A HoTT Quantum Equational Theory

Jennifer Paykin
Galois, Inc

Steve Zdancewic
University of Pennsylvania

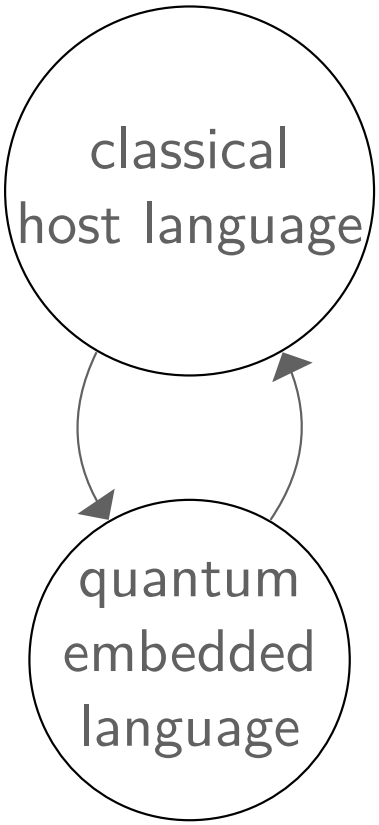
QPL
June 10, 2019

|galois|

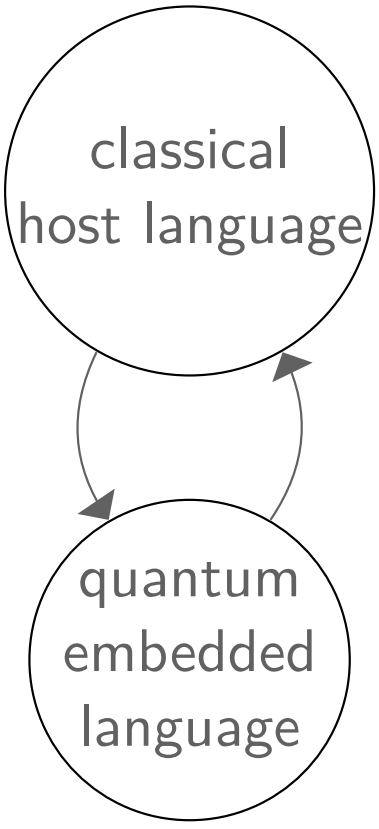
© 2019 Galois, Inc.

|g|

Embedded quantum languages

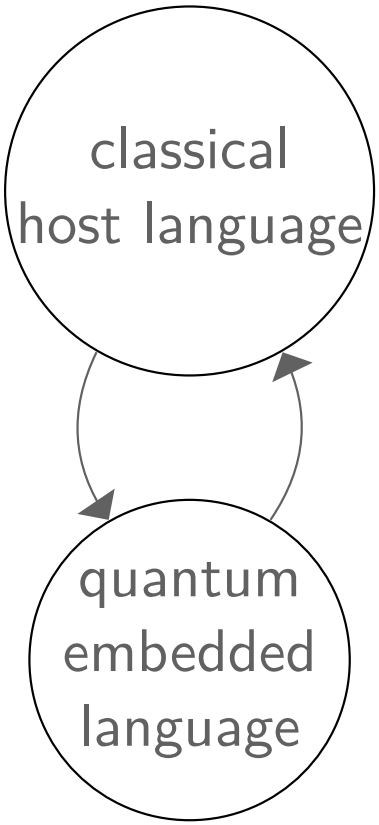


Embedded quantum languages



- Quantum data: qubits, unitaries, measurement
  - gate model

Embedded quantum languages

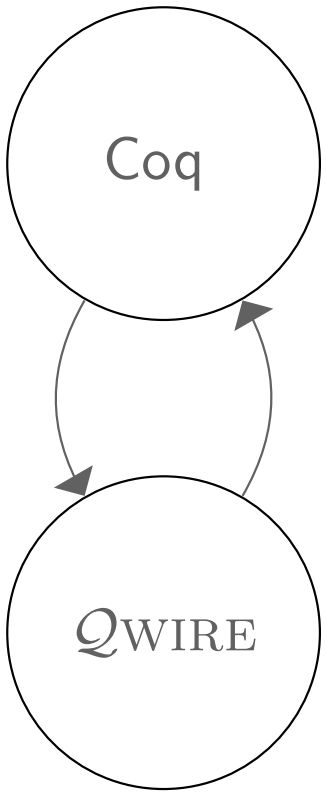


- Quantum data: qubits, unitaries, measurement
  - gate model
- Classical control: host abstractions (functions, recursion, types), tools

Reasoning about standalone quantum languages

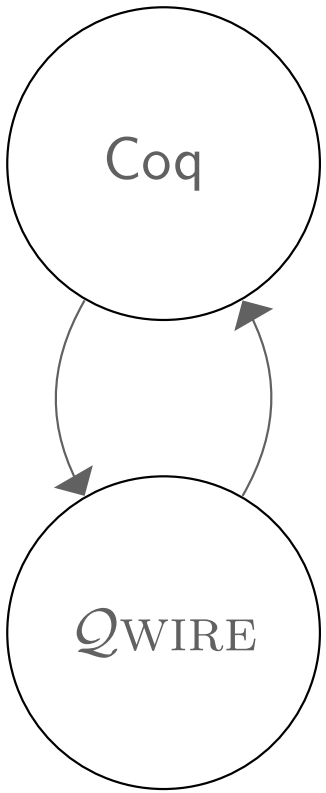
- Denotational semantics
  - QPL [Selinger, 2004], Proto-Quipper-M [Rios and Selinger, 2018]
- Program logics
  - Ying [2011], Unruh [2019]
- Equational semantics
  - ZX calculus [Coecke and Duncan, 2011], algebraic effects [Staton, 2015]

