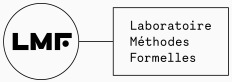
The Semantics of Effects: Centrality, Quantum Control and Reversible Recursion

Louis LEMONNIER

Supervised by: Pablo ARRIGHI, Benoît VALIRON, Vladimir ZAMDZHIEV



PhD Defence. 19th June 2024

Introduction: Languages and Types

λ-calculus:

- Form a function: $f = \lambda x.x + 2 \equiv x \mapsto x + 2$
- Apply the function: $f(3)$, or $f\,3$.

Introduction: Languages and Types

λ-calculus:

- Form a function: $f = \lambda x.x + 2 \equiv x \mapsto x + 2$
- Apply the function: $f\,(3)$, or $f\,3$.

A type A is a specification of a term t, we write $t : A$.

For example:

$$\texttt{True} : \texttt{Bool} \qquad 2 : \texttt{Nat} \qquad \lambda x.x + 2 : \texttt{Nat} \to \texttt{Nat}$$

Introduction: Languages and Types

λ-calculus:

- Form a function: $f = \lambda x.x + 2 \equiv x \mapsto x + 2$
- Apply the function: $f\,(3)$, or $f\,3$.

A type A is a specification of a term t, we write $t : A$.

For example:

$$\texttt{True} : \texttt{Bool} \qquad 2 : \texttt{Nat} \qquad \lambda x.x + 2 : \texttt{Nat} \to \texttt{Nat}$$

We also have variables: $\texttt{if}\ x\ \texttt{then}\ y\ \texttt{else}\ y + 2$

Example of judgement:

$$x : \texttt{Bool}, y : \texttt{Nat} \vdash \texttt{if}\ x\ \texttt{then}\ y\ \texttt{else}\ y + 2 : \texttt{Nat}$$

Introduction: Type constructors and destructors

Types	Constructor	Destructor
$\mathtt{I}$	$*$	$\mathtt{let}\, * = t\, \mathtt{in}\, t'$

Introduction: Type constructors and destructors

Types	Constructor	Destructor
$\mathtt{I}$	$*$	$\mathtt{let}\, * = t\, \mathtt{in}\, t'$
$A \to B$	$\lambda x.t$	tt'

Introduction: Type constructors and destructors

Types	**Constructor**	**Destructor**
$\mathtt{I}$	$*$	$\mathtt{let}\, * = t\ \mathtt{in}\ t'$
$A \to B$	$\lambda x.t$	tt'
$A \otimes B$	$\langle t_1, t_2 \rangle$	$\mathtt{let}\, \langle x, y \rangle = t\ \mathtt{in}\ t'$

Introduction: Type constructors and destructors

Types	**Constructor**	**Destructor**
$\mathtt{I}$	$*$	$\mathtt{let}\, * = t\, \mathtt{in}\, t'$
$A \to B$	$\lambda x.t$	tt'
$A \otimes B$	$\langle t_1, t_2 \rangle$	$\mathtt{let}\, \langle x, y \rangle = t\, \mathtt{in}\, t'$
$\mathtt{Nat}$	$\mathtt{zero},\ \mathtt{S}\ t$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{zero} \mapsto t_1 \mid \mathtt{S}\ \mapsto t_2\}$

Introduction: Type constructors and destructors

Types	**Constructor**	**Destructor**
$\mathtt{I}$	$*$	$\mathtt{let}\ * = t\ \mathtt{in}\ t'$
$A \to B$	$\lambda x.t$	tt'
$A \otimes B$	$\langle t_1, t_2 \rangle$	$\mathtt{let}\ \langle x, y \rangle = t\ \mathtt{in}\ t'$
$\mathtt{Nat}$	$\mathtt{zero},\ \mathtt{S}\ t$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{zero} \mapsto t_1 \mid \mathtt{S}\ \mapsto t_2\}$
$\mathtt{List}(A)$	$\mathtt{Empty},\ \mathtt{Cons}\ \langle h, t \rangle$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{Empty} \mapsto t_1 \mid \mathtt{Cons} \mapsto t_2\}$

Introduction: Type constructors and destructors

Types	**Constructor**	**Destructor**
$\mathtt{I}$	$*$	$\mathtt{let}\ * = t\ \mathtt{in}\ t'$
$A \to B$	$\lambda x.t$	tt'
$A \otimes B$	$\langle t_1, t_2 \rangle$	$\mathtt{let}\ \langle x, y \rangle = t\ \mathtt{in}\ t'$
$\mathtt{Nat}$	$\mathtt{zero},\ \mathtt{S}\ t$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{zero} \mapsto t_1 \mid \mathtt{S}\ \mapsto t_2\}$
$\mathtt{List}(A)$	$\mathtt{Empty},\ \mathtt{Cons}\ \langle h, t \rangle$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{Empty} \mapsto t_1 \mid \mathtt{Cons} \mapsto t_2\}$
$A \oplus B$	$\mathtt{left}\ t,\ \mathtt{right}\ t$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{left}\ \mapsto t_1 \mid \mathtt{right}\ \mapsto t_2\}$

Introduction: Type constructors and destructors

Types	**Constructor**	**Destructor**
$\mathtt{I}$	$*$	$\mathtt{let}\ * = t\ \mathtt{in}\ t'$
$A \to B$	$\lambda x.t$	tt'
$A \otimes B$	$\langle t_1, t_2 \rangle$	$\mathtt{let}\ \langle x, y \rangle = t\ \mathtt{in}\ t'$
$\mathtt{Nat}$	$\mathtt{zero},\ \mathtt{S}\ t$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{zero} \mapsto t_1 \mid \mathtt{S}\ \mapsto t_2\}$
$\mathtt{List}(A)$	$\mathtt{Empty},\ \mathtt{Cons}\ \langle h, t \rangle$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{Empty} \mapsto t_1 \mid \mathtt{Cons} \mapsto t_2\}$
$A \oplus B$	$\mathtt{left}\ t,\ \mathtt{right}\ t$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{left}\ \mapsto t_1 \mid \mathtt{right}\ \mapsto t_2\}$

$$\mathtt{Nat} \cong \mathtt{I} \oplus \mathtt{Nat}$$

Introduction: Type constructors and destructors

Types	**Constructor**	**Destructor**
$\mathtt{I}$	$*$	$\mathtt{let}\ * = t\ \mathtt{in}\ t'$
$A \to B$	$\lambda x.t$	tt'
$A \otimes B$	$\langle t_1, t_2 \rangle$	$\mathtt{let}\ \langle x, y \rangle = t\ \mathtt{in}\ t'$
$\mathtt{Nat}$	$\mathtt{zero},\ \mathtt{S}\ t$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{zero} \mapsto t_1 \mid \mathtt{S}\ \mapsto t_2\}$
$\mathtt{List}(A)$	$\mathtt{Empty},\ \mathtt{Cons}\ \langle h, t \rangle$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{Empty} \mapsto t_1 \mid \mathtt{Cons} \mapsto t_2\}$
$A \oplus B$	$\mathtt{left}\ t,\ \mathtt{right}\ t$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{left}\ \mapsto t_1 \mid \mathtt{right}\ \mapsto t_2\}$

$$\mathtt{Nat} \cong \mathtt{I} \oplus \mathtt{Nat} \qquad \mathtt{List}(A) \cong \mathtt{I} \oplus (A \otimes \mathtt{List}(A))$$

