The Quantum Effect

A recipe for Quantum Π

arXiv:2302.01885

Jacques Carette
McMaster University

Chris Heunen
<chris.heunen@ed.ac.uk>
The University of Edinburgh

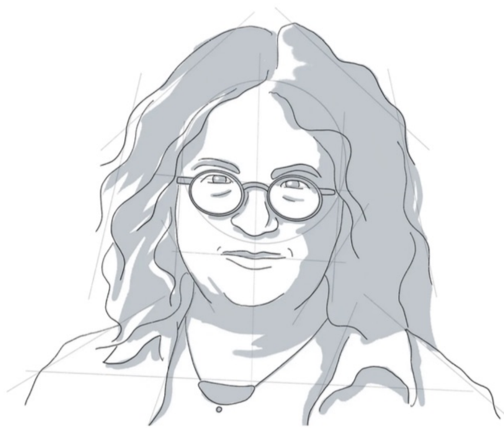
Robin Kaarsgaard
SDU
University of Southern Denmark

Amr Sabry
Indiana University

What makes Quantum ... Quantum ?

Why This Quantum Pioneer Thinks We Need More People Working on Quantum Algorithms

Qiskit · Follow
Published in Qiskit · 8 min read · Mar 15, 2022



Dorit Aharonov (Illustration by J. Russell Huffman)

"One very, very interesting thing about quantum computation is that it touches so many different fields in mathematics," she said. "It's not like that in classical computation. It's really something that is special for quantum computation because it's somehow 'complete' — quantum computation is some kind of completion, mathematically, of classical computation.

Goal: Quantum computing as a completion of classical computing

Computational Effects

make programs do actually useful things:

- receive input, output 
- use randomness 
- provide multiple answers 

Programs that cannot communicate with the outside world are beautiful, perfect, and absolutely useless

