglement

- The state vector of multiple qubits can sometimes be teased into the product of simpler state vectors:

$$|FF\rangle + |FT\rangle = |F\rangle * (|F\rangle + |T\rangle)$$

- If the qubits are entangled, this is impossible:

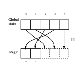
$$|FF\rangle + |TT\rangle \neq (|F\rangle + |T\rangle) * (|F\rangle + |T\rangle)$$

(or any other product we might try)

- Must basically manipulate the global state at all times *even if we want to apply an operation to only one qubit*

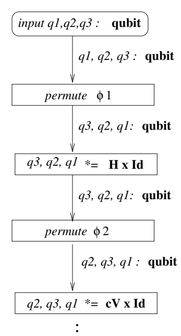
QCL [Knill]

- A global state with n qubits
- Registers are realized using pointers to the global state
- Apply operation U to register r using $\Pi^\dagger.(U \times I_{(n-m)}).\Pi$ where $\Pi^\dagger$ is the inverse of Π and I is the identity



Flowchart notation [Selinger]

- A global state with n variables that can be assigned once
- To apply operation U to part of the state, use the same idea that is used in QCL:
  - re-order the variables to bring relevant variables to the front
  - compose U with the identity and apply it to the entire state



Lambda-calculus extension [Valiron and Selinger]

Idea: the lambda term gives classical control over the quantum data which is accessed via pointers to a global data structure.

- The state is a triple $[Q, Q_f, M]$
- Q is the state vector
- M is a lambda term with free variables
- Q_f is a linking function which maps every free variable of M to a qubit in Q

Virtual values and adaptors [Sabry]

- Also uses a global state and pointers mediated using adaptors
- Hides the management of pointers using virtual values
- Adaptors can almost be derived from the types but their actual generation is tedious and ugly

$$
\begin{array}{l}
\mathit{toffoli\ state}\ = \\
\quad \textbf{let}\ b \quad = \mathit{virt\ state\ adaptor}_0 \\
\quad\quad\ \ mb = \mathit{virt\ state\ adaptor}_1 \\
\quad\quad\ \ tm = \mathit{virt\ state\ adaptor}_2 \\
\quad\quad\ \ tb \ \ = \mathit{virt\ state\ adaptor}_3 \\
\quad \textbf{in do}\ \mathit{app\ hadamard\ b} \\
\quad\quad\quad\ \ \mathit{app\ cv\ mb} \\
\quad\quad\quad\ \ \mathit{app\ cnot\ tm} \\
\quad\quad\quad\ \ \mathit{app\ cvt\ mb} \\
\quad\quad\quad\ \ \mathit{app\ cnot\ tm} \\
\quad\quad\quad\ \ \mathit{app\ cv\ tb} \\
\quad\quad\quad\ \ \mathit{app\ hadamard\ b}
\end{array}
$$

Can we do better than pointers to a global state?

Common theme so far:

- A global state vector accessed via pointers
- Each operation transforms the global state to a new state
- Monads are often used to structure and reason about computational effects.

$$
\begin{array}{l}
\textbf{class}\ \mathit{Monad}\ m\ \textbf{where} \\
\quad \mathit{return} :: \forall\, a.\quad a \rightarrow m\ a \\
\quad (\gg\!=)\ \ :: \forall\, a\ b.\ m\ a \rightarrow (a \rightarrow m\ b) \rightarrow m\ b
\end{array}
$$

- Is there a nice monad here?

Embedding in Haskell using monads

Finite sets

- We only consider computations over finite bases. For a type a to be a type of observables, it needs to represent a finite set:

$$\begin{array}{l} \textbf{class}\ Eq\ a \Rightarrow Basis\ a\ \textbf{where}\ basis :: [a] \\ \textbf{instance}\ Basis\ Bool\ \ \textbf{where}\ basis\ = [False, True] \end{array}$$

- Can automatically construct more complicated sets (on demand):

$$\begin{array}{l} \textbf{instance}\ (Basis\ a,\ Basis\ b) \Rightarrow Basis(a,\ b)\ \textbf{where} \\ \quad basis = [(a,\ b) \mid a \leftarrow basis,\ b \leftarrow basis\ ] \end{array}$$

- Programs at the end produce classical observable values: $False$, $(True, False)$, etc