Introduction: Type constructors and destructors

Types	**Constructor**	**Destructor**
$\mathtt{I}$	$*$	$\mathtt{let}\ * = t\ \mathtt{in}\ t'$
$A \to B$	$\lambda x.t$	tt'
$A \otimes B$	$\langle t_1, t_2 \rangle$	$\mathtt{let}\ \langle x, y \rangle = t\ \mathtt{in}\ t'$
$\mathtt{Nat}$	$\mathtt{zero},\ \mathtt{S}\ t$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{zero} \mapsto t_1 \mid \mathtt{S}\ \mapsto t_2\}$
$\mathtt{List}(A)$	$\mathtt{Empty},\ \mathtt{Cons}\ \langle h, t \rangle$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{Empty} \mapsto t_1 \mid \mathtt{Cons} \mapsto t_2\}$
$A \oplus B$	$\mathtt{left}\ t,\ \mathtt{right}\ t$	$\mathtt{match}\ t\ \mathtt{with}\ \{\mathtt{left}\ \mapsto t_1 \mid \mathtt{right}\ \mapsto t_2\}$

$$\mathtt{Nat} \cong \mathtt{I} \oplus \mathtt{Nat} \qquad \mathtt{List}(A) \cong \mathtt{I} \oplus (A \otimes \mathtt{List}(A))$$

General inductive types: $\mu X.A$.

$$\mathtt{Nat} = \mu X.\mathtt{I} \oplus X \qquad \mathtt{List}(A) = \mu X.\mathtt{I} \oplus (A \times X)$$

Introduction: Effects

Dichotomy between purely functional programs and an interaction with the environment.

Introduction: Effects

Dichotomy between purely functional programs and an interaction with the environment.

Effectful types are written with $\mathcal{T}X$.

$$\mathsf{do}\ x \leftarrow t;\ \ t'$$

Introduction: Effects

Dichotomy between purely functional programs and an interaction with the environment.

Effectful types are written with $\mathcal{T}X$.

$$\texttt{do}\ x \leftarrow t;\ \ t'$$

Writer effect

- a tape external to the program
- effectful programs write on the tape

Introduction: Effects

Dichotomy between purely functional programs and an interaction with the environment.

Effectful types are written with $\mathcal{T}X$.

$$\texttt{do}\ x \leftarrow t;\ t'$$

Writer effect

- a tape external to the program
- effectful programs write on the tape

Non-deterministic effect

- Example: $\texttt{x} \leftarrow \texttt{True} \parallel \texttt{False}$.
- No control over the value.

Introduction: Effects

Dichotomy between purely functional programs and an interaction with the environment.

Effectful types are written with $\mathcal{T}X$.

$$\texttt{do}\ x \leftarrow t;\ t'$$

Writer effect

- a tape external to the program
- effectful programs write on the tape

Probabilistic effect

- Example: $\texttt{x} \leftarrow \texttt{Coin}()$.
- According to a probability distribution.

Non-deterministic effect

- Example: $\texttt{x} \leftarrow \texttt{True} \parallel \texttt{False}$.
- No control over the value.

Introduction: Effects

Dichotomy between purely functional programs and an interaction with the environment.

Effectful types are written with $\mathcal{T}X$.

$$\texttt{do}\ x \leftarrow t;\ t'$$

Writer effect

- a tape external to the program
- effectful programs write on the tape

Probabilistic effect

- Example: $\texttt{x} \leftarrow \texttt{Coin}()$.
- According to a probability distribution.

Non-deterministic effect

- Example: $\texttt{x} \leftarrow \texttt{True} \parallel \texttt{False}$.
- No control over the value.

Quantum effect

- Example: $\texttt{x} \leftarrow \texttt{QCoin}()$.
- Superposition.
- Can be observed with measurement.

Introduction: Quantum Computing

- Quantum bits in $\mathbb{C} \oplus \mathbb{C}$, we write $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Generally: quantum data as normalised vectors in a Hilbert space.

Introduction: Quantum Computing

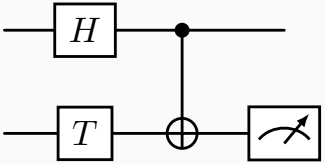
- Quantum bits in $\mathbb{C} \oplus \mathbb{C}$, we write $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Generally: quantum data as normalised vectors in a Hilbert space.
- Physically admissible operations are unitaries (reversible) and measurement (irreversible).

Introduction: Quantum Computing

- Quantum bits in $\mathbb{C} \oplus \mathbb{C}$, we write $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Generally: quantum data as normalised vectors in a Hilbert space.
- Physically admissible operations are unitaries (reversible) and measurement (irreversible).
- The most used model of quantum computation is circuits.

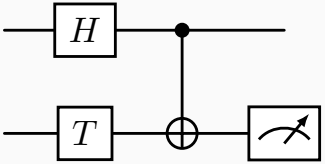
Introduction: Quantum Computing

- Quantum bits in $\mathbb{C} \oplus \mathbb{C}$, we write $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Generally: quantum data as normalised vectors in a Hilbert space.
- Physically admissible operations are unitaries (reversible) and measurement (irreversible).
- The most used model of quantum computation is circuits.



Introduction: Quantum Computing

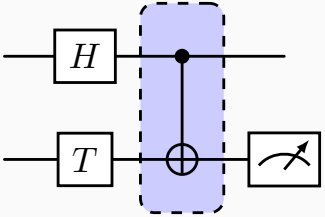
- Quantum bits in $\mathbb{C} \oplus \mathbb{C}$, we write $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Generally: quantum data as normalised vectors in a Hilbert space.
- Physically admissible operations are unitaries (reversible) and measurement (irreversible).
- The most used model of quantum computation is circuits.



Input: $|x, y\rangle$
Program: $x \leftarrow Hx$
$y \leftarrow Ty$
`if` x `then` $\langle x, 1 - y\rangle$
`else` $\langle x, y\rangle$
`meas`(y)

Introduction: Quantum Computing

- Quantum bits in $\mathbb{C} \oplus \mathbb{C}$, we write $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Generally: quantum data as normalised vectors in a Hilbert space.
- Physically admissible operations are unitaries (reversible) and measurement (irreversible).
- The most used model of quantum computation is circuits.



Controlled Not

Input: $|x, y\rangle$
Program: $x \leftarrow Hx$
$y \leftarrow Ty$
`if` x `then` $\langle x, 1 - y \rangle$
`else` $\langle x, y \rangle$
`meas`(y)

Introduction: Classical vs. Quantum Control

Quantum λ-calculus [SelingerValiron09].