Monads, Applicatives, Arrows,

Applicatives: product-preserving functors

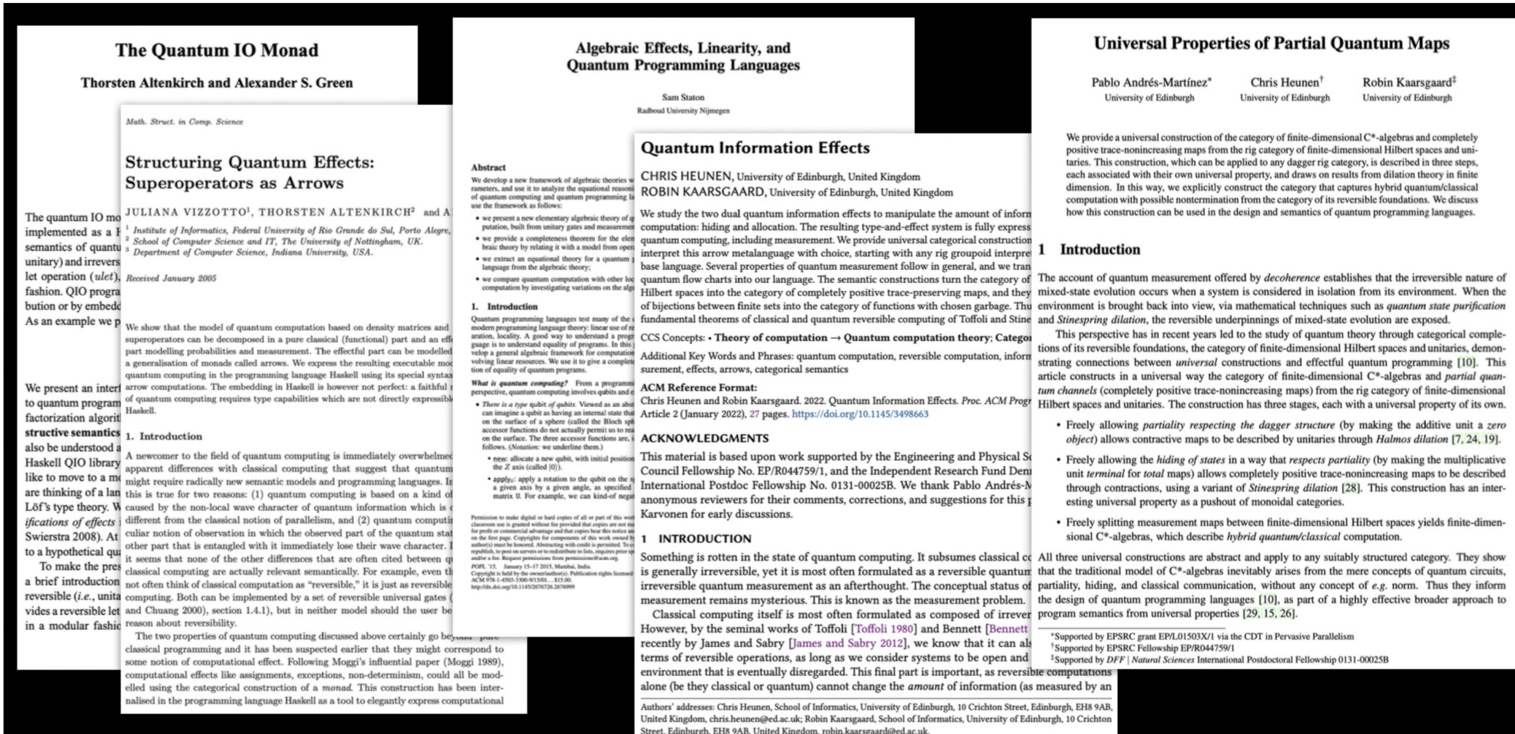
Arrows: Freyd categories

Completions:

$$\begin{tikzcd} \mathbb{C} \arrow[r] \arrow[dr, "F"'] & \hat{\mathbb{C}} \arrow[d, dashed, "\exists!\,\hat{F}"] \\ & \mathbb{D} \end{tikzcd}$$

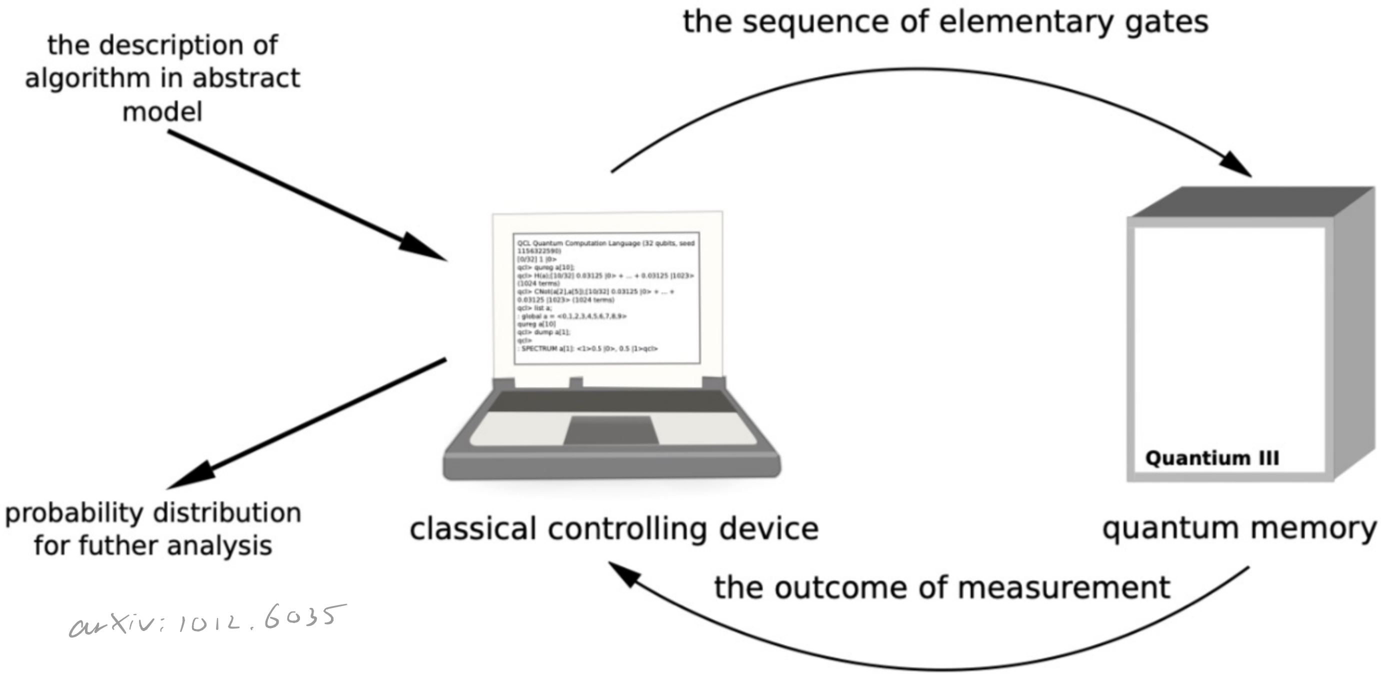
Quantum Computation via Computational Effects

