Semantic Models of Quantum Programming Languages:

Recursion in Categorical Models

Michael Mislove

Department of Computer Science
Tulane University

Work Supported by US AFOSR

Joint work with Bert Lindenhovius and Vladimir Zamdzhiev

QuILT Workshop
University of New Orleans
March 25, 2019

Prototypical Quantum Computer

- Knill's QRAM model: A classical computer with a quantum co-processor

$$
\begin{tikzcd}
\text{Classical Computer} \arrow[rr, bend left, "\text{circuits}"] & & \text{Quantum Co-processor} \arrow[ll, bend left, "\text{measurements}"]
\end{tikzcd}
$$

- Circuit: sequence of unitary operators

How do we program such a device?

Some Basics

Logical Foundations

Predicate calculus:

Predicate symbols: $P, Q, R, \ldots$ each with a fixed arity

Functions symbols: $f, g, h, \ldots$ each with a fixed arity

Terms: $t ::= x \mid c \mid f(t_1, \ldots, t_n)$ where

x is a variable, c a nullary function, and f a function symbol with arity n.

Formulas: $\varphi ::= P(t_1, \ldots, t_n) \mid \perp \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \forall x\, \varphi \mid \exists x\, \varphi.$

- Sound and complete: $\vdash P$ iff $\models P$.

Some Basics

Logical Foundations

Predicate calculus:

Set Theory:

Standard ZF axioms, including *Axiom of Infinity:*
$\exists S.\emptyset \in S \ \wedge (\forall T) T \in S \implies T \cup \{T\} \in S.$

- Sound, *semantically complete*. But,

  Theorem:[Gödel] Any system powerful enough to do arithmetic is incapable of proving its own consistency.

Some Basics

Logical Foundations

Predicate calculus:

Set Theory:

Intuitionistic Logic:(Brouwer, 1907)

Does not include $A \vee \neg A$, or equivalently, $\neg\neg A \to A$.

Emphasis is on *proof*, not validity.

The logic for classical computation.

Some Basics

Logical Foundations

Predicate calculus:

Set Theory:

Intuitionistic Logic:(Brouwer, 1907)

Linear Logic:(Girard, 1987)

- Regards formulas as resources.
- Each hypothesis used once and only once.
- Derivations that obey this paradigm are called *linear*.
- Logic for reasoning about computing over quantum systems.

A Primer on the Basics

Computational Structures

Lambda calculus:(Church, 1934)

- *Untyped λ-calculus:* Terms: $t ::= x \mid \lambda x.t \mid t\,t$ where x is a variable, t a term.
- *Conversion Rules:*
  - $\lambda x.e \equiv_\alpha \lambda y.e[y/x]$ when $y \notin FV(e)$
  - $(\lambda x.e)(e') \to_\beta e[e'/x]$ if $FV(e') \cap BV(e) = \emptyset$.
- Turing complete:
  - Supports full recursion: $\mathrm{rec}\, x.t = t[\mathrm{rec}\, x.t/x]$.
  - Paradoxical combinator $Y := (\lambda x.x\,x)(\lambda x.x\,x)$ produces fixed point for any term

A Primer on the Basics

Computational Structures

Lambda calculus:(Church, 1934)

Untyped λ-calculus:

Simply typed lambda calculus, $\lambda^{\rightarrow}$:

Types: $\tau ::= 1 \mid Int \mid Bool \mid \tau \rightarrow \tau.$
Terms: $t ::= x \mid null \mid n \mid true \mid false \mid t\ t \mid \lambda^{\rightarrow} x : \tau . t.$
Typing Judgements:

$$\frac{}{\Gamma \vdash null{:}1} \qquad \frac{}{\Gamma \vdash n{:}Int} \qquad \frac{}{\Gamma \vdash true{:}Bool} \qquad \frac{}{\Gamma \vdash false{:}Bool}$$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x{:}\tau} \qquad \frac{\Gamma \vdash e{:}\ \tau \rightarrow \tau' \qquad \Gamma \vdash e'{:}\tau}{\Gamma \vdash e\ e'{:}\tau'} \qquad \frac{\Gamma, x{:}\tau \vdash e{:}\tau'}{\Gamma \vdash (\lambda^{\rightarrow} x{:}\tau . e){:}\tau \rightarrow \tau'}$$