Reasoning about embedded quantum languages



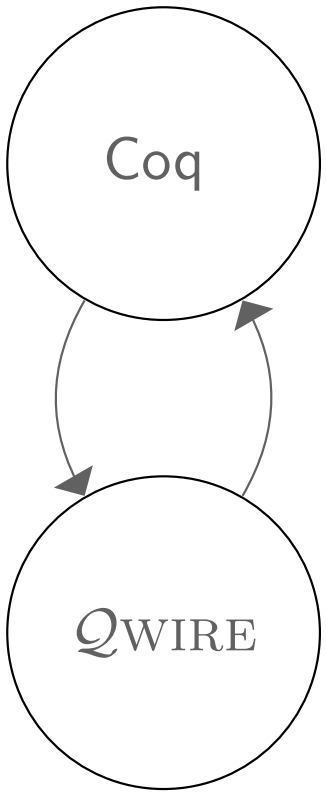
- Host language as logical framework
  - QWIRE in Coq [Rand et al., 2017]

Reasoning about embedded quantum languages



- Host language as logical framework
  - QWIRE in Coq [Rand et al., 2017]
- Denotational semantics
  - Density matrices [Paykin et al., 2017]
  - Enriched category theory [Rennela and Staton, 2018]

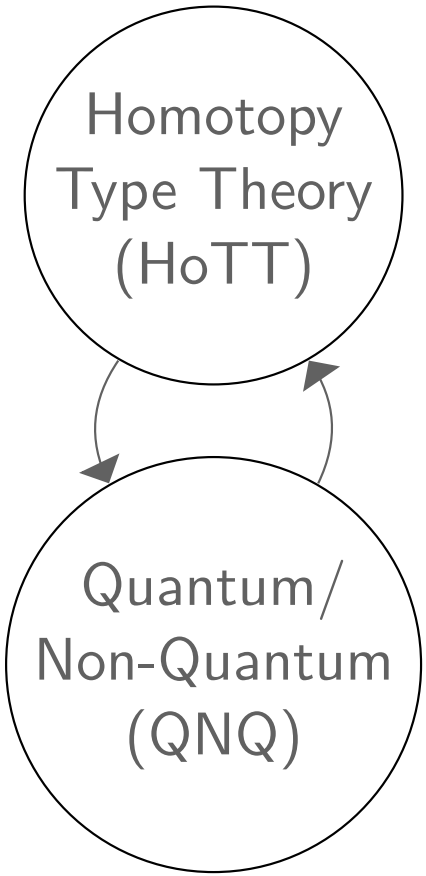
Reasoning about embedded quantum languages



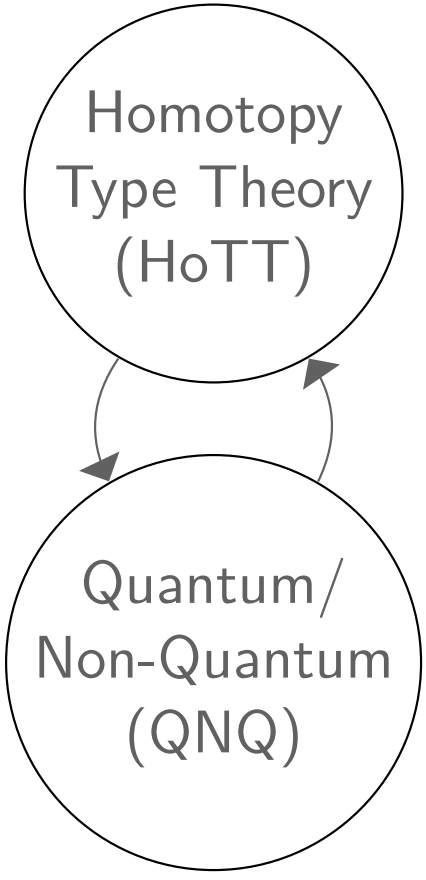
- Host language as logical framework
  - QWIRE in Coq [Rand et al., 2017]
- Denotational semantics
  - Density matrices [Paykin et al., 2017]
  - Enriched category theory [Rennela and Staton, 2018]
- Equational theory?

Homotopy type theory (HoTT): a type theory of equality

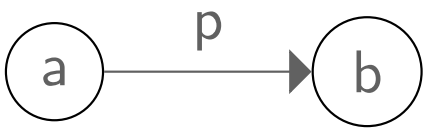
- Equality type: $a = b$
  - where $a, b : \alpha$



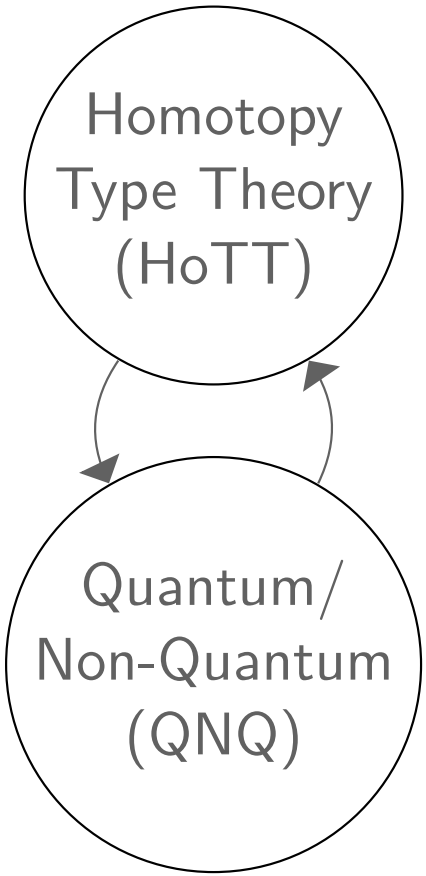
Homotopy type theory (HoTT): a type theory of equality



- Equality type: $a = b$
  - where $a, b : \alpha$
- Proofs $p : a = b$ called *paths*