Add quantum features to classical reversible functional programming



measurement, decoherence, nontermination

Is Quantum capability a computational effect?



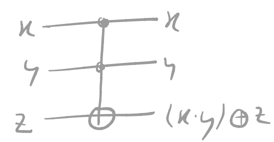
arXiv: 1012.6035

What makes universal (quantum) computing?

classical computing: NAND gate

	0	1
0	1	1
1	1	0

classical reversible computing: TOFFOLI gate



$\boxed{\text{quantum (reversible) computing (Aharonov-Shi): Toffoli + Hadamard is computationally universal}}$

Π: a reversible combinator language

Syntax

$$
\begin{aligned}
b &::= 0 \mid 1 \mid b + b \mid b \times b && \text{(base types)}\\
t &::= b \leftrightarrow b && \text{(combinator types)}\\
a &::= id \mid swap^{+} \mid unit^{+} \mid uniti^{+} \mid assoc^{+} \mid associ^{+} \\
&\quad \mid swap^{\times} \mid unit^{\times} \mid uniti^{\times} \mid assoc^{\times} \mid associ^{\times} \\
&\quad \mid distrib \mid distribi \mid distribo \mid distriboi && \text{(primitive combinators)}\\
c &::= a \mid c \fatsemi c \mid c + c \mid c \times c && \text{(combinators)}
\end{aligned}
$$

Typing rules

$$
\begin{array}{rcccl}
id & : & b \leftrightarrow b & : & id \\
swap^{+} & : & b_1 + b_2 \leftrightarrow b_2 + b_1 & : & swap^{+} \\
unit^{+} & : & b + 0 \leftrightarrow b & : & uniti^{+} \\
assoc^{+} & : & (b_1 + b_2) + b_3 \leftrightarrow b_1 + (b_2 + b_3) & : & associ^{+} \\
swap^{\times} & : & b_1 \times b_2 \leftrightarrow b_2 \times b_1 & : & swap^{\times} \\
unit^{\times} & : & b \times 1 \leftrightarrow b & : & uniti^{\times} \\
assoc^{\times} & : & (b_1 \times b_2) \times b_3 \leftrightarrow b_1 \times (b_2 \times b_3) & : & associ^{\times} \\
distrib & : & b_1 \times (b_2 + b_3) \leftrightarrow (b_1 \times b_2) + (b_1 \times b_3) & : & distribi \\
distribo & : & b \times 0 \leftrightarrow 0 & : & distriboi
\end{array}
$$

$$
\frac{c_1 : b_1 \leftrightarrow b_2 \quad c_2 : b_2 \leftrightarrow b_3}{c_1 \fatsemi c_2 : b_1 \leftrightarrow b_3}
\qquad
\frac{c_1 : b_1 \leftrightarrow b_3 \quad c_2 : b_2 \leftrightarrow b_4}{c_1 + c_2 : b_1 + b_2 \leftrightarrow b_3 + b_4}
\qquad
\frac{c_1 : b_1 \leftrightarrow b_3 \quad c_2 : b_2 \leftrightarrow b_4}{c_1 \times c_2 : b_1 \times b_2 \leftrightarrow b_3 \times b_4}
$$