Input: $|x, y\rangle$

Program: $x \leftarrow U_1 x$

$\texttt{if meas}(x)\ \texttt{then}\ U_2 y$

$\texttt{else}\ y$

$\texttt{meas}(x)$ is 0 or 1 with a certain probability.

Therefore, **compute** $U_2 y$ is only probable.

Introduction: Classical vs. Quantum Control

Quantum λ-calculus [SelingerValiron09].

Input: $|x, y\rangle$

Program: $x \leftarrow U_1 x$

$\texttt{if meas}(x)\ \texttt{then}\ U_2 y\ \texttt{else}\ y$

$\texttt{meas}(x)$ is 0 or 1 with a certain probability.
Therefore, **compute** $U_2 y$ is only probable.

Our goal (Chapter 3).

Input: $|x, y\rangle$

Program: $x \leftarrow U_1 x$

$\texttt{if}\ x\ \texttt{then}\ |x, U_2 y\rangle\ \texttt{else}\ |x, y\rangle$

If x is a superposition $\alpha |0\rangle + \beta |1\rangle$, we get

$$\alpha |0, y\rangle + \beta |1, U_2 y\rangle .$$

Introduction: Classical vs. Quantum Control

Quantum λ-calculus [SelingerValiron09].

Input: $|x, y\rangle$

Program: $x \leftarrow U_1 x$

$\texttt{if meas}(x)\ \texttt{then}\ U_2 y\ \texttt{else}\ y$

$\texttt{meas}(x)$ is 0 or 1 with a certain probability.
Therefore, **compute** $U_2 y$ is only probable.

Studied with a point of view from semantics.

Our goal (Chapter 3).

Input: $|x, y\rangle$

Program: $x \leftarrow U_1 x$

$\texttt{if } x\ \texttt{then}\ |x, U_2 y\rangle\ \texttt{else}\ |x, y\rangle$

If x is a superposition $\alpha |0\rangle + \beta |1\rangle$, we get

$$\alpha |0, y\rangle + \beta |1, U_2 y\rangle .$$

Introduction: Why study semantics?

Comes from Ancient Greek, *to provide a meaning*.

Introduction: Why study semantics?

Comes from Ancient Greek, *to provide a meaning*.

Operational semantics: formal program execution $t \to t'$.

Introduction: Why study semantics?

Comes from Ancient Greek, *to provide a meaning*.

Operational semantics: formal program execution $t \to t'$.

For example, $(\lambda x.x + 2)3 \to$

Introduction: Why study semantics?

Comes from Ancient Greek, *to provide a meaning*.

Operational semantics: formal program execution $t \to t'$.

For example, $(\lambda x.x + 2)3 \to 3 + 2 \to$

Introduction: Why study semantics?

Comes from Ancient Greek, *to provide a meaning*.

Operational semantics: formal program execution $t \to t'$.

For example, $(\lambda x.x + 2)3 \to 3 + 2 \to 5$.

Introduction: Why study semantics?

Comes from Ancient Greek, *to provide a meaning*.

Operational semantics: formal program execution $t \to t'$.

$$\text{For example, } (\lambda x.x + 2)3 \to 3 + 2 \to 5.$$

Denotational semantics: mathematical function $[\![t]\!]$.

Introduction: Why study semantics?

Comes from Ancient Greek, *to provide a meaning*.

Operational semantics: formal program execution $t \to t'$.

$$\text{For example, } (\lambda x.x + 2)3 \to 3 + 2 \to 5.$$

Denotational semantics: mathematical function $[\![t]\!]$.

Strong enough if relation between operational and denotational:

$$t \simeq t' \text{ iff } [\![t]\!] = [\![t']\!].$$

Introduction: Why study semantics?

Comes from Ancient Greek, *to provide a meaning*.

Operational semantics: formal program execution $t \to t'$.

$$\text{For example, } (\lambda x.x + 2)3 \to 3 + 2 \to 5.$$

Denotational semantics: mathematical function $[\![t]\!]$.

Strong enough if relation between operational and denotational:

$$t \simeq t' \text{ iff } [\![t]\!] = [\![t']\!].$$

- Prove that the language does what it is supposed to do,
- Compile the language and perform optimisations safely,
- Inform on which features can be added to the language.

Introduction: Category theory

Category

- Objects: $X, Y, Z, \ldots$
- Morphisms: $f\colon X \to Y$.
- Composition:

$$\frac{f\colon X \to Y \quad g\colon Y \to Z}{g \circ f\colon X \to Z}$$

- Identity morphism: $\mathrm{id}_X\colon X \to X$.

Type system

- Types: $A, B, C, \ldots$
- Type judgements: $x\colon A \vdash t\colon B$.
- Also composable:

$$\frac{x\colon A \vdash t\colon B \quad y\colon B \vdash t'\colon C}{x\colon A \vdash t'[t/y]\colon C}$$

- Identity axiom: $x\colon A \vdash x\colon A$.

Introduction: Category theory

Category

- Objects: $X, Y, Z, \ldots$
- Morphisms: $f\colon X \to Y$.
- Composition:

$$\frac{f\colon X \to Y \quad g\colon Y \to Z}{g \circ f\colon X \to Z}$$

- Identity morphism: $\mathrm{id}_X\colon X \to X$.

Type system

- Types: $A, B, C, \ldots$
- Type judgements: $x\colon A \vdash t\colon B$.
- Also composable:

$$\frac{x\colon A \vdash t\colon B \quad y\colon B \vdash t'\colon C}{x\colon A \vdash t'[t/y]\colon C}$$

- Identity axiom: $x\colon A \vdash x\colon A$.

Most effects are interpreted as a (strong) monad $\mathcal{T}$ [Moggi89].

Introduction: Category theory

Category

- Objects: $X, Y, Z, \ldots$
- Morphisms: $f\colon X \to Y$.
- Composition:

$$\frac{f\colon X \to Y \quad g\colon Y \to Z}{g \circ f\colon X \to Z}$$

- Identity morphism: $\mathrm{id}_X\colon X \to X$.

Type system

- Types: $A, B, C, \ldots$
- Type judgements: $x\colon A \vdash t\colon B$.
- Also composable:

$$\frac{x\colon A \vdash t\colon B \quad y\colon B \vdash t'\colon C}{x\colon A \vdash t'[t/y]\colon C}$$

- Identity axiom: $x\colon A \vdash x\colon A$.

Most effects are interpreted as a (strong) monad $\mathcal{T}$ [Moggi89].

Example (writer effect), with a monoid $(M, \cdot, e)$: Monad: $M \times -$ Objects: $M \times X$

Introduction: Category theory

Category

- Objects: $X, Y, Z, \ldots$
- Morphisms: $f\colon X \to Y$.
- Composition:

$$\frac{f\colon X \to Y \quad g\colon Y \to Z}{g \circ f\colon X \to Z}$$

- Identity morphism: $\mathrm{id}_X\colon X \to X$.