A Primer on the Basics

Computational Structures

Lambda calculus:(Church, 1934)

Untyped λ-calculus:

Simply typed lambda calculus, $\lambda^{\rightarrow}$:

<u>Curry–Howard Correspondence:</u>

Intuitionistic Propositional Natural Deduction		Simply Typed Lambda Calculus
Propositions	$\longleftrightarrow$	*Types*
Proofs	$\longleftrightarrow$	*Terms*

Intuitionistic logic is the logic of classical computation

Computational Structures

Lambda calculus:(Church, 1934)

Untyped λ-calculus:

Simply typed lambda calculus, $\lambda^{\rightarrow}$:

Linear lambda calculus:

$$
\begin{array}{lll}
\text{Types} \quad A, B & ::= & 0 \mid A + B \mid \mathrm{I} \mid A \otimes B \mid A \multimap B \mid !A \\
\text{Terms} \quad M, N, P & ::= & x \mid c \mid \text{let } x = M \text{ in } N \mid \Box_A M \mid \text{left}_{A,B} M \mid \text{right}_{A,B} M \\
 & & \mid \text{case } M \text{ of } \{\text{left } x \to N \mid \text{right } y \to P\} \mid * \mid \langle M, N \rangle \\
 & & \mid \text{let } \langle x, y \rangle = M \text{ in } N \mid \lambda x^A.M \mid MN \mid \text{lift } M \mid \text{force } M
\end{array}
$$

lift $M (= !M)$ – Allows multiple instances of resource M.

force M – produces an instance of M' when $M = !M'$.

Semantic Models

A *category* $\mathcal{C}$ consists of:

(i) a family of objects $obj\ \mathcal{C}$, and

(ii) for each pair $A, B \in obj\ \mathcal{C}$ a family of *morphisms* $\mathcal{C}(A, B)$ satisfying:

$\circ\colon \mathcal{C}(B, C) \times \mathcal{C}(A, B) \to \mathcal{C}(A, C)$ is associative, and $(\forall A, B \in obj\ \mathcal{C})$

$1_A\colon A \to A$ is an identity with $1_B \circ f = f \circ 1_A$.

Example: **Set**, the category of sets and functions.

A category $\mathcal{C}$ is *Cartesian closed* if $\mathcal{C}$ has

finite products – $A \times B$,

a terminal object, $\perp$ satisfying $|\mathcal{C}(A, \perp)| = 1$ for all objects A,

and an internal hom $[A, B]$ satisfying $\mathcal{C}(A \times B, C) \simeq \mathcal{C}(A, [A, B])$.

For example, **Set** is Cartesian closed.

Semantic Models

Lambek's Theorem

There is a one-to-one correspondence between Cartesian closed categories and models of the typed lambda calculus.

For example, **Set** is a model for $\lambda^{\to}$

Scott's Corollary

There is a one-to-one correspondence between *reflexive objects* $[X \to X] \leftrightarrows X$ in Cartesian closed categories and models of the untyped lambda calculus

For example, $D_\infty \simeq [D_\infty \to D_\infty]$ is a model.

Fact

The only known *non-degenerate* reflexive objects in Cartesian closed categories are domains.

Semantic Models

Lambek's Theorem

There is a one-to-one correspondence between Cartesian closed categories and models of the typed lambda calculus.

For example, **Set** is a model for $\lambda^{\rightarrow}$

Scott's Corollary

There is a one-to-one correspondence between *reflexive objects* $[X \rightarrow X] \leftrightarrows X$ in Cartesian closed categories and models of the untyped lambda calculus

For example, $D_\infty \simeq [D_\infty \rightarrow D_\infty]$ is a model.

Fact

The only known *non-degenerate* reflexive objects in Cartesian closed categories are domains.

What does all this have to do with quantum computing?

Categorical Quantum Mechanics

The Hilbert space formalism provides the basic model of quantum mechanics.

The category **FdHilb** of finite dimensional Hilbert spaces and linear maps has a number of important properties:

FdHilb has a *symmetric tensor product:* $H \otimes K \simeq K \otimes H$.

FdHilb has a unit object $\mathbb{C}$: $\mathbb{C} \otimes H \simeq H$.