Π is universal for classical reversible computing:

$$\mathrm{NOT} :: 2 \leftrightarrow 2$$
$$\mathrm{NOT} = swap^{+}$$

$$ctrl :: b \leftrightarrow b \to 2 \times b \leftrightarrow 2 \times b$$
$$ctrl\ f = swap^{\times} \ggg distrib \ggg (unit^{\times} + unit^{\times}) \ggg (id + f) \ggg (uniti^{\times} + uniti^{\times}) \ggg distribi \ggg swap^{\times}$$

$$\mathrm{CNOT} :: 2 \times 2 \leftrightarrow 2 \times 2$$
$$\mathrm{CNOT} = ctrl\ \mathrm{NOT}$$

$$\mathrm{TOFFOLI} :: 2 \times (2 \times 2) \leftrightarrow 2 \times (2 \times 2)$$
$$\mathrm{TOFFOLI} = ctrl\ \mathrm{CNOT}$$

Semantics of Π : rig category $(\mathcal{C}, \otimes, I, \oplus, 0)$

$$A \otimes (B \oplus C) \simeq (A \otimes B) \oplus (A \otimes C) \qquad 0 \otimes A \simeq 0$$

$$(A \oplus B) \otimes C \simeq (A \otimes C) \oplus (B \otimes C) \qquad A \otimes 0 \simeq 0$$

e.g. $\mathbf{FinBij}$, $\mathbf{Top}$, $\mathbf{Hilb}$, $\mathbf{Unitary}$

Thm (Elgueta): $\mathbf{FinBij}$ is bi-initial in $\mathbf{RigCat}$:
every rig cat $\mathcal{C}$ has a rig functor $\mathbf{FinBij} \to \mathcal{C}$
unique up to natural iso

Cor: Π is fully abstract wrt its $\mathbf{FinBij}$-semantics:

$$[\![c_1]\!] = [\![c_2]\!] \text{ in } \mathbf{FinBij} \iff [\![c_1]\!] = [\![c_2]\!] \text{ in any rig cat}$$

So Π is the programming language of rig cats; has TOFFOLI but not Hadamard.

Classical semantics

$[\![\mathrm{NOT}]\!] = \mathrm{NOT}$

$[\![\mathrm{Toffoli}]\!] = \mathrm{Toffoli}$

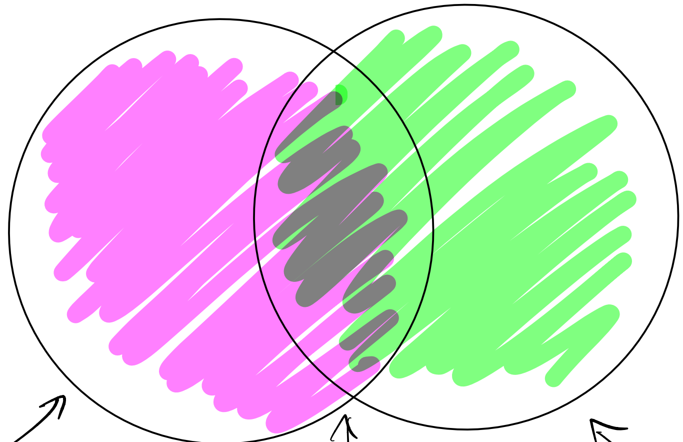
Twisted semantics

for 2×2 unitary M

$[\![c]\!]_M = M^{-1} \circ [\![c]\!] \circ M$

if $M = R_y(\pi/4)$

then $[\![\mathrm{NOT}]\!]_M = H$



✓ Toffoli

✗ Hadamard

Need both
for universal quantum computation!

✗ Toffoli

✓ Hadamard

What if we had two languages?