State vectors

- A state vector associates a complex probability amplitude with each basis element:

$$\begin{array}{lcl} \textbf{type}\ \mathit{PA} & = & \mathit{Complex\ Double} \\ \textbf{type}\ \mathit{Vec}\ a & = & a \rightarrow \mathit{PA} \end{array}$$

- Can add, substract, and multiply vectors: (definitions omitted)

$$\begin{array}{lcl} \mathit{vzero} & :: & \mathit{Vec}\ a \\ (\langle + \rangle) & :: & \mathit{Vec}\ a \rightarrow \mathit{Vec}\ a \rightarrow \mathit{Vec}\ a \\ (\langle - \rangle) & :: & \mathit{Vec}\ a \rightarrow \mathit{Vec}\ a \rightarrow \mathit{Vec}\ a \\ (\$*) & :: & \mathit{PA} \rightarrow \mathit{Vec}\ a \rightarrow \mathit{Vec}\ a \\ (\langle * \rangle) & :: & \mathit{Vec}\ a \rightarrow \mathit{Vec}\ b \rightarrow \mathit{Vec}\ (a, b) \\ (\langle \cdot \rangle) & :: & \mathit{Basis}\ a \Rightarrow \mathit{Vec}\ a \rightarrow \mathit{Vec}\ a \rightarrow \mathit{PA} \end{array}$$

Examples of vectors over *Bool*

- The simplest vector is a unit representing a basis element:

$$\begin{array}{lcl} \mathit{unit} & :: & \mathit{Basis}\ a \Rightarrow a \rightarrow \mathit{Vec}\ a \\ \mathit{unit}\ a & = & (\backslash\ b \rightarrow \mathbf{if}\ a == b\ \mathbf{then}\ 1\ \mathbf{else}\ 0) \end{array}$$

- The two basic unit vectors:

$$\begin{array}{lcll} \mathit{qFalse} & = & \mathit{unit\ False} & \text{— in Dirac notation } |0\rangle \\ \mathit{qTrue} & = & \mathit{unit\ True} & \text{— in Dirac notation } |1\rangle \end{array}$$

- Vector representing superpositions:

$$\begin{array}{lcll} \mathit{qFT} & = & (1\ /\ \mathit{sqrt}\ 2)\ \$*\ (\mathit{qFalse}\ \langle+\rangle\ \mathit{qTrue}) & \text{—}\ \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ \mathit{qFmT} & = & (1\ /\ \mathit{sqrt}\ 2)\ \$*\ (\mathit{qFalse}\ \langle-\rangle\ \mathit{qTrue}) & \text{—}\ \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{array}$$

Examples of vectors over $(Bool, Bool)$

- Using the tensor product:

$$p1 \;=\; qFalse \;\langle * \rangle\; qFT \quad \text{— in Dirac notation } |0\rangle * (\tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle))$$

- Using the classical product over the basis:

$$\begin{aligned} qFF &= unit\,(False, False) && \text{— in Dirac notation } |00\rangle \\ qTT &= unit\,(True, True) && \text{— in Dirac notation } |11\rangle \end{aligned}$$

- A vector representing the EPR pair $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$:

$$epr \;=\; (1 \;/\; sqrt\; 2)\; \$*\; (qFF \;\langle + \rangle\; qTT)$$

Linear operators

- Given a function $f :: a \rightarrow Vec\ b$, we can produce the linear operator of type $Vec\ a \rightarrow Vec\ b$
- Apply f to each basis element and accumulate the results:

$$
\begin{array}{lll}
linop & :: & Basis\ a \Rightarrow (a \rightarrow Vec\ b) \rightarrow (Vec\ a \rightarrow Vec\ b) \\
linop\ f\ va & = & (\backslash\ b \rightarrow sum\,[\,(va\ a) * (f\ a\ b) \mid a \leftarrow basis])
\end{array}
$$

- So we can define:

$$
\textbf{type}\ Lin\ a\ b = a \rightarrow Vec\ b
$$

Examples of linear operators

- Construct a linear operator from any pure function:

$$\begin{array}{lcl} \mathit{fun2lin} & :: & (\mathit{Basis}\ a,\ \mathit{Basis}\ b) \Rightarrow (a \rightarrow b) \rightarrow \mathit{Lin}\ a\ b \\ \mathit{fun2lin}\ f\ a & = & \mathit{unit}\ (f\ a) \end{array}$$