Type system

- Types: $A, B, C, \ldots$
- Type judgements: $x\colon A \vdash t\colon B$.
- Also composable:

$$\frac{x\colon A \vdash t\colon B \quad y\colon B \vdash t'\colon C}{x\colon A \vdash t'[t/y]\colon C}$$

- Identity axiom: $x\colon A \vdash x\colon A$.

Most effects are interpreted as a (strong) monad $\mathcal{T}$ [Moggi89].

Example (writer effect), with a monoid $(M, \cdot, e)$: Monad: $M \times -$ Objects: $M \times X$

$$\text{Unit: } \eta_X = \begin{cases} X & \to & M \times X \\ x & \mapsto & (e, x) \end{cases}$$

Introduction: Category theory

Category

- Objects: $X, Y, Z, \ldots$
- Morphisms: $f\colon X \to Y$.
- Composition:

$$\frac{f\colon X \to Y \quad g\colon Y \to Z}{g \circ f\colon X \to Z}$$

- Identity morphism: $\mathrm{id}_X\colon X \to X$.

Type system

- Types: $A, B, C, \ldots$
- Type judgements: $x\colon A \vdash t\colon B$.
- Also composable:

$$\frac{x\colon A \vdash t\colon B \quad y\colon B \vdash t'\colon C}{x\colon A \vdash t'[t/y]\colon C}$$

- Identity axiom: $x\colon A \vdash x\colon A$.

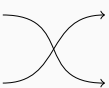
Most effects are interpreted as a (strong) monad $\mathcal{T}$ [Moggi89].

Example (writer effect), with a monoid $(M, \cdot, e)$: Monad: $M \times -$ Objects: $M \times X$

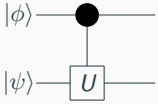
Unit: $\eta_X = \begin{cases} X & \to & M \times X \\ x & \mapsto & (e, x) \end{cases}$ Multiplication: $\mu_X = \begin{cases} M \times (M \times X) & \to & M \times X \\ (m, (m', x) & \mapsto & (m \cdot m', x) \end{cases}$

Contents

Effects



Centrality
Chapter 2, LICS'23

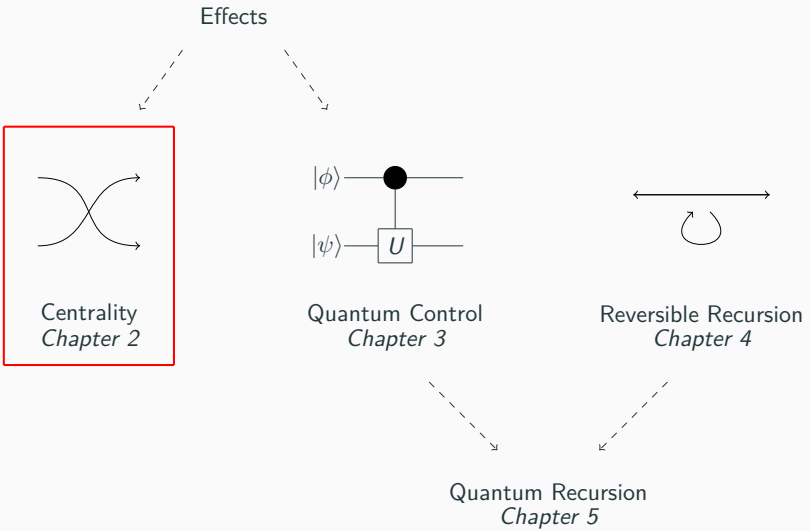


Quantum Control
Chapter 3, Submitted

Reversible Recursion
Chapter 4, FSCD'24

Quantum Recursion
Chapter 5

Contents


Effects
$|\phi\rangle$
$|\psi\rangle$
U
Centrality
Chapter 2
Quantum Control
Chapter 3
Reversible Recursion
Chapter 4
Quantum Recursion
Chapter 5

Centrality: Motivation

- Monads represent computational effects.
- Monads have an algebraic flavour (category theory).
- Centre (algebraically): elements that commute with all others.

Centrality: Motivation

- Monads represent computational effects.
- Monads have an algebraic flavour (category theory).
- Centre (algebraically): elements that commute with all others.

```
p1:                    p2:
  do                     do
    x <- op1               y <- op2
    y <- op2               x <- op1
    f x y                  f x y
```

Two examples of monadic sequencing in Haskell.

- **Intuition:** If `op1` **or** `op2` is central, `p1` and `p2` should be equivalent.

Centrality: Motivation

- Monads represent computational effects.
- Monads have an algebraic flavour (category theory).
- Centre (algebraically): elements that commute with all others.

```
p1:                    p2:
  do                     do
    x <- op1               y <- op2
    y <- op2               x <- op1
    f x y                  f x y
```

Two examples of monadic sequencing in Haskell.

- **Intuition:** If `op1` **or** `op2` is central, `p1` and `p2` should be equivalent.

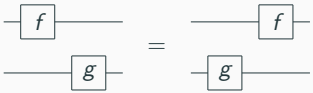
Question:

What is centrality for monads?

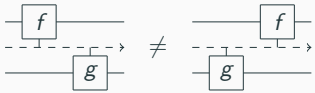
Centrality: Background on Premonoidal categories

[PowerRobinson97] Premonoidal categories as model of effects.

Monoidal category



Premonoidal category



- A premonoidal category $\mathcal{P}$ has a centre $Z(\mathcal{P})$;
- $Z(\mathcal{P})$ is a monoidal subcategory of $\mathcal{P}$.

Centrality: Strong monads

Strong monad $\mathcal{T}$: combines effect with pairing ($\otimes$) with a strength τ:

$$\tau_{X,Y} \colon X \otimes \mathcal{T}Y \to \mathcal{T}(X \otimes Y).$$

Operationally:

$$\begin{aligned} \text{Input:} \quad & \langle x, M \rangle \\ \text{Output:} \quad & \texttt{do}\ y \leftarrow M\ ;\ \texttt{return}\ \langle x, y \rangle \end{aligned}$$

With strong monad $\mathcal{T} \colon \mathbf{C} \to \mathbf{C}$, the category $\mathbf{C}_{\mathcal{T}}$:

- Objects of $\mathbf{C}$;
- Morphisms $X \to \mathcal{T}Y$.

is called Kleisli category, and is a premonoidal category.

Centrality: Examples on Set

Writer monad

- Monoid M with centre $Z(M)$.
- Monad – $(M \times -) : \mathbf{Set} \to \mathbf{Set}$.
- Centre – $(Z(M) \times -) : \mathbf{Set} \to \mathbf{Set}$.

Powerset monad: non-determinism

- Commutative.
- Centre – itself.