How to bake a Quantum Π

A simple programming language for combining programs written in two other languages

Syntax

$$
\begin{array}{lr}
b ::= 0 \mid 1 \mid b + b \mid b \times b & \text{(base types)} \\
t ::= b \leftrightarrow b & \text{(combinator types)} \\
a ::= \mathit{id} \mid \mathit{swap}^{+} \mid \mathit{unit}^{+} \mid \mathit{uniti}^{+} \mid \mathit{assoc}^{+} \mid \mathit{associ}^{+} & \\
\quad \mid \mathit{swap}^{\times} \mid \mathit{unit}^{\times} \mid \mathit{uniti}^{\times} \mid \mathit{assoc}^{\times} \mid \mathit{associ}^{\times} & \\
\quad \mid \mathit{distrib} \mid \mathit{distribi} \mid \mathit{distribo} \mid \mathit{distriboi} & \text{(primitive combinators)} \\
c ::= a \mid c \fatsemi c \mid c + c \mid c \times c & \text{(combinators)}
\end{array}
$$

Typing rules

$$
\begin{array}{rccl}
\mathit{id} & : & b \leftrightarrow b & : \mathit{id} \\
\mathit{swap}^{+} & : & b_1 + b_2 \leftrightarrow b_2 + b_1 & : \mathit{swap}^{+} \\
\mathit{unit}^{+} & : & b + 0 \leftrightarrow b & : \mathit{uniti}^{+} \\
\mathit{assoc}^{+} & : & (b_1 + b_2) + b_3 \leftrightarrow b_1 + (b_2 + b_3) & : \mathit{associ}^{+} \\
\mathit{swap}^{\times} & : & b_1 \times b_2 \leftrightarrow b_2 \times b_1 & : \mathit{swap}^{\times} \\
\mathit{unit}^{\times} & : & b \times 1 \leftrightarrow b & : \mathit{uniti}^{\times} \\
\mathit{assoc}^{\times} & : & (b_1 \times b_2) \times b_3 \leftrightarrow b_1 \times (b_2 \times b_3) & : \mathit{associ}^{\times} \\
\mathit{distrib} & : & b_1 \times (b_2 + b_3) \leftrightarrow (b_1 \times b_2) + (b_1 \times b_3) & : \mathit{distribi} \\
\mathit{distribo} & : & b \times 0 \leftrightarrow 0 & : \mathit{distriboi}
\end{array}
$$

$$
\frac{c_1 : b_1 \leftrightarrow b_2 \quad c_2 : b_2 \leftrightarrow b_3}{c_1 \fatsemi c_2 : b_1 \leftrightarrow b_3}
\qquad
\frac{c_1 : b_1 \leftrightarrow b_3 \quad c_2 : b_2 \leftrightarrow b_4}{c_1 + c_2 : b_1 + b_2 \leftrightarrow b_3 + b_4}
\qquad
\frac{c_1 : b_1 \leftrightarrow b_3 \quad c_2 : b_2 \leftrightarrow b_4}{c_1 \times c_2 : b_1 \times b_2 \leftrightarrow b_3 \times b_4}
$$

Syntax

$$
\begin{array}{lr}
b ::= 0 \mid 1 \mid b + b \mid b \times b & \text{(base types)} \\
t ::= b \leftrightarrow b & \text{(combinator types)} \\
a ::= \mathit{id} \mid \mathit{swap}^{+} \mid \mathit{unit}^{+} \mid \mathit{uniti}^{+} \mid \mathit{assoc}^{+} \mid \mathit{associ}^{+} & \\
\quad \mid \mathit{swap}^{\times} \mid \mathit{unit}^{\times} \mid \mathit{uniti}^{\times} \mid \mathit{assoc}^{\times} \mid \mathit{associ}^{\times} & \\
\quad \mid \mathit{distrib} \mid \mathit{distribi} \mid \mathit{distribo} \mid \mathit{distriboi} & \text{(primitive combinators)} \\
c ::= a \mid c \fatsemi c \mid c + c \mid c \times c & \text{(combinators)}
\end{array}
$$