- Common linear operators on booleans:

$$\begin{array}{lcl} \mathit{qnot} & = & \mathit{fun2lin}\ \mathit{not} \\ & & \\ \mathit{hadamard}\ \mathit{False} & = & \mathit{qFT} \\ \mathit{hadamard}\ \mathit{True} & = & \mathit{qFmT} \end{array}$$

More linear operations

- Outer product:

$$
\begin{array}{lcl}
(\rangle{*}\langle) & :: & Basis\ a \Rightarrow Vec\ a \rightarrow Vec\ a \rightarrow Lin\ a\ a \\
(v_1 \rangle{*}\langle v_2)\ a_1\ a_2 & = & v_1\ a_1 * conjugate\ (v_2\ a_2)
\end{array}
$$

- Composition:

$$
\begin{array}{lcl}
o & :: & (Basis\ a,\ Basis\ b,\ Basis\ c) \Rightarrow \\
 & & Lin\ a\ b \rightarrow Lin\ b\ c \rightarrow Lin\ a\ c \\
o\ f\ g\ a & = & linop\ g\ (f\ a)
\end{array}
$$

- Controlled-operations:

$$
\begin{array}{lcl}
controlled & :: & Basis\ a \Rightarrow \\
 & & Lin\ a\ a \rightarrow Lin\ (Bool, a)\ (Bool, a) \\
controlled\ f\ (b_1, b_2) & = & (unit\ b_1)\ \langle * \rangle\ (\mathbf{if}\ b_1\ \mathbf{then}\ f\ b_2\ \mathbf{else}\ unit\ b_2)
\end{array}
$$

Almost a monad!

- We can define:

$$
\begin{array}{lcl}
\mathit{return} & :: & \mathit{Basis}\ a \Rightarrow a \rightarrow \mathit{Vec}\ a \\
\mathit{return} & = & \mathit{unit} \\
\\
(\gg\!=) & :: & \mathit{Basis}\ a \Rightarrow \mathit{Vec}\ a \rightarrow (a \rightarrow \mathit{Vec}\ b) \rightarrow \mathit{Vec}\ b \\
(\mathit{va} \gg\!= f) & = & \mathit{linop}\ f\ \mathit{va}
\end{array}
$$

- The right equations are satisfied
- The types are wrong: the extra constraints mean that the construction is not universal. In Haskell terms, we cannot use the **do**-notation
- Already observed [Mu and Bird, 2001] but in a system restricted to manipulating lists of qubits

Toffoli circuit

$$
\begin{array}{l}
\mathit{toffoli} :: \mathit{Lin}\;(\mathit{Bool}, \mathit{Bool}, \mathit{Bool})\;(\mathit{Bool}, \mathit{Bool}, \mathit{Bool}) \\
\mathit{toffoli}\;(\mathit{top}, \mathit{middle}, \mathit{bottom}) \;= \\
\quad \mathbf{let}\; \mathit{cnot} \;=\; \mathit{controlled}\; \mathit{qnot} \\
\qquad\;\; \mathit{cphase} \;=\; \mathit{controlled}\; \mathit{phase} \\
\quad \mathbf{in}\; \mathit{hadamard}\; \mathit{bottom} \ggg\!= \backslash\, b_1 \rightarrow \\
\qquad \mathit{cphase}\;(\mathit{middle}, b_1) \ggg\!= \backslash\, (m_1, b_2) \rightarrow \\
\qquad \mathit{cnot}\;(\mathit{top}, m_1) \ggg\!= \backslash\, (t_1, m_2) \rightarrow \\
\qquad \mathit{controlled}\;(\mathit{adjoint}\; \mathit{phase})\;(m_2, b_2) \ggg\!= \backslash\, (m_3, b_3) \rightarrow \\
\qquad \mathit{cnot}\;(t_1, m_3) \ggg\!= \backslash\, (t_2, m_4) \rightarrow \\
\qquad \mathit{cphase}\;(t_2, b_3) \ggg\!= \backslash\, (t_3, b_4) \rightarrow \\
\qquad \mathit{hadamard}\; b_4 \ggg\!= \backslash\, b_5 \rightarrow \\
\qquad \mathit{return}\;(t_3, m_4, b_5)
\end{array}
$$

So far

- (Almost) monads can be used to structure quantum parallelism: no explicit global state and pointers
- Connections to category theory, etc
- That's the easy part . . . How do we deal with measurement?

