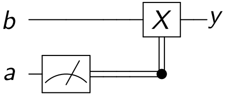
Classical Control and Quantum Circuits in Enriched Category Theory

Mathys Rennela (Radboud University)
Sam Staton (Oxford University)
MFPS XXXIII
Wednesday 14 June 2017

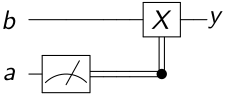
Where we are, sofar

Programming quantum circuits



$$
\begin{aligned}
-; a, b : \mathrm{qubit} \vdash C \stackrel{\text{def}}{=} & x \leftarrow \textbf{gate meas } a; \\
& (x, y) \leftarrow \textbf{gate } (\textbf{bit-control } X)\,(x, b); \\
& () \leftarrow \textbf{gate discard } x; \textbf{output } y \qquad : \ \mathrm{qubit}
\end{aligned}
$$

Programming quantum circuits



$$
\begin{aligned}
-; a, b : \mathrm{qubit} \vdash C \stackrel{\mathsf{def}}{=} x &\leftarrow \textbf{gate meas } a; \\
(x, y) &\leftarrow \textbf{gate } (\textbf{bit-control } X)\,(x, b); \\
() &\leftarrow \textbf{gate discard } x; \textbf{output } y \qquad : \ \mathrm{qubit}
\end{aligned}
$$

- Problem: not all quantum protocols are that simple...

Embedding as enrichment

- **Circuit language:** symmetric monoidal category $\mathbf{C}$
- Wire type W: object W of $\mathbf{C}$

Embedding as enrichment

- **Circuit language:** symmetric monoidal category $\mathbf{C}$
- Wire type W: object W of $\mathbf{C}$
- Program = Circuit of entry type W_1 and output type W_2:
  $\mathbf{C}$-homomorphism $W_1 \to W_2$

Embedding as enrichment

- **Circuit language:** symmetric monoidal category $\mathbf{C}$
- Wire type W: object W of $\mathbf{C}$
- Program = Circuit of entry type W_1 and output type W_2:
$$\mathbf{C}\text{-homomorphism } W_1 \to W_2$$
- **Host language:** cartesian closed category $\mathbf{H}$
- "General purpose" language
- Host type: object of $\mathbf{H}$

Embedding as enrichment

- **Circuit language:** symmetric monoidal category $\mathbf{C}$
- Wire type W: object W of $\mathbf{C}$
- Program = Circuit of entry type W_1 and output type W_2:
  $$\mathbf{C}\text{-homomorphism } W_1 \to W_2$$

- **Host language:** cartesian closed category $\mathbf{H}$
- "General purpose" language
- Host type: object of $\mathbf{H}$

- What if we add types $\mathrm{Circ}(W_1, W_2)$ to the host language?
- **Requirement:** $\mathbf{C}(W_1, W_2)$ is an object of $\mathbf{H}$.
- Composition of circuits: host language program which combines programs from the circuit language.

Embedding as enrichment

- **Circuit language:** symmetric monoidal category $\mathbf{C}$
- Wire type W: object W of $\mathbf{C}$
- Program = Circuit of entry type W_1 and output type W_2:
  $$\mathbf{C}\text{-homomorphism } W_1 \to W_2$$

- **Host language:** cartesian closed category $\mathbf{H}$
- "General purpose" language
- Host type: object of $\mathbf{H}$

- What if we add types $\mathrm{Circ}(W_1, W_2)$ to the host language?
- **Requirement:** $\mathbf{C}(W_1, W_2)$ is an object of $\mathbf{H}$.
- Composition of circuits: host language program which combines programs from the circuit language.

Embedding a circuit language in the host language is an instance of enriched category theory

EWire: a language for embedded circuits

- Circuit language = first order typed language.
- Wire types, such as a type for qubits.
- Linear type system (think about no-cloning)

EWire: a language for embedded circuits

- Circuit language = first order typed language.
- Wire types, such as a type for qubits.
- Linear type system (think about no-cloning)

- Host language = higher order language (computational lambda-calculus, Haskell, Coq, ...)