FdHilb has a 0-object, the degenerate Hilbert space: $0 \otimes H \simeq 0$.

FdHilb has biproducts: $H \oplus K$.

FdHilb is *dagger compact closed:* there is an involution $H \mapsto H^*$ that extends to $f \mapsto f_*$ on linear maps satisfying $H^{**} \simeq H$, $f_{**} = f$ and $(f_*)^\dagger = (f^\dagger)_*$.

Categorical Quantum Mechanics

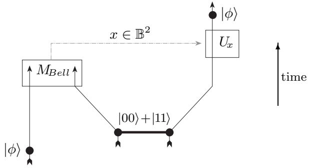
The Hilbert space formalism provides the basic model of quantum mechanics.

Abramsky & Coecke: Dagger compact closed (symmetric monoidal) categories with biproducts model finitary quantum mechanics.

These categories are a model of (multiplicative) linear logic.

They are an abstract setting in which to reason precisely about quantum protocols, such as teleportation and entanglement swapping.

There also is a diagrammatic calculus for reasoning in these categories:



Prototypical Quantum Computer

- Returning to Knill's QRAM model:

$$
\begin{tikzcd}
\text{Classical Computer} \arrow[rr, bend left, "\text{circuits}"] & & \text{Quantum Co-processor} \arrow[ll, bend left, "\text{measurements}"]
\end{tikzcd}
$$

- Circuit: sequence of unitary operators

Prototypical Quantum Computer

- A *quantum programming language* is a classical functional language together with a linear language of *quantum circuits*:

$$\begin{tikzcd} \mathit{Functional\ Language} \arrow[rr, bend left] & & \mathit{Linear\ language} \arrow[ll, bend left] \end{tikzcd}$$

- We elide measurements and focus on a classical functional language for *constructing circuits* and a linear language for *modeling them* as linear morphisms.

- We model *circuit description languages* using Linear / Nonlinear Models

Linear/Non-Linear models

A Linear/Non-Linear (LNL) model is given by the following data:

- A cartesian closed category **C**.
- A symmetric monoidal closed category **L**.
- A symmetric monoidal adjunction:

$$
\begin{tikzcd}
\mathbf{C} \arrow[rr, bend left, "F"] \arrow[rr, phantom, "\perp"] & & \mathbf{L} \arrow[ll, bend left, "G"]
\end{tikzcd}
$$

$$
\begin{array}{rcl}
F(X \times Y) & \simeq & F(X) \otimes F(Y) \\
F(X + Y) & \simeq & F(X) + F(Y) \\
F(\emptyset) = 0 & & F(1) = I \\
F \circ G & = & ! \text{ -- the lift comonad}
\end{array}
$$

An LNL model is a model of Intuitionistic Linear Logic.[1]

[1] Nick Benton. *A mixed linear and non-linear logic: Proofs, terms and models.* CSL'94

Proto-Quipper-M (*Rios and Selinger*)

Types	A, B	$::=$	$\alpha \mid 0 \mid A + B \mid \mathrm{I} \mid A \otimes B \mid A \multimap B \mid !A \mid \mathrm{Circ}(T, U)$
Intuitionistic types	P, R	$::=$	$0 \mid P + R \mid \mathrm{I} \mid P \otimes R \mid !A \mid \mathrm{Circ}(T, U)$
M-types	T, U	$::=$	$\alpha \mid \mathrm{I} \mid T \otimes U$

$$
\begin{array}{lll}
\text{Terms} \quad M, N & ::= & x \mid \ell \mid c \mid \text{let } x = M \text{ in } N \\
& & \mid \Box_A M \mid \text{left}_{A,B} M \mid \text{right}_{A,B} M \mid \text{case } M \text{ of } \{\text{left } x \to N \mid \text{right } y \to P\} \\
& & \mid * \mid M; N \mid \langle M, N \rangle \mid \text{let } \langle x, y \rangle = M \text{ in } N \mid \lambda x^A.M \mid MN \\
& & \mid \text{lift } M \mid \text{force } M \mid \mathbf{box}_T M \mid \text{apply}(M, N) \mid (\vec{\ell}, C, \vec{\ell'})
\end{array}
$$

- All types other than Intuitionistic types are *linear*
- M-types: morphisms from a symmetric monoidal category such as M = **FdHilb**
- Only use one (combined) form of type judgement

Example

Assume $H : Q \multimap Q$ is a constant representing the Hadamard gate.