Measurement

Measurement & collapse

Measuring qFT $(= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))$:

- returns 0 with probability 1/2
  and *as a side-effect* collapses qFT to $|0\rangle$, or

- returns 1 with probability 1/2
  and *as a side-effect* collapses qFT to $|1\rangle$

Measurement & spooky action at a distance

Measuring the left qubit of epr: $(= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle))$:

- returns 0 with probability 1/2
  and *as a side-effect* collapses epr to $|00\rangle$, or

- returns 1 with probability 1/2
  and *as a side-effect* collapses epr to $|11\rangle$

- The right qubit is affected even if physically distant!

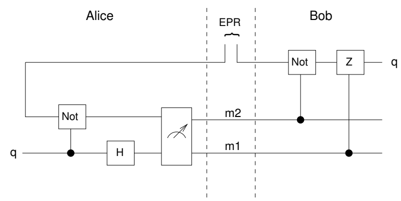
Ignore measurement?

- Measurements can always be delayed to the end;
  many formalisms ignore them

- Mu and Bird use the IO monad to explain measurement;
  cannot mix measurement with linear operations

- Can we deal with measurements in the formalism?

Teleportation



Communication uses a classical channel, sending classical bits.

Quantum Computing (II)

State vectors have too much information

- Perhaps vectors are not expressive enough ?
- Vector is exact state of the system but much of the information in the state is not observable
- Take qFT and measure it. The result is either:

$$\frac{1}{\sqrt{2}}|0\rangle \qquad \text{or} \qquad \frac{1}{\sqrt{2}}|1\rangle$$

- Apply Hadamard to the result:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad \text{or} \qquad \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

- The two configurations are indistinguishable (observationally equivalent)

Density matrices

- Statistical perspective of the state vector
- Technically, we use the outer product:

$$\textbf{type}\ \mathit{Dens}\ a\ =\ \mathit{Vec}\ (a, a)$$

$$\begin{aligned} &\mathit{pureD} :: \mathit{Basis}\ a \Rightarrow \mathit{Vec}\ a \rightarrow \mathit{Dens}\ a \\ &\mathit{pureD}\ v = \mathit{lin2vec}\ (v\ \rangle{*}\langle\ v) \end{aligned}$$

- Examples:

$qFalse$	$qTrue$	qFT
$\lvert 0\rangle$	$\lvert 1\rangle$	$\frac{1}{\sqrt{2}}(\lvert 0\rangle + \lvert 1\rangle)$
$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$

Density matrices and measurement

- When we measure $qFT\ (\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))$ we get:
  False with probability $1/2$, or *True* with probability $1/2$:

$$\begin{pmatrix} 1/2 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1/2 \end{pmatrix} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$

- The density matrix can represent a mixed state

- Operations are linear:

$$H \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} = H \begin{pmatrix} 1/2 & 0 \\ 0 & 0 \end{pmatrix} + H \begin{pmatrix} 0 & 0 \\ 0 & 1/2 \end{pmatrix} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$

- The two states are indeed observationally equivalent.

Superoperators

- Every linear operator can be lifted to an operator on density matrices

- Such operators are called superoperators:

$$
\begin{array}{l}
\textbf{type}\ \mathit{Super}\ a\ b\ =\ (a, a) \rightarrow \mathit{Dens}\ b \\
\\
\mathit{lin2super} :: (\mathit{Basis}\ a,\ \mathit{Basis}\ b) \Rightarrow \mathit{Lin}\ a\ b \rightarrow \mathit{Super}\ a\ b \\
\mathit{lin2super}\ f\ (a_1, a_2)\ =\ (f\ a_1)\ \langle * \rangle\ (\mathit{dual}\ (\mathit{adjoint}\ f)\ a_2) \\
\quad \textbf{where}\ \mathit{dual}\ f\ a\ b\ =\ f\ b\ a
\end{array}
$$