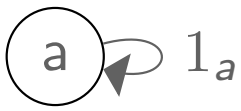


Homotopy type theory (HoTT): a type theory of equality

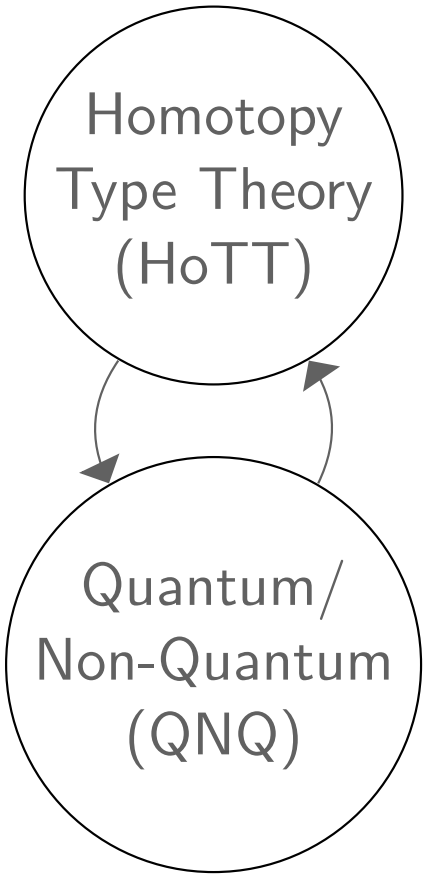


- Equality type: $a = b$
  - where $a, b : \alpha$
- Proofs $p : a = b$ called *paths*

- Constructor: $1_a : a = a$

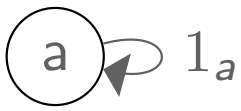


Homotopy type theory (HoTT): a type theory of equality



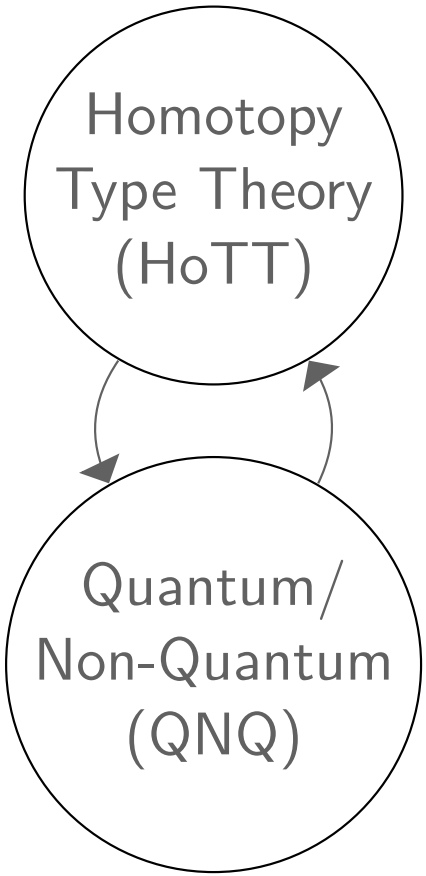
- Equality type: $a = b$
  - where $a, b : \alpha$
- Proofs $p : a = b$ called *paths*

- Constructor: $1_a : a = a$



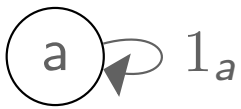
- Path induction:
  - to show $\forall(p : a = b).\ H(p)$,
  - suffices to show $\forall a.\ H(1_a)$.

Homotopy type theory (HoTT): a type theory of equality



- Equality type: $a = b$
  - where $a, b : \alpha$
- Proofs $p : a = b$ called *paths*

- Constructor: $1_a : a = a$



- Path induction:
  - to show $\forall(p : a = b).\ H(p)$,
  - suffices to show $\forall a.\ H(1_a)$.
- Not all paths are equal to 1_a...

Higher Inductive Type (HIT)

Definition

The quotient of a type α by a relation $R : \alpha \to \alpha \to \text{Prop}$ is a type α/R with data constructor:

$$\frac{a : \alpha}{[a]_R : \alpha/R}$$

...

Higher Inductive Type (HIT)

Definition

The quotient of a type α by a relation $R : \alpha \to \alpha \to \text{Prop}$ is a type α/R with data constructor:

$$\frac{a : \alpha}{[a]_R : \alpha/R}$$

... and path constructor:

$$\frac{a, b : \alpha \qquad p : R(a, b)}{[p] : [a]_R = [b]_R}$$

Higher Inductive Type (HIT)

Definition

The quotient of a type α by a relation $R : \alpha \to \alpha \to \text{Prop}$ is a type α/R with data constructor:

$$\frac{a : \alpha}{[a]_R : \alpha/R}$$

... and path constructor:

$$\frac{a, b : \alpha \qquad p : R(a, b)}{[p] : [a]_R = [b]_R}$$

Note

$p : R(a, a)$ *does not imply* $[p] = 1_a$

Idea: Represent unitaries as paths

- Higher Inductive Types (HITs) use paths to encode *equivalence relations* or *groupoids*.
  - Groupoid: category where all morphisms are invertible

Idea: Represent unitaries as paths

- Higher Inductive Types (HITs) use paths to encode *equivalence relations* or *groupoids*.
  - Groupoid: category where all morphisms are invertible

$$\frac{f : G(\alpha, \beta)}{[f] : [\alpha] = [\beta]}$$

Idea: Represent unitaries as paths

- Higher Inductive Types (HITs) use paths to encode *equivalence relations* or *groupoids*.
  - Groupoid: category where all morphisms are invertible

$$\frac{f : G(\alpha, \beta)}{[f] : [\alpha] = [\beta]}$$

- Path induction still holds of HITs:
  - Prove theorems with base case $1_a : a = a$.
  - Simplify proofs about groupoids