Example

two-hadamard : $\mathrm{Circ}(Q, Q)$
two-hadamard $\equiv \mathrm{box}_Q$ lift $\lambda q^Q.HHq$

This program creates a completed circuit consisting of two H gates. The term is intuitionistic (can be copied, deleted).

Example

Shor's algorithm for integer factorization may be seen as an infinite family of quantum circuits – each circuit is a procedure for factoring an *n*-bit integer, for a fixed *n*.

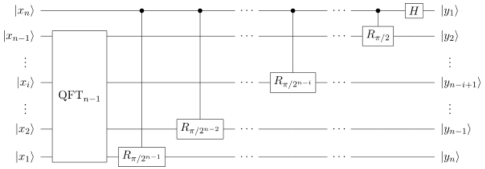


Figure: Quantum Fourier Transform on *n* qubits (subroutine in Shor's algorithm).[2]

Proto-Quipper-M is used to describe *families* of morphisms in an arbitrary, but fixed, symmetric monoidal category, **M**.

[2]Figure source: `https://commons.wikimedia.org/w/index.php?curid=14545612`

Concrete model of Proto-Quipper-M

A simple Proto-Quipper-M model is given by the LNL model:

$$\begin{tikzcd} \mathbf{Set} \arrow[r, bend left, "- \odot I"] \arrow[r, phantom, "\perp"] & \overline{\mathbf{M}} \arrow[l, bend left, "\overline{\mathbf{M}}(I, -)"] \end{tikzcd}$$

where $\overline{\mathbf{M}} = [\mathbf{M}^{\mathrm{op}}, \mathbf{Set}]$ is a closed, product complete category containing given SMC $\mathbf{M}$

Theorem (Rios & Selinger)

The simple categorical model of Proto-Quipper-M is type-safe, sound, and computationally adequate

Concrete model of Proto-Quipper-M

There are two semantic models:

- For all types, $[\![P]\!] \in \overline{\mathbf{M}}$
- For intuitionistic types, also have $(\!|P|\!) \in \mathbf{Set}$

Theorem

For any intuitionistic type P, there exists a canonical isomorphism $\alpha_P : [\![P]\!] \to F(\!|P|\!)$.

So we can define copy and discard morphisms for each intuitionistic type P:

$$\Delta_P := [\![P]\!] \xrightarrow{\alpha_P} F(\!|P|\!) \xrightarrow{F\langle id, id\rangle} F((\!|P|\!) \times (\!|P|\!)) \xrightarrow{\cong} F(\!|P|\!) \otimes F(\!|P|\!) \xrightarrow{\alpha_P^{-1} \otimes \alpha_P^{-1}} [\![P]\!] \otimes [\![P]\!]$$

$$\diamond_P := [\![P]\!] \xrightarrow{\alpha_P} F(\!|P|\!) \xrightarrow{F1} F1 \xrightarrow{\cong} I$$

where $FX = X \odot I$

Our Work: Adding Recursion

- Focus on adding recursive *types*.
  - Term recursion follows from recursive types.
- Main difficulty is with the categorical model.
- How can we copy/discard intuitionistic recursive types?
  - A list of qubits should be *linear* – cannot copy/discard.
  - A list of natural numbers should be *intuitionistic* – can *implicitly* copy/discard.
- For the rest of the talk we focus on the linear/non-linear type structure.
- How do we design a linear/non-linear FPC [3] ?

[3] FPC is an intuitionistic Fixed Point Calculus studied by Fiore and Plotkin.

Adding Recursive Datatypes

Type Variables	X, Y		
Types	A, B	$::=$	$X \mid \alpha \mid A + B \mid \mathrm{I} \mid A \otimes B \mid A \multimap B \mid !A \mid \mathrm{Circ}(T, U)$
			$\mid \mu X.A$
Intuitionistic types	P, R	$::=$	$X \mid P + R \mid \mathrm{I} \mid P \otimes R \mid !A \mid \mathrm{Circ}(T, U) \mid \mu X.P$
M-types	T, U	$::=$	$\alpha \mid \mathrm{I} \mid T \otimes U$

These types are accompanied by some formation rules, which we omit.