Centrality: Examples on Set

Writer monad

- Monoid M with centre $Z(M)$.
- Monad – $(M \times -) : \mathbf{Set} \to \mathbf{Set}$.
- Centre – $(Z(M) \times -) : \mathbf{Set} \to \mathbf{Set}$.

Powerset monad: non-determinism

- Commutative.
- Centre – itself.

Obtained by considering the appropriate subset $\mathcal{Z}X \subseteq \mathcal{T}X$.

Centrality: Central cone

Commutative monad:
for all M and N:

$$\begin{array}{l} x \leftarrow N \\ y \leftarrow M \\ \texttt{return}\ \langle x, y \rangle \end{array} \quad \equiv \quad \begin{array}{l} y \leftarrow M \\ x \leftarrow N \\ \texttt{return}\ \langle x, y \rangle \end{array}$$

Central cone: A pair $(Z, \iota \colon Z \to TX)$
such that, for all M:

$$\begin{array}{l} x \leftarrow \iota(z) \\ y \leftarrow M \\ \texttt{return}\ \langle x, y \rangle \end{array} \quad \equiv \quad \begin{array}{l} y \leftarrow M \\ x \leftarrow \iota(z) \\ \texttt{return}\ \langle x, y \rangle \end{array}$$

Centrality: Central cone

Commutative monad:
for all M and N:

$$\begin{array}{l} x \leftarrow N \\ y \leftarrow M \\ \texttt{return}\ \langle x, y \rangle \end{array} \quad \equiv \quad \begin{array}{l} y \leftarrow M \\ x \leftarrow N \\ \texttt{return}\ \langle x, y \rangle \end{array}$$

Central cone: A pair $(Z, \iota \colon Z \to TX)$
such that, for all M:

$$\begin{array}{l} x \leftarrow \iota(z) \\ y \leftarrow M \\ \texttt{return}\ \langle x, y \rangle \end{array} \quad \equiv \quad \begin{array}{l} y \leftarrow M \\ x \leftarrow \iota(z) \\ \texttt{return}\ \langle x, y \rangle \end{array}$$

If the universal central cone at X exists, write $\mathcal{Z}X \stackrel{\text{def}}{=} Z$.

Centrality: Conditions for Centralisability

Theorem 2.11: Equivalent conditions for a monad $\mathcal{T}$ to have a centre:

1. All universal central cones.
2. A monad $\mathcal{Z}$ linked to $Z(\mathbf{C}_{\mathcal{T}})$.
3. An adjunction between $\mathbf{C}$ and $Z(\mathbf{C}_{\mathcal{T}})$.

Centrality: Conditions for Centralisability

Theorem 2.11: Equivalent conditions for a monad $\mathcal{T}$ to have a centre:

1. All universal central cones.
2. A monad $\mathcal{Z}$ linked to $Z(\mathbf{C}_{\mathcal{T}})$.
3. An adjunction between $\mathbf{C}$ and $Z(\mathbf{C}_{\mathcal{T}})$.

Corollaries (1) Strong monads on $\mathbf{Set}, \mathbf{DCPO}, \mathbf{Top}, \mathbf{Vect}, \ldots$ are centralisable.

Centrality: Conditions for Centralisability

Theorem 2.11: Equivalent conditions for a monad $\mathcal{T}$ to have a centre:

1. All universal central cones.
2. A monad $\mathcal{Z}$ linked to $Z(\mathbf{C}_{\mathcal{T}})$.
3. An adjunction between $\mathbf{C}$ and $Z(\mathbf{C}_{\mathcal{T}})$.

Corollaries (1) Strong monads on $\mathbf{Set}, \mathbf{DCPO}, \mathbf{Top}, \mathbf{Vect}, \ldots$ are centralisable.
(2) A commutative monad is its own centre.

Centrality: Conditions for Centralisability

Theorem 2.11: Equivalent conditions for a monad $\mathcal{T}$ to have a centre:

1. All universal central cones.
2. A monad $\mathcal{Z}$ linked to $Z(\mathbf{C}_{\mathcal{T}})$.
3. An adjunction between $\mathbf{C}$ and $Z(\mathbf{C}_{\mathcal{T}})$.

Corollaries
(1) Strong monads on $\mathbf{Set}, \mathbf{DCPO}, \mathbf{Top}, \mathbf{Vect}, \ldots$ are centralisable.
(2) A commutative monad is its own centre.
(3) If $\mathbf{C}$ closed and total, every strong monad on it admits a centre.

Centrality: Conditions for Centralisability

Theorem 2.11: Equivalent conditions for a monad $\mathcal{T}$ to have a centre:

1. All universal central cones.
2. A monad $\mathcal{Z}$ linked to $Z(\mathbf{C}_{\mathcal{T}})$.
3. An adjunction between $\mathbf{C}$ and $Z(\mathbf{C}_{\mathcal{T}})$.

Corollaries
(1) Strong monads on $\mathbf{Set}, \mathbf{DCPO}, \mathbf{Top}, \mathbf{Vect}, \ldots$ are centralisable.
(2) A commutative monad is its own centre.
(3) If $\mathbf{C}$ closed and total, every strong monad on it admits a centre.

All strong monads centralisable?

Centrality: Conditions for Centralisability

Theorem 2.11: Equivalent conditions for a monad $\mathcal{T}$ to have a centre:

1. All universal central cones.
2. A monad $\mathcal{Z}$ linked to $Z(\mathbf{C}_{\mathcal{T}})$.
3. An adjunction between $\mathbf{C}$ and $Z(\mathbf{C}_{\mathcal{T}})$.

Corollaries
(1) Strong monads on $\mathbf{Set}, \mathbf{DCPO}, \mathbf{Top}, \mathbf{Vect}, \ldots$ are centralisable.
(2) A commutative monad is its own centre.
(3) If $\mathbf{C}$ closed and total, every strong monad on it admits a centre.

All strong monads centralisable? No, but only artificial counterexamples!

Centrality: Conditions for Centralisability

Theorem 2.11: Equivalent conditions for a monad $\mathcal{T}$ to have a centre:

1. All universal central cones.
2. A monad $\mathcal{Z}$ linked to $Z(\mathbf{C}_{\mathcal{T}})$.
3. An adjunction between $\mathbf{C}$ and $Z(\mathbf{C}_{\mathcal{T}})$.

Corollaries
(1) Strong monads on $\mathbf{Set}, \mathbf{DCPO}, \mathbf{Top}, \mathbf{Vect}, \ldots$ are centralisable.
(2) A commutative monad is its own centre.
(3) If $\mathbf{C}$ closed and total, every strong monad on it admits a centre.

All strong monads centralisable? No, but only artificial counterexamples!

If M monoid, $X \subseteq Z(M)$ contains only central elements.