Typing rules

$$
\begin{array}{rccl}
\mathit{id} & : & b \leftrightarrow b & : \mathit{id} \\
\mathit{swap}^{+} & : & b_1 + b_2 \leftrightarrow b_2 + b_1 & : \mathit{swap}^{+} \\
\mathit{unit}^{+} & : & b + 0 \leftrightarrow b & : \mathit{uniti}^{+} \\
\mathit{assoc}^{+} & : & (b_1 + b_2) + b_3 \leftrightarrow b_1 + (b_2 + b_3) & : \mathit{associ}^{+} \\
\mathit{swap}^{\times} & : & b_1 \times b_2 \leftrightarrow b_2 \times b_1 & : \mathit{swap}^{\times} \\
\mathit{unit}^{\times} & : & b \times 1 \leftrightarrow b & : \mathit{uniti}^{\times} \\
\mathit{assoc}^{\times} & : & (b_1 \times b_2) \times b_3 \leftrightarrow b_1 \times (b_2 \times b_3) & : \mathit{associ}^{\times} \\
\mathit{distrib} & : & b_1 \times (b_2 + b_3) \leftrightarrow (b_1 \times b_2) + (b_1 \times b_3) & : \mathit{distribi} \\
\mathit{distribo} & : & b \times 0 \leftrightarrow 0 & : \mathit{distriboi}
\end{array}
$$

$$
\frac{c_1 : b_1 \leftrightarrow b_2 \quad c_2 : b_2 \leftrightarrow b_3}{c_1 \fatsemi c_2 : b_1 \leftrightarrow b_3}
\qquad
\frac{c_1 : b_1 \leftrightarrow b_3 \quad c_2 : b_2 \leftrightarrow b_4}{c_1 + c_2 : b_1 + b_2 \leftrightarrow b_3 + b_4}
\qquad
\frac{c_1 : b_1 \leftrightarrow b_3 \quad c_2 : b_2 \leftrightarrow b_4}{c_1 \times c_2 : b_1 \times b_2 \leftrightarrow b_3 \times b_4}
$$

Types: same as Π

Terms: $[c_1, c_2, c_3, c_4, \ldots]$

Composition: list concatenation

Identities: empty list

Computationally universal !

Semantics of Quantum Π

① Automorphism construction

Let $\mathbb{C}$ be semi-simple rig cat.
and $R : I \oplus I \longrightarrow I \oplus I$.

Make $\mathrm{Aut}_R(\mathbb{C})$ with same objects as $\mathbb{C}$
and conjugated morphisms

Then $\mathrm{Aut}_R(\mathbb{C})$ again rig cat $(\simeq \mathbb{C})$

Unitary gives semantics for Π
$\mathrm{Aut}_{R_y(\pi/4)}(\mathrm{Unitary})$ for Π

② Amalgamation of sym. mon. cats

Let $\mathbb{C}, \mathbb{D}$ be sym. mon. cats w same objs.

Make $\mathrm{Amalg}(\mathbb{C}, \mathbb{D})$ with same objects
and morphisms $[f_1, f_2, f_3, f_4, f_5, f_6, \ldots]$
with $\mathrm{cod}(f_i) = \mathrm{dom}(f_{i+1})$ subject to
$[f_1, \ldots, f_m, id, f_{m+2}, \ldots, f_n] \sim [f_1, \ldots, f_m, f_{m+2}, \ldots, f_n]$
$[f_1, \ldots, f_m, f_{m+1}, \ldots, f_n] \sim [f_1, \ldots, f_m \circ f_{m+1}, \ldots, f_n]$

Further identify $\otimes$ in $\mathbb{C}$ and $\mathbb{D}$.

Then $\mathrm{Amalg}(\mathbb{C}, \mathbb{D})$ is again sym. mon. cat.

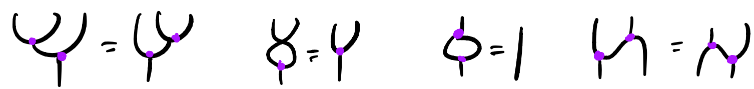
$\mathrm{Amalg}(\mathrm{Unitary}, \mathrm{Aut}_{R_y(\pi/4)}(\mathrm{Unitary}))$
gives semantics for Quantum Π

Carefully chosen semantics $\rightsquigarrow$ computationally universal

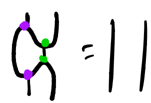
Can equation about π and π guarantee this?