EWire: a language for embedded circuits

- Circuit language = first order typed language.
- Wire types, such as a type for qubits.
- Linear type system (think about no-cloning)

- Host language = higher order language (computational lambda-calculus, Haskell, Coq, ...)
- Special host type $\mathrm{Circ}(W_1, W_2)$

EWire: a language for embedded circuits

- Circuit language = first order typed language.
- Wire types, such as a type for qubits.
- Linear type system (think about no-cloning)

- Host language = higher order language (computational lambda-calculus, Haskell, Coq, ...)
- Special host type $\mathrm{Circ}(W_1, W_2)$

QWire is an instance of EWire with:

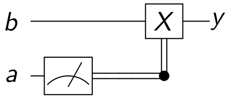
- one classical wire type, bit
- one circuit-only wire type, qubit
- basic gates such as $\mathrm{meas} \in \mathcal{G}(\mathrm{qubit}, \mathrm{bit})$ and $\mathrm{new} \in \mathcal{G}(\mathrm{bit}, \mathrm{qubit})$.

J. Paykin, R. Rand, and S. Zdancewic. QWIRE: a core language for quantum circuits. POPL'17.

J. Egger, R. E. Møgelberg, and A. Simpson. The enriched effect calculus: syntax and semantics. J. of Logic and Computation, 2012.

Where we are, sofar

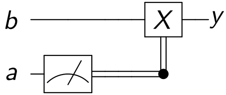
Boxing/Unboxing

$C \stackrel{\text{def}}{=}$ 

$$t \stackrel{\text{def}}{=} \textbf{box } (a, b) \Rightarrow C(a, b) : \mathrm{Circ}(\mathrm{qubit} \otimes \mathrm{qubit}, \mathrm{qubit})$$

Boxing/Unboxing

$C \stackrel{\text{def}}{=}$

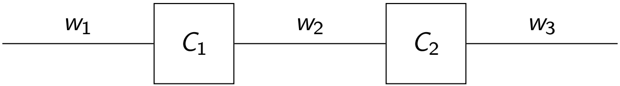


$$t \stackrel{\text{def}}{=} \textbf{box } (a, b) \Rightarrow C(a, b) : \mathrm{Circ}(\mathrm{qubit} \otimes \mathrm{qubit}, \mathrm{qubit})$$

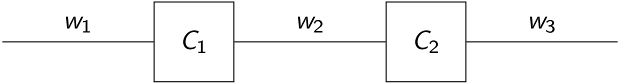
$$\frac{\Gamma \vdash t : \mathrm{Circ}(\mathrm{qubit} \otimes \mathrm{qubit}, \mathrm{qubit}) \quad \Omega \Longrightarrow p : \mathrm{qubit} \otimes \mathrm{qubit}}{\Gamma; \Omega \vdash \textbf{unbox } t\, p : W_2}$$

unbox $t\, w$ reduces to C

Composition of circuits



Composition of circuits

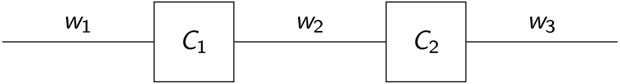


$$\text{comp} \stackrel{\text{def}}{=} \lambda(C_1, C_2).\ \textbf{box}\ w_1 \Rightarrow$$
$$\left(w_2 \leftarrow \textbf{unbox}\ C_1 w_1;\ w_3 \leftarrow \textbf{unbox}\ C_2 w_2;\ \textbf{output}\ w_3\right)$$

$$\text{comp} : \mathrm{Circ}(W_1, W_2) \times \mathrm{Circ}(W_2, W_3) \to \mathrm{Circ}(W_1, W_3)$$

W_i type of the wire w_i for $i \in \{1, 2, 3\}$

Composition of circuits



$$\begin{aligned}\mathsf{comp} \;&\stackrel{\text{def}}{=}\; \lambda(C_1, C_2).\ \textbf{box}\ w_1 \Rightarrow \\ &\big(w_2 \leftarrow \textbf{unbox}\ C_1 w_1;\, w_3 \leftarrow \textbf{unbox}\ C_2 w_2;\, \textbf{output}\ w_3\big)\end{aligned}$$

$$\mathsf{comp} \;:\; \mathrm{Circ}(W_1, W_2) \times \mathrm{Circ}(W_2, W_3) \to \mathrm{Circ}(W_1, W_3)$$