Centrality: Conditions for Centralisability

Theorem 2.11: Equivalent conditions for a monad $\mathcal{T}$ to have a centre:

1. All universal central cones.
2. A monad $\mathcal{Z}$ linked to $Z(\mathbf{C}_{\mathcal{T}})$.
3. An adjunction between $\mathbf{C}$ and $Z(\mathbf{C}_{\mathcal{T}})$.

Corollaries
(1) Strong monads on $\mathbf{Set}, \mathbf{DCPO}, \mathbf{Top}, \mathbf{Vect}, \ldots$ are centralisable.
(2) A commutative monad is its own centre.
(3) If $\mathbf{C}$ closed and total, every strong monad on it admits a centre.

All strong monads centralisable? No, but only artificial counterexamples!

If M monoid, $X \subseteq Z(M)$ contains only central elements.

Central submonad: $\mathcal{S} \hookrightarrow \mathcal{T}$ with only central effects.

Centrality: Computational Interpretation

Language	Model
Simply-typed λ-calculus (STλC)	Cartesian Closed Category (CCC)
Moggi's metalanguage	CCC with strong monad $\mathcal{T}$ [Moggi89]
???	CCC with central submonad $\mathcal{S} \hookrightarrow \mathcal{T}$

Centrality: Computational Interpretation

Language	Model
Simply-typed λ-calculus (STλC)	Cartesian Closed Category (CCC)
Moggi's metalanguage	CCC with strong monad $\mathcal{T}$ [Moggi89]
CSC	CCC with central submonad $\mathcal{S} \hookrightarrow \mathcal{T}$

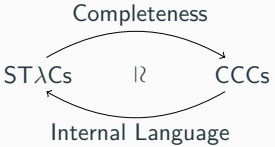
CSC (Central Submonad Calculus):

Simply-typed λ-calculus;
+ new types: $\mathcal{S}X$ and $\mathcal{T}X$;
+ terms for monadic computation (à la Moggi);
+ equational rules, such as:

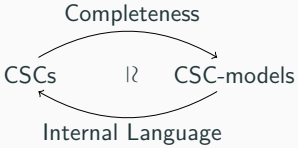
$$\Gamma \vdash \mathtt{do}_{\mathcal{T}}\ x \leftarrow \iota M;\ \mathtt{do}_{\mathcal{T}}\ y \leftarrow N;\ P$$
$$= \mathtt{do}_{\mathcal{T}}\ y \leftarrow N;\ \mathtt{do}_{\mathcal{T}}\ x \leftarrow \iota M;\ P : \mathcal{T}C$$

Centrality: Completeness and Internal Language

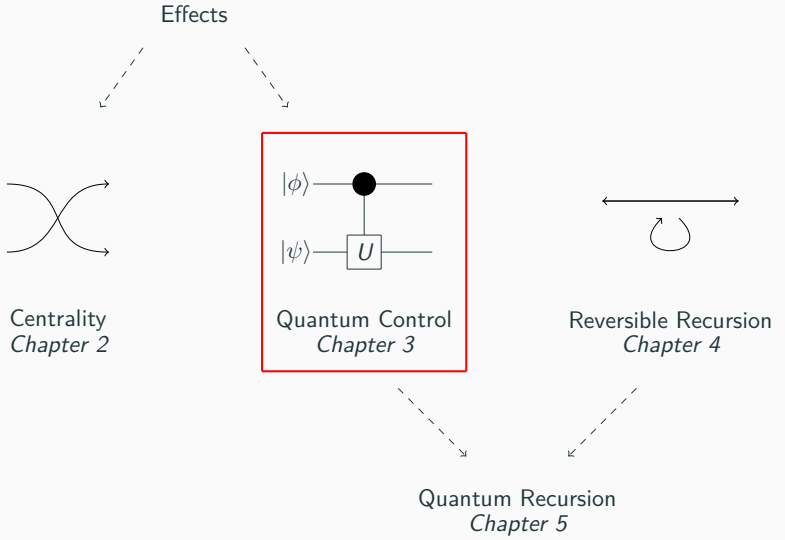
Folklore:



Theorem 2.60



Contents


Effects
$|\phi\rangle$
$|\psi\rangle$
U
Centrality
Chapter 2
Quantum Control
Chapter 3
Reversible Recursion
Chapter 4
Quantum Recursion
Chapter 5

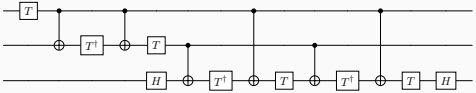
Quantum Control: Motivation

Usual quantum programming languages: unitaries (circuits) and measurement.

Quantum Control: Motivation

Usual quantum programming languages: unitaries (circuits) and measurement.

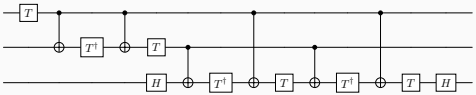
Example, **Toffoli**:



Quantum Control: Motivation

Usual quantum programming languages: unitaries (circuits) and measurement.

Example, **Toffoli**:

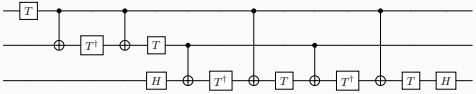


$$\begin{cases} |0\rangle \otimes y \otimes z & \mapsto & |0\rangle \otimes y \otimes z \\ x \otimes |0\rangle \otimes z & \mapsto & x \otimes |0\rangle \otimes z \\ |1\rangle \otimes |1\rangle \otimes z & \mapsto & |1\rangle \otimes |1\rangle \otimes (1 - z) \end{cases}$$

Quantum Control: Motivation

Usual quantum programming languages: unitaries (circuits) and measurement.

Example, **Toffoli**:



$$\begin{cases} |0\rangle \otimes y \otimes z & \mapsto & |0\rangle \otimes y \otimes z \\ x \otimes |0\rangle \otimes z & \mapsto & x \otimes |0\rangle \otimes z \\ |1\rangle \otimes |1\rangle \otimes z & \mapsto & |1\rangle \otimes |1\rangle \otimes (1 - z) \end{cases}$$

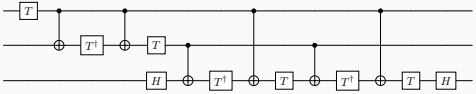
Goal:

Represent syntactically normalised quantum states and unitaries. Based on [SVV18].

Quantum Control: Motivation

Usual quantum programming languages: unitaries (circuits) and measurement.

Example, **Toffoli**:



$$\begin{cases} |0\rangle \otimes y \otimes z & \mapsto & |0\rangle \otimes y \otimes z \\ x \otimes |0\rangle \otimes z & \mapsto & x \otimes |0\rangle \otimes z \\ |1\rangle \otimes |1\rangle \otimes z & \mapsto & |1\rangle \otimes |1\rangle \otimes (1 - z) \end{cases}$$

Goal:

Represent syntactically normalised quantum states and unitaries. Based on [SVV18].

Difficulties:

- A notion of orthogonality for the norm.
- A notion of orthonormal basis for unitarity.