Prop: H is unique real, unitary, involutive transformation between computational basis and mutually unbiased one up to correction by X and -1

- Add states & effects (as further computational effects), define copy (and copy)
- Demand Frobenius:



- Demand complementarity:



- Then NOT is <u>involutive</u> transformation between mutually unbiased bases
- One more equation makes NOT real.

Canonicity by complementarity

Theorem 28 (Canonicity). *If a categorical semantics* $[\![-]\!]$ *for* $\langle \Pi \diamond \rangle$ *in* Contraction *satisfies the classical structure laws and the execution laws (defined in Prop. 24) and the complementarity law (Def. 26), then it is computationally universal. Specifically, it must be the semantics of Sec. 7.3 with the semantics of* x_ϕ *being the Hadamard gate (up to conjugation by* X *and* Z*) and:*

$$[\![copy_Z]\!] : |i\rangle \mapsto |ii\rangle \qquad\qquad [\![zero]\!] = |0\rangle$$

$$[\![copy_X]\!] : |\pm\rangle \mapsto |\pm\pm\rangle \qquad\qquad [\![assertZero]\!] = \langle 0|$$

up to a global unitary.

Q: What's the effect?

A: $\mathbb{C} \longrightarrow \mathrm{Amalg}(\mathbb{C}, \mathrm{Aut}_R(\mathbb{C}))$ is a Freyd category

i.e. a computational effect over the programming language (Π) of $\mathbb{C}$

c.f. SILQ:

qfree ← ad hoc

We use the annotation `qfree` to indicate that evaluating functions or expressions neither introduces nor destroys superpositions. Annotation `qfree` (i) ensures that evaluating `qfree` functions on classical arguments yields classical results and (ii) enables automatic uncomputation.

Example 1 (not `qfree`): `H` is not `qfree` as it introduces superpositions: It maps $|0\rangle$ to $\frac{1}{\sqrt{2^n}}\Big(|0\rangle + |1\rangle\Big)$.

Example 2: `X` is `qfree` as it neither introduces nor destroys superpositions: It maps $\sum_{b=0}^{1} \gamma_b |b\rangle$ to $\sum_{b=0}^{1} \gamma_b |1-b\rangle$.

Example 3: Logical disjunction (as in `x||y`) is of type `const B×const B→qfree B`, since ORing two values neither introduces nor destroys superpositions.

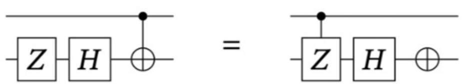
Example 4: Function `myEval` (below) takes a `qfree` function `f` and evaluates it on `false`. Thus, `myEval` itself is also `qfree`.

```
def myEval(f:B→qfree B)qfree{
  return f(false); //   ^ myEval is qfree
}
```