Tracing and measurement

- trL measures and "forgets" the result of measurement
  $meas$ measures and returns the result of measurement

$$trL :: (Basis\ a,\ Basis\ b) \Rightarrow Super\ (a, b)\ b$$
$$trL\ ((a_1, b_1), (a_2, b_2)) = \textbf{if}\ a_1 == a_2\ \textbf{then}\ return\ (b_1, b_2)\ \textbf{else}\ vzero$$

$$meas :: Basis\ a \Rightarrow Super\ a\ (a, a)$$
$$meas\ (a_1, a_2) = \textbf{if}\ a_1 == a_2\ \textbf{then}\ return\ ((a_1, a_1), (a_1, a_1))\ \textbf{else}\ vzero$$

- Measuring qFT and forgetting the collapsed quantum state:

$$\boxed{pureD\ qFT \gg= meas \gg= trL}$$

evaluates to:

$$\begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$

No longer a monad

- At least we can't prove it is a monad

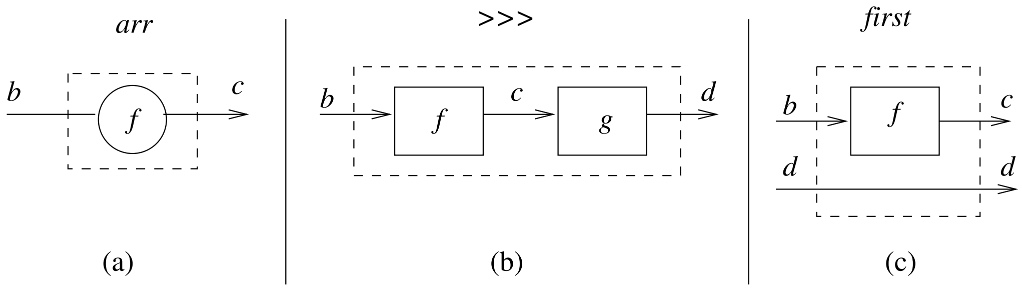
- Superoperators do not form a basis

- We seem to have lost all our structure

Arrows

A generalization of monads

$$
\begin{array}{l}
\textbf{class } \mathit{Arrow}\ a\ \textbf{where} \\
\quad \mathit{arr} \quad :: (b \to c) \to a\ b\ c \\
\quad (\ggg) :: a\ b\ c \to a\ c\ d \to a\ b\ d \\
\quad \mathit{first}\ :: a\ b\ c \to a\ (b, d)\ (c, d)
\end{array}
$$



More about arrows

Look up excellent work at Yale

Embedding in Haskell using arrows

Superoperators are arrows

Well ... almost: the types have additional constraints

$$
\begin{aligned}
&arr :: (Basis\ b,\ Basis\ c) \Rightarrow (b \rightarrow c) \rightarrow Super\ b\ c \\
&arr\ f = fun2lin\ (\backslash\ (b_1, b_2) \rightarrow (f\ b_1,\ f\ b_2)) \\
\\
&(\ggg) :: (Basis\ b,\ Basis\ c,\ Basis\ d) \Rightarrow \\
&\qquad Super\ b\ c \rightarrow Super\ c\ d \rightarrow Super\ b\ d \\
&(\ggg) = o \\
\\
&first :: (Basis\ b,\ Basis\ c,\ Basis\ d) \Rightarrow Super\ b\ c \rightarrow Super\ (b, d)\ (c, d) \\
&first\ f\ ((b_1, d_1), (b_2, d_2)) = permute\ ((f\ (b_1, b_2))\ \langle * \rangle\ (return\ (d_1, d_2))) \\
&\quad \mathbf{where}\ permute\ v\ ((b_1, b_2), (d_2, d_2)) = v\ ((b_1, d_1), (b_2, d_2))
\end{aligned}
$$

Superoperators as a model of quantum computing

- The category of superoperators is known to be an adequate model of quantum computation [Selinger]
- This work suggests that this category corresponds to a functional language with arrows
- Can we accurately express quantum computation in a functional language with arrows?