Idea: Represent unitaries as paths

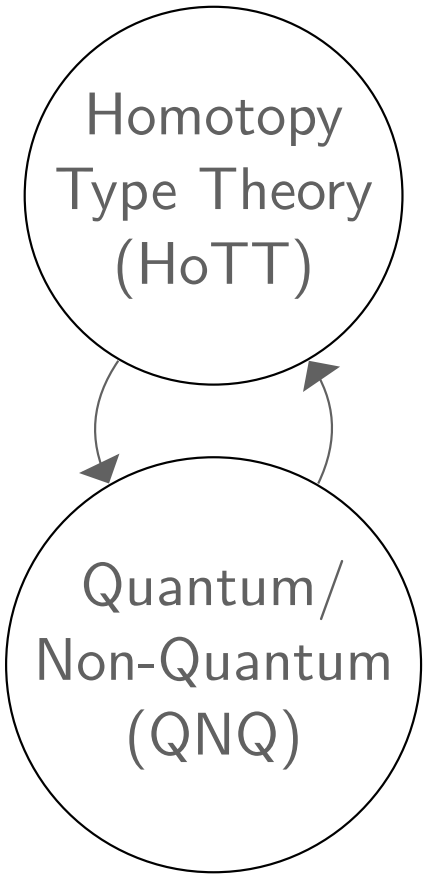
- Higher Inductive Types (HITs) use paths to encode *equivalence relations* or *groupoids*.
  - Groupoid: category where all morphisms are invertible

$$\frac{f : G(\alpha, \beta)}{[f] : [\alpha] = [\beta]}$$

- Path induction still holds of HITs:
  - Prove theorems with base case $1_a : a = a$.
  - Simplify proofs about groupoids

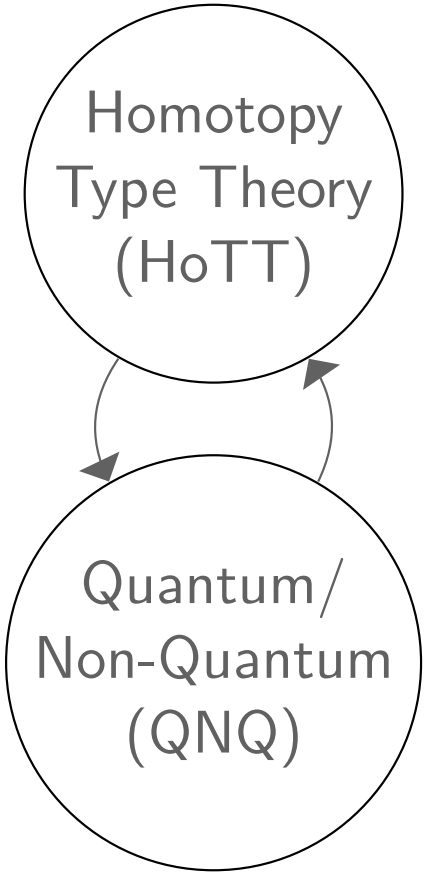
- Unitary transformations form a groupoid.

Quantum/non-quantum calculus [Paykin, 2018]



- Host terms $a : \alpha$

Quantum/non-quantum calculus [Paykin, 2018]



- Host terms $a : \alpha$
- Embedded quantum expressions $e : \mathsf{QExp}\ \sigma$

Quantum operators

$$\frac{b : \text{Bool}}{|b\rangle : \text{QExp Qubit}}\ \text{INIT}$$

Quantum operators

$$\frac{b : \mathsf{Bool}}{|b\rangle : \mathsf{QExp}\ \mathsf{Qubit}}\ \textsc{init}$$

$$\frac{e : \mathsf{QExp}\ \mathsf{Qubit} \qquad f : \mathsf{Bool} \to \mathsf{QExp}\ \tau}{e >!\ f : \mathsf{QExp}\ \tau}\ \textsc{measure}$$

Idea: Represent unitaries as paths

Idea: Represent unitaries as paths

- UMatrix(α, β): unitary matrices of dimension $|\alpha| \times |\beta|$.
  - α, β : FinType are finite types
  - Because unitaries are square, $|\alpha| = |\beta|$.

Idea: Represent unitaries as paths

- UMatrix(α, β): unitary matrices of dimension $|\alpha| \times |\beta|$.
  - α, β : FinType are finite types
  - Because unitaries are square, $|\alpha| = |\beta|$.

- Quantum types: QType = FinType/UMatrix.
  - Qubit = $[\mathsf{Bool}]_{\mathsf{UMatrix}}$

Idea: Represent unitaries as paths

- UMatrix(α, β): unitary matrices of dimension $|\alpha| \times |\beta|$.
  - α, β : FinType are finite types
  - Because unitaries are square, $|\alpha| = |\beta|$.
- Quantum types: QType $=$ FinType/UMatrix.
  - Qubit $=$ $[\mathsf{Bool}]_{\mathsf{UMatrix}}$
- Unitaries are paths:

$$\frac{U : \mathsf{UMatrix}(\alpha, \beta)}{[U] : [\alpha] = [\beta]}$$

Idea: Represent unitaries as paths

- UMatrix(α, β): unitary matrices of dimension $|\alpha| \times |\beta|$.
  - α, β : FinType are finite types
  - Because unitaries are square, $|\alpha| = |\beta|$.
- Quantum types: QType $=$ FinType/UMatrix.
  - Qubit $= [\mathsf{Bool}]_{\mathsf{UMatrix}}$
- Unitaries are paths:

$$\frac{U : \mathsf{UMatrix}(\alpha, \beta)}{[U] : [\alpha] = [\beta]}$$

  - $H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$, $X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$

Idea: Represent unitaries as paths

- UMatrix(α, β): unitary matrices of dimension $|\alpha| \times |\beta|$.
  - α, β : FinType are finite types
  - Because unitaries are square, $|\alpha| = |\beta|$.