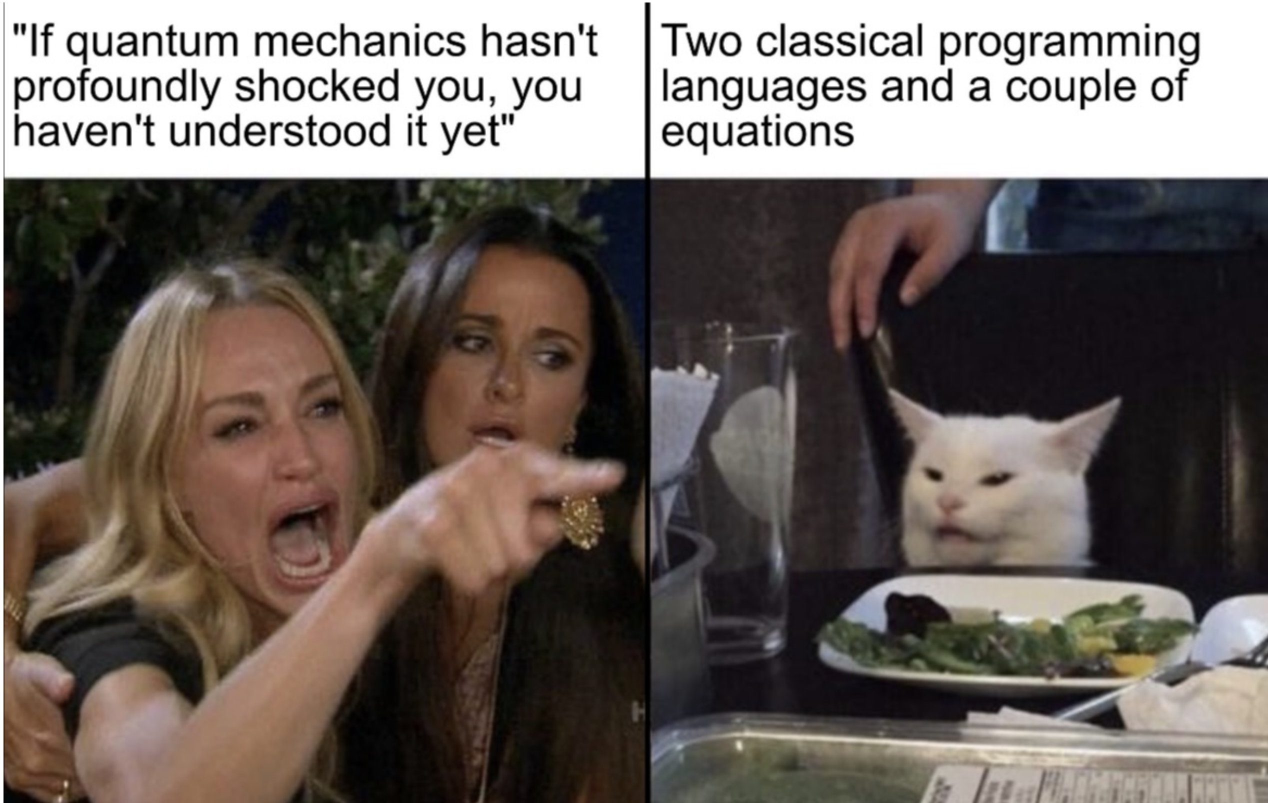
Q: Can effect system give more principled solution?

Reasoning in Quantum ΠΠ

Formalised in Agda




```
zhcx : (id⇔ *** Z) >>> (id⇔ *** H) >>> cx ≡ cz >>> (id⇔ *** H) >>> (id⇔ *** X)
zhcx = begin
  (id⇔ *** Z) >>> (id⇔ *** H) >>> cx
    ≡⟨ id≡ ⟩
  (id⇔ *** (H >>> X >>> H)) >>> (id⇔ *** H) >>> cx
    ≡⟨ assoc>>>l ⊙ (homL*** ⊙ (idl>>>l ⟩⊗⟨id)) ⟩⨾⟨id ⟩
  (id⇔ *** ((H >>> X >>> H) >>> H)) >>> cx
    ≡⟨ id⟩⊗⟨ pullʳ (cancelʳ hadInv) ⟩⨾⟨id ⟩
  id⇔ *** (H >>> X) >>> cx
    ≡⟨ (idl>>>r ⟩⊗⟨id ⊙ homR***) ⟩⨾⟨id ⊙ assoc>>>r ⟩
  (id⇔ *** H) >>> (id⇔ *** X) >>> cx
    ≡⟨ id⟩⨾⟨ xcxA ⟩
  (id⇔ *** H) >>> cx >>> (id⇔ *** X)
    ≡⟨ id⟩⨾⟨ id⟩⨾⟨ insertˡ 1*HInv ⟩
  (id⇔ *** H) >>> cx >>> (id⇔ *** H) >>> (id⇔ *** H) >>> (id⇔ *** X)
    ≡⟨ assoc>>>l ⊙ assoc>>>l ⊙ assoc>>>r ⟩⨾⟨id ⟩
  (id⇔ *** H >>> cx >>> id⇔ *** H) >>> (id⇔ *** H) >>> (id⇔ *** X)
    ≡⟨ id≡ ⟩
  cz >>> (id⇔ *** H) >>> (id⇔ *** X)      ∎
```


"If quantum mechanics hasn't profoundly shocked you, you haven't understood it yet"
Two classical programming languages and a couple of equations