Some useful recursive datatypes

Example

$\mathsf{Nat} \equiv \mu X.I + X \qquad$ (intuitionistic)

Example

$\mathsf{List\ Nat} \equiv \mu X.I + X \otimes \mathsf{Nat} \qquad$ (intuitionistic)

Example

$\mathsf{List\ Qubit} \equiv \mu X.I + X \otimes \mathsf{Qubit} \qquad$ (linear)

Example

$\mathsf{Stream\ Qubit} \equiv \mu X.I \multimap (X \otimes \mathsf{Qubit}) \qquad$ (linear)

Example

$\mathsf{Stream\ Nat} \equiv \mu X.!(X \otimes \mathsf{Nat}) \qquad$ (intuitionistic)

A CPO-enriched model

CPO – ω-complete partial orders and monotone maps preserving suprema of ω-chains.

If $\mathcal{C}$ is Cartesian closed (or even monoidal closed), then the category $\mathcal{B}$ is $\mathbb{C}$-enriched if:

obj $\mathcal{B}$ is a set

For each $B, B' \in \textit{obj } \mathcal{B}$, the family $\mathcal{B}(B, B') \in \textit{obj } \mathcal{C}$.

The relevant morphisms – composition, etc., in $\mathcal{B}$ are $\mathcal{C}$-morphisms:

E.g., $\circ\colon \mathcal{B}(B', B") \times \mathcal{B}(B, B') \to \mathcal{B}(B, B")$ is a $\mathcal{C}$-morphism.

Examples: 1) Since **Set** is Cartesian closed, every *concrete* category is **Set**-enriched.

2) **CPO** is Cartesian closed, so **CPO** is self-enriched.

3) $\mathbf{CPO}_{\perp!}$ is **CPO**-enriched, where $\mathbf{CPO}_{\perp!}$ is the subcategory of **CPO** where every object has a least element ($\perp$) and morphisms preserve $\perp$.

A **CPO**-enriched model

CPO – ω-complete partial orders and monotone maps preserving suprema of ω-chains.

A **CPO**–enriched LNL model includes:

1. A **CPO**-symmetric monoidal closed category $\mathcal{L}$ with finite **CPO**-coproducts.
2. A **CPO**-symmetric monoidal adjunction:

$$
\begin{tikzcd}
\mathbf{CPO} \arrow[rr, bend left, "F = - \odot I"] \arrow[rr, phantom, "\perp"] & & \mathcal{L} \arrow[ll, bend left, "\mathcal{L}(I, -)"]
\end{tikzcd},
$$

3. The category $\mathcal{L}$ is $\mathbf{CPO}_{\perp !}$-enriched and has ω-colimits

Example: $\mathcal{L} = \mathbf{CPO}_{\perp !}$ is the simplest example: $I = \{\perp\}_\perp$ and $F(D) \simeq D_\perp$ for all **CPO**s D.

A **CPO**-enriched model

CPO – ω-complete partial orders and monotone maps preserving suprema of ω-chains.

A **CPO**–enriched LNL model includes:

1. A **CPO**-symmetric monoidal closed category $\mathcal{L}$ with finite **CPO**-coproducts.
2. A **CPO**-symmetric monoidal adjunction:

$$\begin{tikzcd} \mathbf{CPO} \arrow[rr, bend left, "F = - \odot I"] \arrow[rr, phantom, "\perp"] && \mathcal{L}, \arrow[ll, bend left, "\mathcal{L}(I, -)"] \end{tikzcd}$$

3. The category $\mathcal{L}$ is $\mathbf{CPO}_{\perp!}$-enriched and has ω-colimits

Remark

1. and 3. imply $\mathcal{L}$ has a zero object and we can solve recursive domain equations.

Interpretation of recursive types

Interpreting recursive types requires finding initial (final) (co)algebras of certain **CPO**-endofunctors.

If $T: \mathcal{C} \to \mathcal{C}$ is an endofunctor, then a T-algebra is an object $C \in obj\ \mathbf{C}$ and a map $\phi_C: TC \to C$.