Toffoli

$$
\begin{array}{l}
\mathit{toffoli} :: \mathit{Super}\ (\mathit{Bool}, \mathit{Bool}, \mathit{Bool})\ (\mathit{Bool}, \mathit{Bool}, \mathit{Bool}) \\
\mathit{toffoli} = \textbf{let}\ \begin{array}[t]{lcl}
\mathit{hadS} & = & \mathit{lin2super}\ \mathit{hadamard} \\
\mathit{cnotS} & = & \mathit{lin2super}\ (\mathit{controlled}\ \mathit{qnot}) \\
\mathit{cphaseS} & = & \mathit{lin2super}\ (\mathit{controlled}\ \mathit{phase}) \\
\mathit{caphaseS} & = & \mathit{lin2super}\ (\mathit{controlled}\ (\mathit{adjoint}\ \mathit{phase}))
\end{array} \\
\qquad\quad \textbf{in proc}\ (a_0, b_0, c_0) \rightarrow \textbf{do} \\
\qquad\qquad\quad c_1 \leftarrow \mathit{hadS} \prec c_0 \\
\qquad\qquad\quad (b_1, c_2) \leftarrow \mathit{cphaseS} \prec (b_0, c_1) \\
\qquad\qquad\quad (a_1, b_2) \leftarrow \mathit{cnotS} \prec (a_0, b_1) \\
\qquad\qquad\quad (b_3, c_3) \leftarrow \mathit{caphaseS} \prec (b_2, c_2) \\
\qquad\qquad\quad (a_2, b_4) \leftarrow \mathit{cnotS} \prec (a_1, b_3) \\
\qquad\qquad\quad (a_3, c_4) \leftarrow \mathit{cphaseS} \prec (a_2, c_3) \\
\qquad\qquad\quad c_5 \leftarrow \mathit{hadS} \prec c_4 \\
\qquad\qquad\quad \mathit{returnA} \prec (a_3, b_4, c_5)
\end{array}
$$

Teleportation (I)

- Can write, type, reason about each component separately.
- Can incorporate measurement in the computation
- Main:

$$
\begin{array}{l}
\mathit{teleport} :: \mathit{Super}\ (\mathit{Bool}, \mathit{Bool}, \mathit{Bool})\ \mathit{Bool} \\
\mathit{teleport}\ =\ \textbf{proc}\ (\mathit{eprL}, \mathit{eprR}, q)\ \rightarrow\ \textbf{do} \\
\qquad\qquad\qquad\quad (m_1, m_2)\ \leftarrow\ \mathit{alice}\ \prec\ (\mathit{eprL}, q) \\
\qquad\qquad\qquad\quad q'\ \leftarrow\ \mathit{bob}\ \prec\ (\mathit{eprR}, m_1, m_2) \\
\qquad\qquad\qquad\quad \mathit{returnA}\ \prec\ q'
\end{array}
$$

Teleportation (II)

$$
\begin{array}{l}
\mathit{alice} :: \mathit{Super}\,(\mathit{Bool}, \mathit{Bool})\,(\mathit{Bool}, \mathit{Bool}) \\
\mathit{alice} \;=\; \textbf{proc}\,(\mathit{eprL}, q) \;\rightarrow\; \textbf{do} \\
\qquad\qquad (q_1, e_1) \;\leftarrow\; \mathit{lin2super}\,(\mathit{controlled}\ \mathit{qnot}) \;\prec\; (q, \mathit{eprL}) \\
\qquad\qquad q_2 \;\leftarrow\; \mathit{lin2super}\ \mathit{hadamard} \;\prec\; q_1 \\
\qquad\qquad ((q_3, e_2), (m_1, m_2)) \;\leftarrow\; \mathit{meas} \;\prec\; (q_2, e_1) \\
\qquad\qquad (m_1', m_2') \;\leftarrow\; \mathit{trL}\,((q_3, e_2), (m_1, m_2)) \\
\qquad\qquad \mathit{returnA} \;\prec\; (m_1', m_2')
\end{array}
$$

$$
\begin{array}{l}
\mathit{bob} :: \mathit{Super}\,(\mathit{Bool}, \mathit{Bool}, \mathit{Bool})\,\mathit{Bool} \\
\mathit{bob} \;=\; \textbf{proc}\,(\mathit{eprR}, m_1, m_2) \;\rightarrow\; \textbf{do} \\
\qquad\qquad (m_2', e_1) \;\leftarrow\; \mathit{lin2super}\,(\mathit{controlled}\ \mathit{qnot}) \;\prec\; (m_2, \mathit{eprR}) \\
\qquad\qquad (m_1', e_2) \;\leftarrow\; \mathit{lin2super}\,(\mathit{controlled}\ z) \;\prec\; (m_1, e_1) \\
\qquad\qquad q' \;\leftarrow\; \mathit{trL} \;\prec\; ((m_1', m_2'), e_2) \\
\qquad\qquad \mathit{returnA} \;\prec\; q'
\end{array}
$$