W_i type of the wire w_i for $i \in \{1, 2, 3\}$

The embedding of the circuit language in the host language is an instance of enriched category theory

Enriched category theory

- $\mathbf{H}$ category with finite products $\times$

Enriched category theory

- $\mathbf{H}$ category with finite products $\times$
- A category $\mathbf{C}$ enriched in $\mathbf{H}$ is given by a collection of objects together with
  - for each pair of objects A and B in $\mathbf{C}$, an object $\mathbf{C}(A, B)$ of $\mathbf{H}$;
  - for each object A of $\mathbf{C}$, a morphism $1 \to \mathbf{C}(A, A)$ in $\mathbf{H}$;
  - for objects A, B, C of $\mathbf{C}$, a morphism $\mathbf{C}(A, B) \times \mathbf{C}(B, C) \to \mathbf{C}(A, C)$ in $\mathbf{H}$

  such that composition satisfies the identity and unit laws.

Enriched category theory

- $\mathbf{H}$ category with finite products $\times$
- A category $\mathbf{C}$ enriched in $\mathbf{H}$ is given by a collection of objects together with
  - for each pair of objects A and B in $\mathbf{C}$, an object $\mathbf{C}(A, B)$ of $\mathbf{H}$;
  - for each object A of $\mathbf{C}$, a morphism $1 \to \mathbf{C}(A, A)$ in $\mathbf{H}$;
  - for objects A, B, C of $\mathbf{C}$, a morphism $\mathbf{C}(A, B) \times \mathbf{C}(B, C) \to \mathbf{C}(A, C)$ in $\mathbf{H}$

  such that composition satisfies the identity and unit laws.
- Example: a locally small category is **Set**-enriched category.

Max Kelly. Basic concepts of enriched category theory, volume 64. CUP Archive, 1982.

Where we are, sofar

How to flip a fair coin in quantum computing

- **Constraint: we work with a linear type system for circuits**
- Why? Because it is ***impossible*** to create an identical copy of an arbitrary unknown quantum state!

How to flip a fair coin in quantum computing

- **Constraint: we work with a linear type system for circuits**
- Why? Because it is ***impossible*** to create an identical copy of an arbitrary unknown quantum state!

$$\mathsf{flip} \stackrel{\mathsf{def}}{=} a \leftarrow \mathbf{gate}\ \mathrm{init}_0\ (); a' \leftarrow \mathbf{gate}\ \mathrm{H}\ a; b \leftarrow \mathbf{gate}\ \mathrm{meas}\ a'; \mathbf{output}\ b$$

Embedding circuits produce computational effects

$$\text{flip} \stackrel{\text{def}}{=} a \leftarrow \textbf{gate}\ \mathrm{init}_0\ (); a' \leftarrow \textbf{gate}\ \mathrm{H}\ a; b \leftarrow \textbf{gate}\ \mathrm{meas}\ a'; \textbf{output}\ b$$

Embedding circuits produce computational effects

$$\mathsf{flip} \stackrel{\mathsf{def}}{=} a \leftarrow \textbf{gate}\ \mathrm{init}_0\ (); a' \leftarrow \textbf{gate}\ \mathrm{H}\ a; b \leftarrow \textbf{gate}\ \mathrm{meas}\ a'; \textbf{output}\ b$$

Let's toss a coin!

$$\vdash \textbf{run}\big(\mathsf{flip}\big) : T(\mathsf{bool})$$

Embedding circuits produce computational effects

$$\text{flip} \stackrel{\text{def}}{=} a \leftarrow \textbf{gate}\ \mathrm{init}_0\ (); a' \leftarrow \textbf{gate}\ \mathrm{H}\ a; b \leftarrow \textbf{gate}\ \mathrm{meas}\ a'; \textbf{output}\ b$$

Let's toss a coin!

$$\vdash \textbf{run}\big(\text{flip}\big) : T(\text{bool})$$

- Probabilistic computational effects are required.
- Deterministic/pure programs = morphisms in $\mathbf{H}$
- Probabilistic/effectful programs = Kleisli morphisms $X \to T(Y)$ in $\mathbf{H}$

E. Moggi. Computational lambda-calculus and monads. LICS'89.

Where we are, sofar

How to use classical outcomes of circuits in the host language

- Classical wire types exist in both circuits and host terms.
- Example: bits.