C is an *initial T-algebra* if for any T-algebra $\phi_D: TD \to D$, there is a unique morphism $f: C \to D$ satisfying $\phi_D \circ Tf = f \circ \phi_C$.

Example: If $T: \mathbf{Set} \to \mathbf{Set}$ is $T(S) = S \cup \{S\}$, then $\mathbb{N}$ is the initial T-algebra.

Dually, a final T-coalgebra is an object C and a morphism $\psi_D: D \to TD$.

D is a *final T-coalgebra* if any other T-coalgebra $\psi_E: E \to TE$ admits a morphism $g: E \to D$ with $\psi_D \circ g = Tg \circ \psi_E$.

Example: If $T: \mathbf{Set} \to \mathbf{Set}$ is $T(S) = \{\emptyset\} \cup S$, then $\{\emptyset\}$ is the final T-coalgebra.

Interpretation of recursive types

Interpreting recursive types requires finding initial (final) (co)algebras of certain **CPO**-endofunctors.

Lemma (Adámek)

Let $\mathbf{C}$ *be a category with an initial object* $\emptyset$ *and let* $T : \mathbf{C} \to \mathbf{C}$ *be an endofunctor. Assume further that the following* ω*-diagram*

$$\emptyset \xrightarrow{\iota} T\emptyset \xrightarrow{T\iota} T^2\emptyset \xrightarrow{T^2\iota} \cdots$$

has a colimit and T *preserves it. Then, the induced isomorphism is the initial* T*-algebra.*

Corollary

In a symmetric monoidal closed category with finite coproducts and ω*-colimits, any endofunctor composed from constants,* $\otimes$ *and* $+$ *has an initial algebra.*

Embedding-projection pairs

Problem: How do we interpret recursive types which also contain ! and $\multimap$?

The problem for $\langle A, B\rangle \mapsto A \multimap B$ is that it is covariant in B and contravariant in A.

Textbook Solution: CPO-enrichment and embedding-projection pairs.

Definition

Given a **CPO**-enriched category $\mathbf{C}$, an *embedding-projection* pair is a pair of morphisms $e : A \to B$ and $p : B \to A$, such that $p \circ e = \mathrm{id}$ and $e \circ p \leq \mathrm{id}$.

Theorem

If e is an embedding, then it has a unique projection, which we denote e^*.

Definition

The subcategory of $\mathbf{C}$ with the same objects, but whose morphisms are embeddings is denoted $\mathbf{C}_e$.

Interpretation of recursive types (contd.)

Theorem (Smyth and Plotkin)

*If $T : \mathbf{C} \to \mathbf{D}$ is a **CPO**-enriched functor and $\mathbf{C}$ has ω-colimits, then T preserves ω-colimits of embeddings. In other words, the restriction $T_e : \mathbf{C}_e \to \mathbf{D}_e$ is ω-continuous.*

Theorem

*In our categorical model, any **CPO**-endofunctor $T : \mathcal{L} \to \mathcal{L}$ has an initial T-algebra, whose inverse is a final T-coalgebra.*

Remark

The above theorem follows directly from results in Fiore's PhD thesis.

Main Lemma

We define $\mathbf{CPO}_{pe}$ to be the full-on-objects subcategory of $\mathbf{CPO}$ whose morphisms f are those satisfying $F(f) \in \mathcal{L}_e$. We call such f *pre-embeddings*.

Then there are two semantic models:

- For all types, $[\![\Theta \vdash P]\!] \in \mathcal{L}$
- For intuitionistic types, also have $(\!|\Theta \vdash P|\!) \in \mathbf{CPO}_{pe}$

There exists a natural isomorphism

$$\alpha_{\Theta \vdash P} : [\![\Theta \vdash P]\!]_s \circ F^{\times n} \Longrightarrow F \circ (\!|\Theta \vdash P|\!)$$

Diagrammatically:

$$\begin{tikzcd}
\mathcal{L}_e^{|\Theta|} \arrow[rr, "{[\![\Theta \vdash P]\!]_e}"] & & \mathcal{L}_e \\
& \arrow[r, phantom, "\overset{\alpha}{\Longrightarrow}"] & \\
\mathbf{CPO}_{pe}^{|\Theta|} \arrow[uu, "F^{\times|\Theta|}"] \arrow[rr, "{(\!|\Theta \vdash P|\!)}"'] & & \mathbf{CPO}_{pe} \arrow[uu, "F"']
\end{tikzcd}$$

Copy and Discard

Let P be an intuitionistic object and $\alpha : P \to F(X)$ an isomorphism.