- Quantum types: QType $=$ FinType/UMatrix.
  - Qubit $= [\mathsf{Bool}]_{\mathsf{UMatrix}}$

- Unitaries are paths:

$$\frac{U : \mathsf{UMatrix}(\alpha, \beta)}{[U] : [\alpha] = [\beta]}$$

  - $H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$, $X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$
  - $[H] : \mathsf{Qubit} = \mathsf{Qubit}$, $[X] : \mathsf{Qubit} = \mathsf{Qubit}$

Idea: Represent unitaries as paths

- UMatrix(α, β): unitary matrices of dimension $|\alpha| \times |\beta|$.
  - α, β : FinType are finite types
  - Because unitaries are square, $|\alpha| = |\beta|$.

- Quantum types: QType $=$ FinType/UMatrix.
  - Qubit $= [\mathsf{Bool}]_{\mathsf{UMatrix}}$

- Unitaries are paths:

$$\frac{U : \mathsf{UMatrix}(\alpha, \beta)}{[U] : [\alpha] = [\beta]}$$

  - $H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$, $X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$
  - $[H] : \mathsf{Qubit} = \mathsf{Qubit}$, $[X] : \mathsf{Qubit} = \mathsf{Qubit}$
  - $[H] \neq [X] \neq 1_{\mathsf{Qubit}}$

HoTT QNQ calculus

$$\sigma \in \mathsf{QType} = \mathsf{FinType}/\mathsf{UMatrix}$$

$$\begin{aligned} e := x \mid\ & \mathsf{let}\ x := e\ \mathsf{in}\ e' \\ \mid\ & (e_1, e_2) \mid \mathsf{let}\ (x_1, x_2) := e\ \mathsf{in}\ e' \\ \mid\ & |a\rangle \mid e >!\ f \end{aligned}$$

HoTT QNQ calculus

$$\sigma \in \mathsf{QType} = \mathsf{FinType}/\mathsf{UMatrix}$$

$$\begin{aligned} e := x \mid\ & \mathsf{let}\ x := e\ \mathsf{in}\ e' \\ \mid\ & (e_1, e_2) \mid \mathsf{let}\ (x_1, x_2) := e\ \mathsf{in}\ e' \\ \mid\ & |a\rangle \mid e >!\ f \end{aligned}$$

- β- and $\eta-$ equivalences:

$$|b\rangle >!\ f \quad \sim_\beta \quad f\ b$$

HoTT QNQ calculus

$$\sigma \in \mathsf{QType} = \mathsf{FinType}/\mathsf{UMatrix}$$

$$\begin{aligned} e := x \mid\ & \mathsf{let}\ x := e\ \mathsf{in}\ e' \\ \mid\ & (e_1, e_2) \mid \mathsf{let}\ (x_1, x_2) := e\ \mathsf{in}\ e' \\ \mid\ & |a\rangle \mid e >!\ f \end{aligned}$$

- β- and $\eta-$ equivalences:

$$|b\rangle >!\ f \quad \sim_\beta \quad f\ b$$

. . . unitaries?

Unitary application

Theorem

Let U be a unitary transformation $U : \sigma = \tau$.
$(\sigma, \tau : QType \equiv FinType/UMatrix)$

If $e : QExp\ \sigma$, there exists $U \# e : QExp\ \tau$.
(apply the unitary U to e)

Unitary application

Theorem

Let U be a unitary transformation $U : \sigma = \tau$.
$(\sigma, \tau : QType \equiv FinType/UMatrix)$

If $e : QExp\ \sigma$, there exists $U \# e : QExp\ \tau$.
(apply the unitary U to e)

Proof.

By path induction. Base case for $1_\sigma : \sigma = \sigma$:

$$1_\sigma \# e \equiv e$$

□

Unitary application

Theorem

Let U be a unitary transformation $U : \sigma = \tau$.
$(\sigma, \tau : QType \equiv FinType/UMatrix)$

If $e : QExp\ \sigma$, there exists $U \# e : QExp\ \tau$.
(apply the unitary U to e)

Proof.

By path induction. Base case for $1_\sigma : \sigma = \sigma$:

$$1_\sigma \# e \equiv e$$

□

Note

$[H] \# e \neq e$ *because* $[H] \neq 1_{Qubit}$

Goal

Equational theory for *embedded* quantum language.

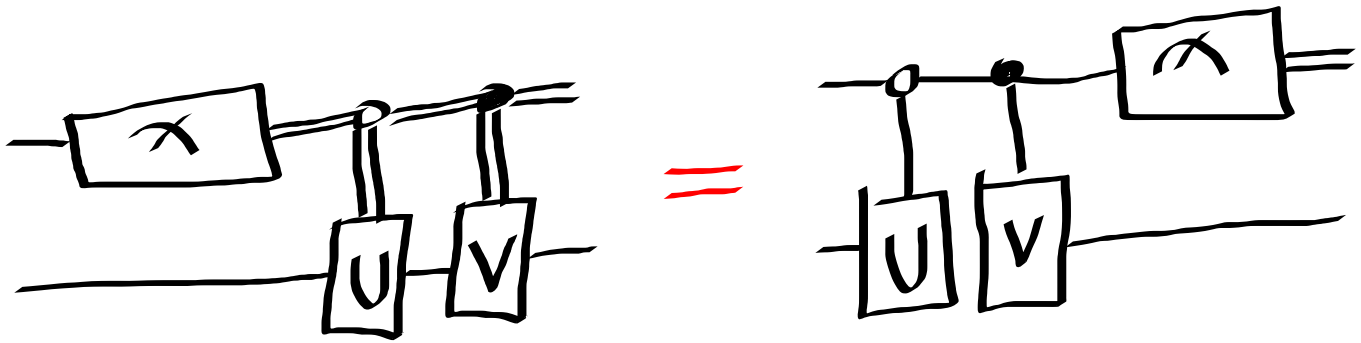
Goal

Equational theory for *embedded* quantum language.