How to use classical outcomes of circuits in the host language

- Classical wire types exist in both circuits and host terms.
- Example: bits.

- Lifting : classical wire type $\mapsto$ first-order host type

Dynamic lifting allows to use the classical outcomes of circuits as parameters in the host language

How to use classical outcomes of circuits in the host language

- Classical wire types exist in both circuits and host terms.
- Example: bits.

- Lifting : classical wire type $\mapsto$ first-order host type

Dynamic lifting allows to use the classical outcomes of circuits as parameters in the host language

- Copower: generalization of an n-fold coproduct.
- Copower $n \odot A = n$ fold coproduct $A + \cdots + A$ ($n \in \mathbb{N}$, $A \in \mathrm{Obj}(\mathbf{C})$).

To give a morphism $n \odot A \to B$
is to give a family of n morphisms $A \to B$.

- More generally: $\mathbf{C}(h \odot A, B) \cong \mathbf{H}(h, \mathbf{C}(A, B))$

B. Jacobs. On block structures in quantum computation. MFPS'13.

Where we are, sofar

Categorical models of EWire

A *categorical model of EWire* $(\mathbf{H}, \mathbf{H}_0, \mathbf{C}, T)$ is given by the following data:

Categorical models of EWire

A *categorical model of EWire* $(\mathbf{H}, \mathbf{H}_0, \mathbf{C}, T)$ is given by the following data:

(1) A cartesian closed category $\mathbf{H}$ with a strong monad T on $\mathbf{H}$.

Categorical models of EWire

A *categorical model of EWire* $(\mathbf{H}, \mathbf{H}_0, \mathbf{C}, T)$ is given by the following data:

(1) A cartesian closed category $\mathbf{H}$ with a strong monad T on $\mathbf{H}$.

(2) A small full subcategory $j : \mathbf{H}_0 \subseteq \mathbf{H}$ for classical wire types

Categorical models of EWire

A *categorical model of EWire* $(\mathbf{H}, \mathbf{H}_0, \mathbf{C}, T)$ is given by the following data:

(1) A cartesian closed category $\mathbf{H}$ with a strong monad T on $\mathbf{H}$.

(2) A small full subcategory $j : \mathbf{H}_0 \subseteq \mathbf{H}$ for classical wire types

(3) An $\mathbf{H}$-enriched symmetric monoidal category $(\mathbf{C}, \otimes, I)$.

Categorical models of EWire

A *categorical model of EWire* $(\mathbf{H}, \mathbf{H}_0, \mathbf{C}, T)$ is given by the following data:

(1) A cartesian closed category $\mathbf{H}$ with a strong monad T on $\mathbf{H}$.

(2) A small full subcategory $j : \mathbf{H}_0 \subseteq \mathbf{H}$ for classical wire types

(3) An $\mathbf{H}$-enriched symmetric monoidal category $(\mathbf{C}, \otimes, I)$.

(4) $\mathbf{C}$ has copowers by the objects of $\mathbf{H}_0$, inducing $J : \mathbf{H}_0 \to \mathbf{C}$ defined by $J(h) = h \odot I$.

Categorical models of EWire

A *categorical model of EWire* $(\mathbf{H}, \mathbf{H}_0, \mathbf{C}, T)$ is given by the following data:

(1) A cartesian closed category $\mathbf{H}$ with a strong monad T on $\mathbf{H}$.

(2) A small full subcategory $j : \mathbf{H}_0 \subseteq \mathbf{H}$ for classical wire types

(3) An $\mathbf{H}$-enriched symmetric monoidal category $(\mathbf{C}, \otimes, I)$.