We can define three maps:

$$\textit{Discard:}\quad \diamond_P^\alpha := P \xrightarrow{\alpha} F(X) \xrightarrow{F(1_X)} F(1) \xrightarrow{\cong} I;$$

$$\textit{Copy:}\quad \Delta_P^\alpha := P \xrightarrow{\alpha} F(X) \xrightarrow{F(\langle \mathrm{id}, \mathrm{id}\rangle)} F(X \times X) \xrightarrow{\cong} F(X) \otimes F(X) \xrightarrow{\alpha^{-1} \otimes \alpha^{-1}} P \otimes P;$$

$$\textit{Lift:}\quad \mathbf{lift}_P^\alpha := P \xrightarrow{\alpha} F(X) \xrightarrow{F(\eta_X)} !F(X) \xrightarrow{!(\alpha^{-1})} !P.$$

Given two intuitionistic objects P_1 and P_2, a morphism $f : P_1 \to P_2$ is called *intuitionistic*, if there exists a morphism $f' \in \mathbf{CPO}(X, Y)$ and two isomorphisms α and β, such that $f = P_1 \xrightarrow{\alpha} F(X) \xrightarrow{F(f')} F(Y) \xrightarrow{\beta} P_2$.

If $f : P_1 \to P_2$ is intuitionistic, then:

- $\diamond_{P_2} \circ f = \diamond_{P_1}$;
- $\Delta_{P_2} \circ f = (f \otimes f) \circ \Delta_{P_1}$;
- $\mathbf{lift}_{P_2} \circ f = !f \circ \mathbf{lift}_{P_1}$.

Thank You!

Questions??

Syntax

$$\frac{}{\Phi, x : A; \emptyset \vdash x : A}\ (\text{var}) \qquad \frac{}{\Phi; \ell : \alpha \vdash \ell : \alpha}\ (\text{label}) \qquad \frac{}{\Phi; \emptyset \vdash c : A_c}\ (\text{const}) \qquad \frac{\Phi, \Gamma_1; Q_1 \vdash m : A \qquad \Phi, \Gamma_2, x : A; Q_2 \vdash n : B}{\Phi, \Gamma_1, \Gamma_2; Q_1, Q_2 \vdash \text{let } x = m \text{ in } n : B}\ (\text{let})$$

$$\frac{\Gamma; Q \vdash m : 0}{\Gamma; Q \vdash \Box_C m : C}\ (\text{initial}) \qquad \frac{\Gamma; Q \vdash m : A}{\Gamma; Q \vdash \text{left}_{A,B} m : A + B}\ (\text{left}) \qquad \frac{\Gamma; Q \vdash m : B}{\Gamma; Q \vdash \text{right}_{A,B} m : A + B}\ (\text{right}) \qquad \frac{}{\Phi; \emptyset \vdash * : I}\ (^*)$$

$$\frac{\Phi, \Gamma_1; Q_1 \vdash m : A + B \qquad \Phi, \Gamma_2, x : A; Q_2 \vdash n : C \qquad \Phi, \Gamma_2, y : B; Q_2 \vdash p : C}{\Phi, \Gamma_1, \Gamma_2; Q_1, Q_2 \vdash \text{case } m \text{ of } \{\text{left } x \to n \mid \text{right } y \to p\} : C}\ (\text{case}) \qquad \frac{\Phi, \Gamma_1; Q_1 \vdash m : I \qquad \Phi, \Gamma_2; Q_2 \vdash n : C}{\Phi, \Gamma_1, \Gamma_2; Q_1, Q_2 \vdash m; n : C}\ (\text{seq})$$

$$\frac{\Phi, \Gamma_1; Q_1 \vdash m : A \qquad \Phi, \Gamma_2; Q_2 \vdash n : B}{\Phi, \Gamma_1, \Gamma_2; Q_1, Q_2 \vdash \langle m, n \rangle : A \otimes B}\ (\text{pair}) \qquad \frac{\Phi, \Gamma_1; Q_1 \vdash m : A \otimes B \qquad \Phi, \Gamma_2, x : A, y : B; Q_2 \vdash n : C}{\Phi, \Gamma_1, \Gamma_2; Q_1, Q_2 \vdash \text{let } \langle x, y \rangle = m \text{ in } n : C}\ (\text{let-pair})$$