- Focus on boundary of quantum data/classical control
  - NOT equational theory for classes of unitaries
  - e.g. $XZX = -Z$ implies $[XZX] = [-Z]$

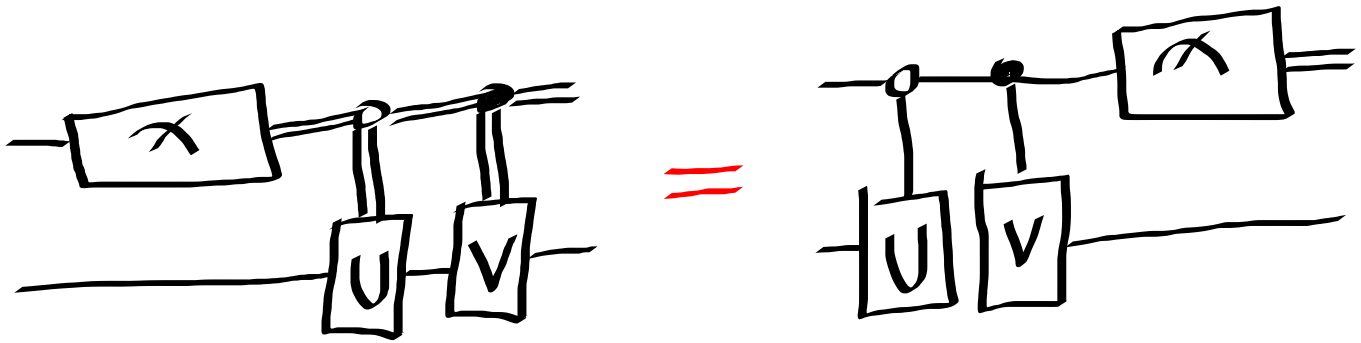
Prior work – Staton [2015]

- Equational theory for *algebra* with unitaries and classical control.

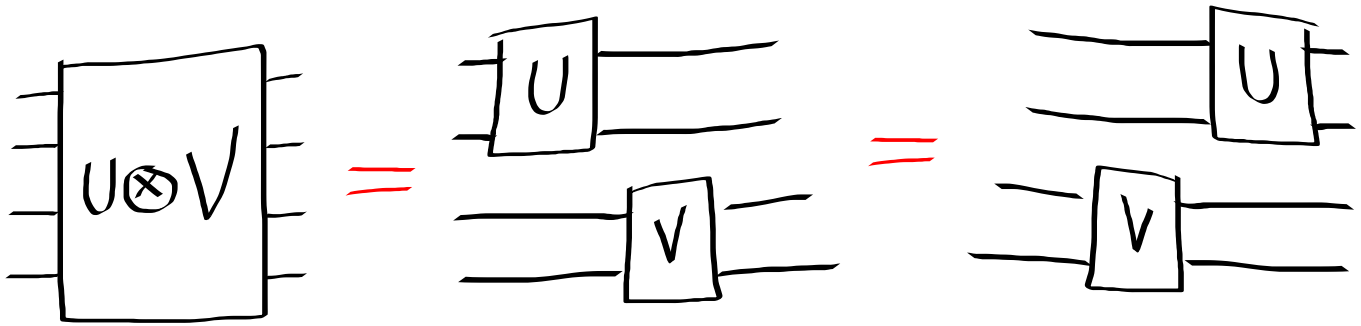


Prior work – Staton [2015]

- Equational theory for *algebra* with unitaries and classical control.

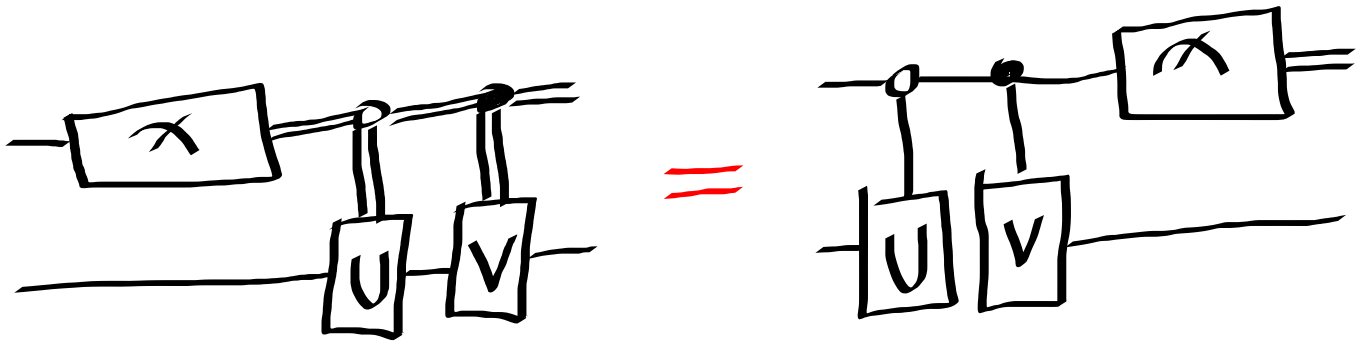


- Procedural axioms based on diagrams
  - symmetric monoidal structure

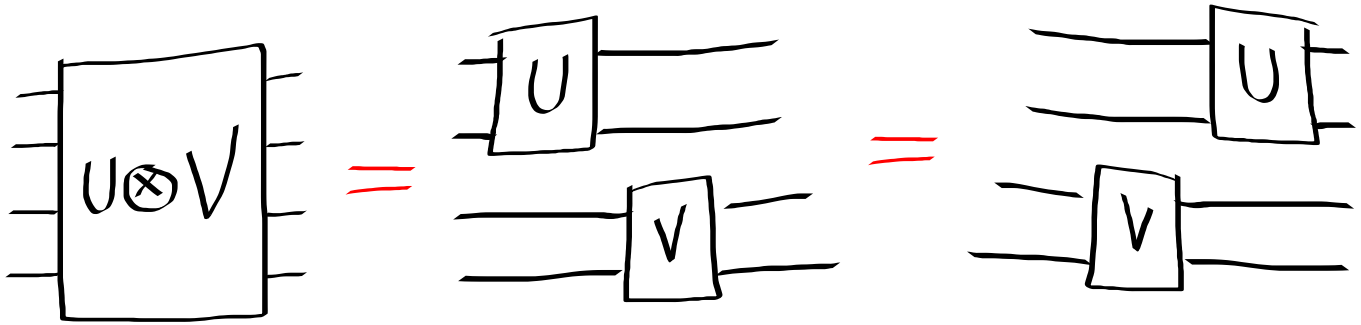


Prior work – Staton [2015]

- Equational theory for *algebra* with unitaries and classical control.