(4) $\mathbf{C}$ has copowers by the objects of $\mathbf{H}_0$, inducing $J : \mathbf{H}_0 \to \mathbf{C}$ defined by $J(h) = h \odot I$.

(5) Copower-preserving functor $A \otimes - : \mathbf{C} \to \mathbf{C}$ for every $A \in \mathrm{Obj}(\mathbf{C})$

Categorical models of EWire

A *categorical model of EWire* $(\mathbf{H}, \mathbf{H}_0, \mathbf{C}, T)$ is given by the following data:

(1) A cartesian closed category $\mathbf{H}$ with a strong monad T on $\mathbf{H}$.

(2) A small full subcategory $j : \mathbf{H}_0 \subseteq \mathbf{H}$ for classical wire types

(3) An $\mathbf{H}$-enriched symmetric monoidal category $(\mathbf{C}, \otimes, I)$.

(4) $\mathbf{C}$ has copowers by the objects of $\mathbf{H}_0$, inducing $J : \mathbf{H}_0 \to \mathbf{C}$ defined by $J(h) = h \odot I$.

(5) Copower-preserving functor $A \otimes - : \mathbf{C} \to \mathbf{C}$ for every $A \in \mathrm{Obj}(\mathbf{C})$

(6) Enriched relative monad morphism $\mathrm{run}_h : \mathbf{C}(I, J(h)) \to T(j(h))$

T. Altenkirch, J. Chapman, and T. Uustalu. Monads need not be endofunctors. FOSSACS'10.

C. Berger, P.-A. Melliès, and Mark Weber. Monads with arities and their associated theories. J. Pure Appl. Algebra 2012.

Example: quantum circuits

- $\mathbf{C} \stackrel{\text{def}}{=} \mathbf{FdC^*\text{-}Alg}^{\mathbf{op}}_{\mathrm{CPU}}$: opposite category of the category of finite-dimensional C*-algebras and completely positive unital maps

Example: quantum circuits

- $\mathbf{C} \stackrel{\text{def}}{=} \mathbf{FdC^*\text{-}Alg}^{\mathbf{op}}_{\mathrm{CPU}}$: opposite category of the category of finite-dimensional C*-algebras and completely positive unital maps
- $\mathbf{H} \stackrel{\text{def}}{=} \mathbf{Set}$: cartesian closed category of sets and functions

Example: quantum circuits

- $\mathbf{C} \stackrel{\text{def}}{=} \mathbf{FdC^*\text{-}Alg}^{\mathbf{op}}_{\mathrm{CPU}}$: opposite category of the category of finite-dimensional C*-algebras and completely positive unital maps
- $\mathbf{H} \stackrel{\text{def}}{=} \mathbf{Set}$: cartesian closed category of sets and functions
- $\mathbf{H}_0 \stackrel{\text{def}}{=} \mathbb{N}$: skeleton of the category of finite sets and functions

Example: quantum circuits

- $\mathbf{C} \stackrel{\text{def}}{=} \mathbf{FdC^*\text{-}Alg}^{\mathbf{op}}_{\mathrm{CPU}}$: opposite category of the category of finite-dimensional C*-algebras and completely positive unital maps

- $\mathbf{H} \stackrel{\text{def}}{=} \mathbf{Set}$: cartesian closed category of sets and functions
- $\mathbf{H}_0 \stackrel{\text{def}}{=} \mathbb{N}$: skeleton of the category of finite sets and functions

- $T \stackrel{\text{def}}{=} \mathcal{D} : \mathbf{Set} \to \mathbf{Set}$, probability distribution monad

Example: quantum circuits

- $\mathbf{C} \stackrel{\text{def}}{=} \mathbf{FdC^*\text{-}Alg}^{\mathbf{op}}_{\mathrm{CPU}}$: opposite category of the category of finite-dimensional C*-algebras and completely positive unital maps

- $\mathbf{H} \stackrel{\text{def}}{=} \mathbf{Set}$: cartesian closed category of sets and functions
- $\mathbf{H}_0 \stackrel{\text{def}}{=} \mathbb{N}$: skeleton of the category of finite sets and functions