$$\frac{\Gamma, x : A; Q \vdash m : B}{\Gamma; Q \vdash \lambda x^A . m : A \multimap B}\ (\text{abs}) \qquad \frac{\Phi, \Gamma_1; Q_1 \vdash m : A \multimap B \qquad \Phi, \Gamma_2; Q_2 \vdash n : A}{\Phi, \Gamma_1, \Gamma_2; Q_1, Q_2 \vdash mn : B}\ (\text{app}) \qquad \frac{\Phi; \emptyset \vdash m : A}{\Phi; \emptyset \vdash \text{lift } m : !A}\ (\text{lift}) \qquad \frac{\Gamma; Q \vdash m : !A}{\Gamma; Q \vdash \text{force } m : A}\ (\text{force})$$

$$\frac{\Gamma; Q \vdash m : !(T \multimap U)}{\Gamma; Q \vdash \text{box}_T m : \text{Diag}(T, U)}\ (\text{box}) \qquad \frac{\Phi, \Gamma_1; Q_1 \vdash m : \text{Diag}(T, U) \quad \Phi, \Gamma_2; Q_2 \vdash n : T}{\Phi, \Gamma_1, \Gamma_2; Q_1, Q_2 \vdash \text{apply}(m, n) : U}\ (\text{apply}) \qquad \frac{\emptyset; Q \vdash \vec{\ell} : T \quad \emptyset; Q' \vdash \vec{\ell}' : U \quad S \in \mathbf{M}_L(Q, Q')}{\Phi; \emptyset \vdash (\vec{\ell}, S, \vec{\ell}') : \text{Diag}(T, U)}\ (\text{diag})$$

Operational semantics

$$\frac{(S, m) \Downarrow (S', v) \quad (S', n) \Downarrow (S'', v')}{(S, \langle m, n\rangle) \Downarrow (S'', \langle v, v'\rangle)} \qquad \frac{(S, m) \Downarrow (S', \langle v, v'\rangle) \quad (S', n[v / x, v' / y]) \Downarrow (S'', w)}{(S, \mathrm{let}\ \langle x, y\rangle = m\ \mathrm{in}\ n) \Downarrow (S'', w)}$$

$$\frac{}{(S, \mathrm{lift}\ m) \Downarrow (S, \mathrm{lift}\ m)} \qquad \frac{(S, m) \Downarrow (S', \mathrm{lift}\ m') \quad (S', m') \Downarrow (S'', v)}{(S, \mathrm{force}\ m) \Downarrow (S'', v)}$$

$$\frac{(S, m) \Downarrow (S', \mathrm{lift}\ n) \quad \mathrm{freshlabels}(T) = (Q, \vec{\ell}) \quad (\mathrm{id}_Q, n\vec{\ell}) \Downarrow (D, \vec{\ell}')}{(S, \mathrm{box}_T m) \Downarrow (S', (\vec{\ell}, D, \vec{\ell}'))}$$

$$\frac{(S, m) \Downarrow (S', (\vec{\ell}, D, \vec{\ell}')) \quad (S', n) \Downarrow (S'', \vec{k}) \quad \mathrm{append}(S'', \vec{k}, \vec{\ell}, D, \vec{\ell}') = (S''', \vec{k}')}{(S, \mathrm{apply}(m, n)) \Downarrow (S''', \vec{k}')}$$

$$\frac{(S, m) \Downarrow (S', (\vec{\ell}, D, \vec{\ell}')) \quad (S', n) \Downarrow (S'', \vec{k}) \quad \mathrm{append}(S'', \vec{k}, \vec{\ell}, D, \vec{\ell}')\ \mathrm{undefined}}{(S, \mathrm{apply}(m, n)) \Downarrow \mathrm{Error}} \qquad \frac{}{(S, (\vec{\ell}, D, \vec{\ell}')) \Downarrow (S, (\vec{\ell}, D, \vec{\ell}'))}$$