- Procedural axioms based on diagrams
  - symmetric monoidal structure



- Sound and complete with respect to C^*-algebras.
  - $d = 2$

Goal

Equational theory for *embedded* QNQ in HoTT.

Goal

Equational theory for *embedded* QNQ in HoTT.

- *languages*, not algebra or diagrams
  - Variables, functions, modularity, tuples,...

Goal

Equational theory for *embedded* QNQ in HoTT.

- *languages*, not algebra or diagrams
  - Variables, functions, modularity, tuples,...
- Highlight interesting axioms; fewer "procedural" axioms.
  - Paths in HoTT already yield these rules
  - Can derive many axioms for free with HoTT

Goal

Equational theory for *embedded* QNQ in HoTT.

- *languages*, not algebra or diagrams
  - Variables, functions, modularity, tuples,...
- Highlight interesting axioms; fewer "procedural" axioms.
  - Paths in HoTT already yield these rules
  - Can derive many axioms for free with HoTT
- Completeness by comparison with Staton's theory.

Unitaries in the HoTT QNQ

Theorem

Let $U : \sigma = \tau$ *and* $V : \tau = \rho$ *be unitaries. Then*

$$V \# (U \# e) = (V \circ U) \# e.$$

Unitaries in the HoTT QNQ

Theorem

Let $U : \sigma = \tau$ and $V : \tau = \rho$ be unitaries. Then

$$V \# (U \# e) = (V \circ U) \# e.$$

Proof.

By path induction on V. If $V \equiv 1_\tau$ then

$$LHS = 1_\tau \# (U \# e) = U \# e$$
$$RHS = (1_\tau \circ U) \# e = U \# e$$

□

We can prove a lot...

Define versions of †, $\otimes$, etc on paths.

We can prove a lot...

Define versions of †, $\otimes$, etc on paths.

Theorem

$U^\dagger \# (U \# e) = e$

We can prove a lot...

Define versions of †, $\otimes$, etc on paths.

Theorem

$U^\dagger \# (U \# e) = e$

Theorem

$(U_1 \otimes U_2) \# (e_1, e_2) = (U_1 \# e_1, U_2 \# e_2)$

We can prove a lot...

Define versions of †, $\otimes$, etc on paths.

Theorem

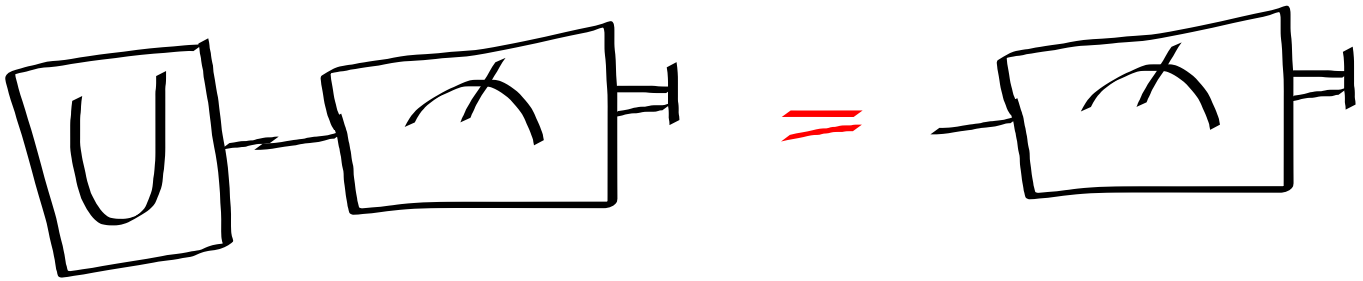
$U^\dagger \# (U \# e) = e$

Theorem

$(U_1 \otimes U_2) \# (e_1, e_2) = (U_1 \# e_1, U_2 \# e_2)$

Theorem

$discard(meas(U \# e)) = discard(meas(e))$



...but not everything

Theorem

$[SWAP] \# (e_1, e_2) = (e_2, e_1)$

Proof.

????

□

...but not everything

Theorem

$[SWAP] \# (e_1, e_2) = (e_2, e_1)$

Proof.

????

□

Theorem

$let\ (x, y) := [SWAP] \# e\ in\ e' = let\ (y, x) := e\ in\ e'$

Proof.

????

□

...but not everything

Theorem

$[SWAP] \# (e_1, e_2) = (e_2, e_1)$

Proof.

????

□

Theorem

$let\ (x, y) := [SWAP] \# e\ in\ e' = let\ (y, x) := e\ in\ e'$

Proof.

????