Quantum Control: The syntax

Types:

$$\texttt{I} \qquad A \oplus B \qquad A \otimes B \qquad \texttt{Nat}$$

Terms and semantics as isometries:

Quantum Control: The syntax

Types:

$$\boxed{\mathtt{I} \qquad A \oplus B} \qquad A \otimes B \qquad \mathtt{Nat}$$

Terms and semantics as isometries:

Syntax	$\mathbb{C}^n$	
$*\colon \mathtt{I}$	$\begin{bmatrix} 1 \end{bmatrix}$	1
$\mathtt{left}\ *\colon \mathtt{I} \oplus \mathtt{I}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\lvert 0 \rangle$
$\mathtt{right}\ *\colon \mathtt{I} \oplus \mathtt{I}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\lvert 1 \rangle$
$(\alpha \cdot \mathtt{left}\ *) + (\beta \cdot \mathtt{right}\ *)\colon \mathtt{I} \oplus \mathtt{I}$	$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$	$\alpha \lvert 0 \rangle + \beta \lvert 1 \rangle$

Quantum Control: The syntax

Types:

$$\mathtt{I} \qquad A \oplus B \qquad \boxed{A \otimes B} \qquad \mathtt{Nat}$$

Terms and semantics as isometries:

Syntax	$\mathbb{C}^n$	
$t_i \colon \mathtt{I} \oplus \mathtt{I}$	$\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}$	$\lvert\phi_i\rangle$
$t_1 \otimes t_2 \colon (\mathtt{I} \oplus \mathtt{I}) \otimes (\mathtt{I} \oplus \mathtt{I})$	$\begin{bmatrix} \alpha_1\alpha_2 \\ \beta_1\alpha_2 \\ \alpha_1\beta_2 \\ \beta_1\beta_2 \end{bmatrix}$	$\lvert\phi_1\rangle \otimes \lvert\phi_2\rangle$

Quantum Control: The syntax

Types:

$$\mathtt{I} \qquad A \oplus B \qquad A \otimes B \qquad \boxed{\mathtt{Nat}}$$

Terms and semantics as isometries:

Syntax	$\ell^2(\mathbb{N})$	
$\mathtt{zero} : \mathtt{Nat}$	${}^t\begin{bmatrix} 1 & 0 & \cdots \end{bmatrix}$	$\lvert \underline{0} \rangle$
$\mathtt{S\ zero} : \mathtt{Nat}$	${}^t\begin{bmatrix} 0 & 1 & 0 & \cdots \end{bmatrix}$	$\lvert \underline{1} \rangle$
$(\alpha \cdot \mathtt{zero}) + (\beta \cdot \mathtt{S\ S\ zero}) : \mathtt{Nat}$	${}^t\begin{bmatrix} \alpha & 0 & \beta & 0 & \cdots \end{bmatrix}$	$\alpha \lvert \underline{0} \rangle + \beta \lvert \underline{2} \rangle$

Quantum Control: Examples of Unitaries

Can be seen as λ-abstractions with pattern-matching.

$$\mathrm{id} = \left\{ x \mapsto x \right\} : \mathtt{qubit} \to \mathtt{qubit}$$

$$\mathrm{NOT} = \left\{ \begin{array}{lcl} \mathtt{ff} & \mapsto & \mathtt{tt} \\ \mathtt{tt} & \mapsto & \mathtt{ff} \end{array} \right\} : \mathtt{qubit} \to \mathtt{qubit}$$

$$\mathrm{QCoin} = \left\{ \begin{array}{lcl} \mathtt{ff} & \mapsto & \frac{1}{\sqrt{2}}\mathtt{ff} + \frac{1}{\sqrt{2}}\mathtt{tt} \\ \mathtt{tt} & \mapsto & \frac{1}{\sqrt{2}}\mathtt{ff} - \frac{1}{\sqrt{2}}\mathtt{tt} \end{array} \right\} : \mathtt{qubit} \to \mathtt{qubit}$$

$$\mathrm{SWAP} = \left\{ x \otimes y \mapsto y \otimes x \right\} : \mathtt{qubit}^{\otimes 2} \to \mathtt{qubit}^{\otimes 2}$$

$$\mathrm{CNOT} = \left\{ \begin{array}{lcl} (\mathtt{ff}) \otimes x & \mapsto & (\mathtt{ff}) \otimes x \\ (\mathtt{tt}) \otimes (\mathtt{ff}) & \mapsto & (\mathtt{tt}) \otimes (\mathtt{tt}) \\ (\mathtt{tt}) \otimes (\mathtt{tt}) & \mapsto & (\mathtt{tt}) \otimes (\mathtt{ff}) \end{array} \right\} : \mathtt{qubit}^{\otimes 2} \to \mathtt{qubit}^{\otimes 2}$$

$$\begin{array}{l} \mathtt{tt} = \mathtt{right}\ * \\ \mathtt{ff} = \mathtt{left}\ * \\ \mathtt{qubit} = \mathtt{I} \oplus \mathtt{I} \\ \mathtt{qubit}^{\otimes 2} = \mathtt{qubit} \otimes \mathtt{qubit} \end{array}$$

Quantum Control: Orthogonality

Mathematically (Hilbert spaces):

- Normalised vector $|\phi\rangle$.
- Orthogonality: $|\phi\rangle \perp |\psi\rangle$.
- Orthonormal basis (ONB): $\{|\phi_i\rangle\}_i$.
- Normalised superposition:
  If $|\phi_i\rangle$ are pairwise orthogonal,
  and $\sum_i |\alpha_i|^2 = 1$,
  then $\sum_i \alpha_i |\phi_i\rangle$ is normalised.

Syntactically:

- Well-typed term $\Delta \vdash t \colon A$.
- Orthogonality relation: $t_1 \perp t_2$.
- Orthogonal decomposition: $\mathrm{OD}_A(\{t_i\}_i)$.
- Typed superposition:

$$\frac{\Delta \vdash t_i \colon A \quad t_j \perp t_k \quad \sum_i |\alpha_i|^2 = 1}{\Delta \vdash \Sigma_i(\alpha_i \cdot t_i) \colon A}$$

Quantum Control: Orthogonal Decomposition

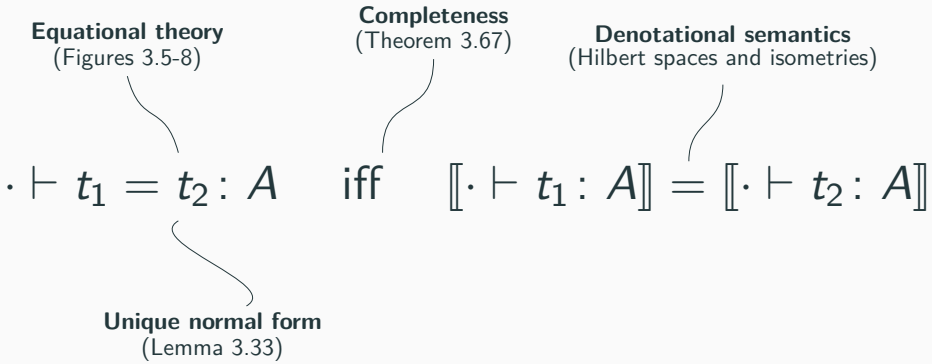
Mathematically (Hilbert spaces):

- Unitaries.
- Orthonormal basis (ONB): $\{|\phi_i\rangle\}_i$.
- Bijection between two ONBs $\leftrightarrow$ unitary.
- Resolution of the identity.