- $T \stackrel{\text{def}}{=} \mathcal{D} : \mathbf{Set} \to \mathbf{Set}$, probability distribution monad

$$\mathfrak{1} \stackrel{\text{def}}{=} \mathbb{C} \qquad \mathfrak{bit} \stackrel{\text{def}}{=} \mathbb{C} \oplus \mathbb{C} \qquad \mathfrak{qubit} \stackrel{\text{def}}{=} M_2$$

$$\mathfrak{u} \stackrel{\text{def}}{=} u^\dagger - u \text{ (for every unitary } u \in \mathcal{U})$$

$$\text{meas} : \mathbb{C} \oplus \mathbb{C} \to M_2 : (a, b) \mapsto \left(\begin{smallmatrix} a & 0 \\ 0 & b \end{smallmatrix}\right)$$

$$\text{new} : M_2 \to \mathbb{C} \oplus \mathbb{C} : \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \mapsto (a, b)$$

Example: 'generalized quantum circuits'

Example: 'generalized quantum circuits'

- Let's add sums!

$$\text{e.g. } \mathsf{in}_1 \in \mathcal{G}(W_1, W_1 \oplus W_2)$$

Example: 'generalized quantum circuits'

- Let's add sums!

$$\text{e.g. } \mathsf{in}_1 \in \mathcal{G}(W_1, W_1 \oplus W_2)$$

- Let's add recursive types!

$$\mathsf{fold}_{\mu X.W} \in \mathcal{G}(W[X \mapsto \mu X.W], \mu X.W)$$
$$\mathsf{unfold}_{\mu X.W} \in \mathcal{G}(\mu X.W, W[X \mapsto \mu X.W])$$

Example: 'generalized quantum circuits'

- Let's add sums!

$$\text{e.g. } \mathsf{in}_1 \in \mathcal{G}(W_1, W_1 \oplus W_2)$$

- Let's add recursive types!

$$\mathsf{fold}_{\mu X.W} \in \mathcal{G}(W[X \mapsto \mu X.W], \mu X.W)$$
$$\mathsf{unfold}_{\mu X.W} \in \mathcal{G}(\mu X.W, W[X \mapsto \mu X.W])$$

- $\mathbf{C} \stackrel{\text{def}}{=} \mathbf{W}^*\textbf{-Alg}^{\textbf{op}}_{\mathrm{CPSU}}$ ('domain-theoretic' C*-algebras)

Example: ‘generalized quantum circuits’

- Let’s add sums!

$$\text{e.g. } \mathsf{in}_1 \in \mathcal{G}(W_1, W_1 \oplus W_2)$$

- Let’s add recursive types!

$$\mathsf{fold}_{\mu X.W} \in \mathcal{G}(W[X \mapsto \mu X.W], \mu X.W)$$
$$\mathsf{unfold}_{\mu X.W} \in \mathcal{G}(\mu X.W, W[X \mapsto \mu X.W])$$

- $\mathbf{C} \stackrel{\text{def}}{=} \mathbf{W}^*\text{-}\mathbf{Alg}^{\mathbf{op}}_{\mathrm{CPSU}}$ (‘domain-theoretic’ C*-algebras)
- $\mathbf{H} \stackrel{\text{def}}{=} \mathbf{Dcpo}_{\perp}$: pointed dcpos and Scott-continuous maps

Example: 'generalized quantum circuits'

- Let's add sums!

$$\text{e.g. } \mathsf{in}_1 \in \mathcal{G}(W_1, W_1 \oplus W_2)$$

- Let's add recursive types!

$$\mathsf{fold}_{\mu X.W} \in \mathcal{G}(W[X \mapsto \mu X.W], \mu X.W)$$
$$\mathsf{unfold}_{\mu X.W} \in \mathcal{G}(\mu X.W, W[X \mapsto \mu X.W])$$

- $\mathbf{C} \stackrel{\text{def}}{=} \mathbf{W}^*\text{-}\mathbf{Alg}^{\mathbf{op}}_{\mathrm{CPSU}}$ ('domain-theoretic' C*-algebras)
- $\mathbf{H} \stackrel{\text{def}}{=} \mathbf{Dcpo}_{\perp}$: pointed dcpos and Scott-continuous maps
- $\mathbf{W}^*\text{-}\mathbf{Alg}^{\mathbf{op}}_{\mathrm{CPSU}}$ is $\mathbf{Dcpo}_{\perp}$-enriched and can be used to denote recursive types (via algebraic compactness).