□

Similar for other "structural" unitaries:

$$\text{ASSOC} : \sigma_1 \otimes (\sigma_2 \otimes \sigma_3) = (\sigma_1 \otimes \sigma_2) \otimes \sigma_3$$

$$\text{DISTR} : \text{Qubit} \otimes \tau = \tau \oplus \tau$$

$\vdots$

Partial initialization axiom

Structural equivalence $\sigma \leftrightsquigarrow \tau$:

$$\mathsf{swap}_{X,Y} : X \times Y \to Y \times X$$
$$\mathsf{swap}_{X,Y}(x, y) = (y, x)$$

Partial initialization axiom

Structural equivalence $\sigma \leftrightsquigarrow \tau$:

$$\mathsf{swap}_{X,Y} : X \times Y \to Y \times X$$
$$\mathsf{swap}_{X,Y}(x, y) = (y, x)$$

Lift structural equivalence to unitary:

$$\widehat{\mathsf{swap}}_{\sigma,\tau} : \sigma \otimes \tau = \tau \otimes \sigma$$

such that

$$\widehat{\mathsf{swap}}_{\sigma,\tau} = [\mathsf{SWAP}_{\sigma,\tau}]$$

Partial initialization axiom

Axiom

Let $f : \sigma \leftrightsquigarrow \tau$ be a structural equivalence. Then

$$\widehat{f} \# \mathit{init}_\sigma(b) \approx \mathit{init}_\tau(f(b))$$

Partial initialization axiom

Axiom

Let $f : \sigma \leftrightsquigarrow \tau$ *be a structural equivalence. Then*

$$\widehat{f} \# \mathit{init}_\sigma(b) \approx \mathit{init}_\tau(f(b))$$

Partial initialization:

$$\widehat{\mathsf{swap}_{X,Y}} \# (e_1, e_2) \approx \mathsf{swap}(e_1, e_2) = (e_2, e_1)$$

Partial measurement axiom

Axiom

Let $f : \sigma \leftrightsquigarrow \tau$. *Then:*

$$\mathit{match}\ \widehat{f}\ \#\ e\ \mathit{with}\ g \approx \mathit{match}\ e\ \mathit{with}\ (g \circ f)$$

Partial measurement axiom

Axiom

Let $f : \sigma \leftrightsquigarrow \tau$. *Then:*

$$\textit{match } \widehat{f} \# \ e \textit{ with } g \approx \textit{match } e \textit{ with } (g \circ f)$$

Partial measurement:

$$\begin{array}{l} \text{match } \widehat{\text{swap}} \ \# \ e \text{ with } \lambda(x, y).e' \\ \approx \\ \text{match } e \text{ with } \lambda(y, x).e' \end{array}$$

Results

- Two axioms:
  - structural unitaries + initialization
  - structural unitaries + measurement

[1] https://github.com/jpaykin/GroupoidQuotient

Results

- Two axioms:
  - structural unitaries + initialization
  - structural unitaries + measurement
- Quantum programming language embedded in HoTT
  - (Finite) classical data, tuples, and sums
  - Partially formalized in Coq HoTT library[1]

[1] https://github.com/jpaykin/GroupoidQuotient

Results

- Two axioms:
  - structural unitaries + initialization
  - structural unitaries + measurement
- Quantum programming language embedded in HoTT
  - (Finite) classical data, tuples, and sums
  - Partially formalized in Coq HoTT library[1]
- Sound with respect to density matrices
- Complete with respect to Staton's equational theory
  - Staton's "interesting" axioms are derived in HoTT
  - Staton's "structural" axioms are interesting in HoTT

[1] https://github.com/jpaykin/GroupoidQuotient

Results

- Two axioms:
  - structural unitaries + initialization
  - structural unitaries + measurement
- Quantum programming language embedded in HoTT
  - (Finite) classical data, tuples, and sums
  - Partially formalized in Coq HoTT library[1]
- Sound with respect to density matrices
- Complete with respect to Staton's equational theory
  - Staton's "interesting" axioms are derived in HoTT
  - Staton's "structural" axioms are interesting in HoTT

[1] https://github.com/jpaykin/GroupoidQuotient

Results

- Pros: theorems for free with path induction

Results

- Pros: theorems for free with path induction
- Cons: theorems not actually free

Results

- Pros: theorems for free with path induction
- Cons: theorems not actually free
- Takeaway: Equations stem (mostly) from quantum data/classical control, not artificial axioms

Results

- Pros: theorems for free with path induction
- Cons: theorems not actually free
- Takeaway: Equations stem (mostly) from quantum data/classical control, not artificial axioms

Thanks!

A HoTT Quantum Equational Theory

Jennifer Paykin
Galois, Inc

Steve Zdancewic
University of Pennsylvania

QPL
June 10, 2019

Supported by FA9550-16-1-0082

| galois |

© 2019 Galois, Inc.

| g |

References I

B. Coecke and R. Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, apr 2011. doi: 10.1088/1367-2630/13/4/043016.

J. Paykin. *Linear/Non-Linear Types for Embedded Domain-Specific Languages*. PhD thesis, University of Pennsylvania, 2018. URL `https://repository.upenn.edu/edissertations/2752`.

J. Paykin, R. Rand, and S. Zdancewic. QWIRE: A core language for quantum circuits. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, POPL 2017, pages 846–858, New York, NY, USA, 2017. ACM. doi: 10.1145/3009837.3009894.

References II

R. Rand, J. Paykin, and S. Zdancewic. QWIRE practice: Formal verification of quantum circuits in Coq. In *Proceedings 14th International Conference on Quantum Physics and Logic, QPL 2017, Nijmegen, The Netherlands, 3-7 July 2017.*, pages 119–132, 2017. doi: 10.4204/EPTCS.266.8.

M. Rennela and S. Staton. Classical control and quantum circuits in enriched category theory. *Electronic Notes in Theoretical Computer Science*, 336:257 – 279, 2018. ISSN 1571-0661. doi: 10.1016/j.entcs.2018.03.027. The Thirty-third Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXIII).

F. Rios and P. Selinger. A categorical model for a quantum circuit description language (extended abstract). *Electronic Proceedings in Theoretical Computer Science*, 266:164–178, feb 2018. doi: 10.4204/eptcs.266.11.

References III

P. Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004. doi: 10.1017/S0960129504004256.

S. Staton. Algebraic effects, linearity, and quantum programming languages. In *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '15, pages 395–406, New York, NY, USA, 2015. ACM. doi: 10.1145/2676726.2676999.

D. Unruh. Quantum relational hoare logic. *Proceedings of the ACM on Programming Languages*, 3(POPL):33, 2019.

M. Ying. Floyd–Hoare logic for quantum programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 33(6):19, 2011.