QML

Why Haskell is not adequate

- There is more to quantum computation than a functional language with arrows
- Cloning?

$$\boxed{\begin{array}{l} \delta :: \mathit{Super\ Bool}\ (\mathit{Bool},\ \mathit{Bool}) \\ \delta\ =\ \mathit{arr}\ (\backslash\ x\ \rightarrow\ (x, x)) \end{array}}$$

- Weakening

$$\boxed{\begin{array}{l} \mathit{weaken} :: \mathit{Super}\ (\mathit{Bool}, \mathit{Bool})\ \mathit{Bool} \\ \mathit{weaken}\ =\ \mathit{arr}\ (\backslash\ (x, y)\ \rightarrow\ y) \end{array}}$$

Cloning

- Well-known “non-cloning” property of quantum states!

$$\begin{array}{l} \delta :: \mathit{Super}\ \mathit{Bool}\ (\mathit{Bool},\ \mathit{Bool}) \\ \delta\ =\ \mathit{arr}\ (\backslash\ x\ \rightarrow\ (x, x)) \end{array}$$

- δ only clones classical information encoded in quantum data
- Applying δ to qFalse will give $\mathit{qFalse}\ \langle * \rangle\ \mathit{qFalse}$
- But applying δ to qFT does not produce $qFT\ \langle * \rangle\ qFT$; rather it produces epr
- One can think of it as cloning a pointer; and sharing the quantum data.

Weakening

- The definition *weaken* allows us to drop some values

$$\begin{array}{l} weaken :: Super\ (Bool, Bool)\ Bool \\ weaken \ = \ arr\ (\backslash\ (x, y) \ \rightarrow\ y) \end{array}$$

- Applying *weaken* to *epr* gives qFT
- But dropping a value amounts to measuring it, and if we measure the left qubit of *epr*, we should be getting either *qFalse* or *qTrue* or the mixed state of both measurements, but never qFT.
- Must prevent weakening

QML [Altenkirch and Grattage]

- A functional language to model quantum computation
- Prevent weakening using strict linear logic
- Two semantics: translation to quantum circuits and translation to superoperators
- Source language models irreversible computations; semantics (compiler) takes care of making everything reversible (by adding a heap input and a garbage output)

QML type system

- Must keep track of uses of variables
- Variables in the context that are not used must be measured:

$$\frac{}{x\colon \sigma, y\colon \tau \vdash x^{\{y\}}\colon \sigma}$$

- Variables in the context can be used more than once (by essentially applying δ)

Using variables

- Does f use x?

$$\boxed{f\ x\ =\ \textbf{if}\ x\ \textbf{then}\ qTrue\ \textbf{else}\ qTrue}$$

- Depends on the semantics of **if**
- Classical control: measure the qubit x to get a classical boolean value and then select appropriate branch
- Quantum control (**if**$^\circ$): return the superposition of the branches
- Quantum control returns $qTrue$ without using x
- The version with **if**$^\circ$ must not be allowed to typecheck

Quantum control & orthogonality

- In an $\mathbf{if}^\circ$ expression, the superposition of e_1 and e_2 is calculated using the probability amplitudes of the superposition of x:

$$\boxed{\mathbf{if}^\circ\ x\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2}$$

- Basically if e_1 and e_2 are orthogonal then it is safe to replace the superposition in x with the superposition of e_1 and e_2.

- This is ok:

$$\boxed{f'^\circ\ x\ =\ \mathbf{if}^\circ\ x\ \mathbf{then}\ qTrue\ \mathbf{else}\ qFalse}$$

Conclusions

- Fairly elegant semantic analysis
- Quantum computing =
  functional language +
  arrows (for parallelism and measurement) +
  some kind of linear type system (to control weakening)
- Formalize the connections between QML and a functional languages with superoperators as arrows
- Still need to explain a few open issues
- Higher-order programs, infinite datatypes, etc