Mathys Rennela. Towards a quantum domain theory. MFPS'13.

Example: ‘generalized quantum circuits’

- Let’s add sums!

$$\text{e.g. in}_1 \in \mathcal{G}(W_1, W_1 \oplus W_2)$$

- Let’s add recursive types!

$$\text{fold}_{\mu X.W} \in \mathcal{G}(W[X \mapsto \mu X.W], \mu X.W)$$
$$\text{unfold}_{\mu X.W} \in \mathcal{G}(\mu X.W, W[X \mapsto \mu X.W])$$

- $\mathbf{C} \overset{\text{def}}{=} \mathbf{W^*\text{-}Alg}^{\mathbf{op}}_{\mathrm{CPSU}}$ (‘domain-theoretic’ C*-algebras)
- $\mathbf{H} \overset{\text{def}}{=} \mathbf{Dcpo}_{\perp}$: pointed dcpos and Scott-continuous maps
- $\mathbf{W^*\text{-}Alg}^{\mathbf{op}}_{\mathrm{CPSU}}$ is $\mathbf{Dcpo}_{\perp}$-enriched and can be used to denote recursive types (via algebraic compactness).

Mathys Rennela. Towards a quantum domain theory. MFPS’13.

- What do we get? A semantics for the Quantum Fourier Transform!

Where we are, sofar

Conclusion

comp : $\mathrm{Circ}(W_1, W_2) \times \mathrm{Circ}(W_2, W_3) \to \mathrm{Circ}(W_1, W_3)$ in the host language

$\circ : \mathbf{C}(W_1, W_2) \times \mathbf{C}(W_2, W_3) \to \mathbf{C}(W_1, W_3)$ in $\mathbf{H}$

Conclusion

comp : $\mathrm{Circ}(W_1, W_2) \times \mathrm{Circ}(W_2, W_3) \to \mathrm{Circ}(W_1, W_3)$ in the host language

$\circ : \mathbf{C}(W_1, W_2) \times \mathbf{C}(W_2, W_3) \to \mathbf{C}(W_1, W_3)$ in $\mathbf{H}$

Embedding a first-order language for circuits into a general purpose host language is an instance of enriched category theory

Conclusion

comp : $\mathrm{Circ}(W_1, W_2) \times \mathrm{Circ}(W_2, W_3) \to \mathrm{Circ}(W_1, W_3)$ in the host language

$\circ : \mathbf{C}(W_1, W_2) \times \mathbf{C}(W_2, W_3) \to \mathbf{C}(W_1, W_3)$ in $\mathbf{H}$

Embedding a first-order language for circuits into a general purpose host language is an instance of enriched category theory

Next step: Implementation in Agda or Coq, with dependent types.

R. Rand, J. Paykin, S. Zdancewic. QWIRE Practice:Formal Verification of Quantum Circuits in Coq. QPL'17

Conclusion

comp : $\mathrm{Circ}(W_1, W_2) \times \mathrm{Circ}(W_2, W_3) \to \mathrm{Circ}(W_1, W_3)$ in the host language

$\circ : \mathbf{C}(W_1, W_2) \times \mathbf{C}(W_2, W_3) \to \mathbf{C}(W_1, W_3)$ in $\mathbf{H}$

Embedding a first-order language for circuits into a general purpose host language is an instance of enriched category theory

Next step: Implementation in Agda or Coq, with dependent types.

R. Rand, J. Paykin, S. Zdancewic. QWIRE Practice:Formal Verification of Quantum Circuits in Coq. QPL'17

THANK YOU!