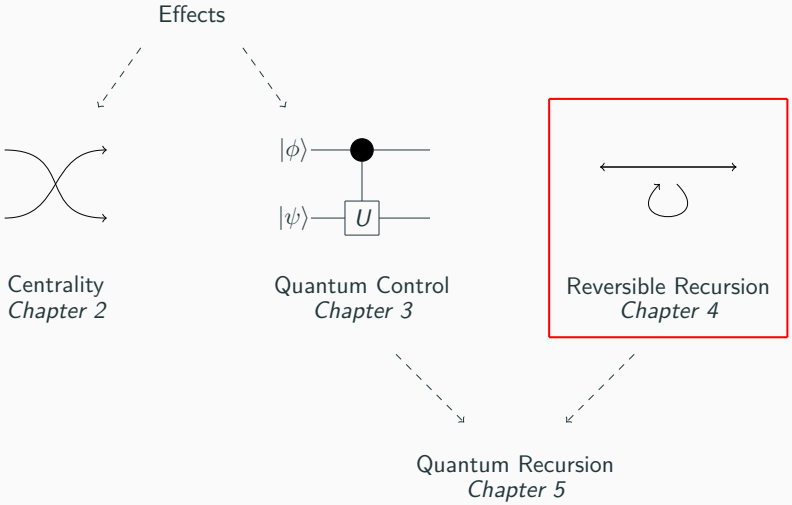
Syntactically:

- Syntactic unitaries.
- Orthogonal decomposition: $\mathrm{OD}_A(\{t_i\}_i)$.
- $$\frac{\mathrm{OD}_A(\{t_i\}_i) \quad \mathrm{OD}_B(\{t'_i\}_i) \quad \Delta_i \vdash t_i \colon A \quad \Delta_i \vdash t'_i \colon B}{\{t_i \mapsto t'_i\} \colon A \to B}$$
- Denotational resolution of the identity.

Quantum Control: Completeness



Contents


Effects
$|\phi\rangle$
$|\psi\rangle$
U
Centrality
Chapter 2
Quantum Control
Chapter 3
Reversible Recursion
Chapter 4
Quantum Recursion
Chapter 5

Reversible Recursion: Motivation

Question:

How does reversibility interact with infinite data?

Reversible Recursion: Motivation

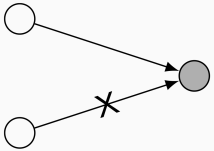
Question:

How does reversibility interact with infinite data?

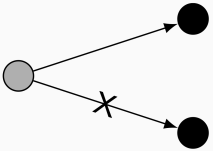
In Chapter 4:

- Focus on classical reversible fragment.
- Inductive types.
- A fixed point operator.

Backward determinism



Forward determinism



[KaarsgaardRennela21]

Reversible Recursion: Mathematical model

Inverse category:

- Every morphism f has a partial inverse.
- Every morphism f has a domain $\bar{f}$.
- $f \leq g$ iff $f = g\bar{f}$.

Reversible Recursion: Mathematical model

Inverse category:

- Every morphism f has a partial inverse.
- Every morphism f has a domain $\bar{f}$.
- $f \leq g$ iff $f = g\bar{f}$.

With a join structure:

If $f,\ g\colon X \to Y$ agree on there domains, then $f \vee g$.

Reversible Recursion: Mathematical model

Inverse category:

- Every morphism f has a partial inverse.
- Every morphism f has a domain $\bar{f}$.
- $f \leq g$ iff $f = g\bar{f}$.

With a join structure:

If $f,\ g\colon X \to Y$ agree on there domains, then $f \vee g$.

Together with function variables and a fixed point operator.

Reversible Recursion: Mathematical model

Inverse category:

- Every morphism f has a partial inverse.
- Every morphism f has a domain $\bar{f}$.
- $f \leq g$ iff $f = g\bar{f}$.

With a join structure:

If $f,\ g\colon X \to Y$ agree on there domains, then $f \vee g$.

Together with function variables and a fixed point operator.

Given two sets A and B, the set $[A \rightharpoonup B]$ has nice properties for fixed points.

(A rig join inverse category is enriched in pointed dcpos.)

Reversible Recursion: Turing-completeness, Adequacy

This language is Turing-complete! (Theorem 4.60, proven by Kostia Chardonnet)

Reversible Recursion: Turing-completeness, Adequacy

This language is Turing-complete! (Theorem 4.60, proven by Kostia Chardonnet)

Theorem 4.29: Sound and adequate denotational semantics.

Reversible Recursion: Turing-completeness, Adequacy

This language is Turing-complete! (Theorem 4.60, proven by Kostia Chardonnet)

Theorem 4.29: Sound and adequate denotational semantics.

Concrete model: partial injective functions.

Theorem 4.61: Any *computable* partial injection is representable in the language.

Reversible Recursion: Turing-completeness, Adequacy

This language is Turing-complete! (Theorem 4.60, proven by Kostia Chardonnet)

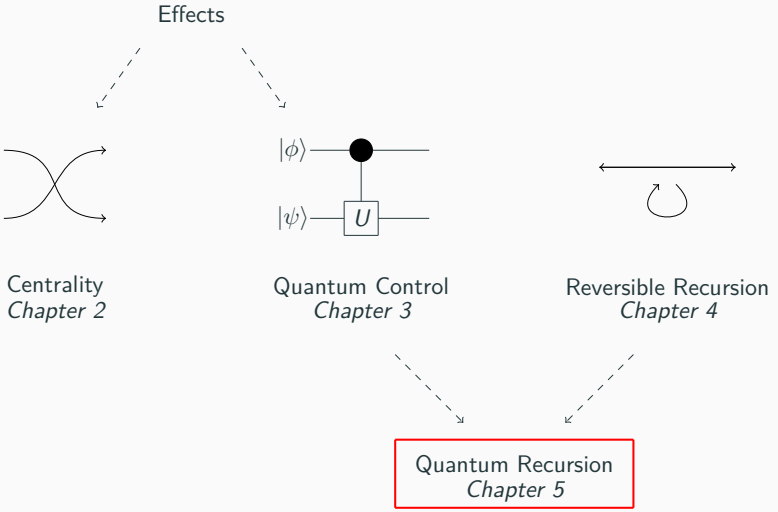
Theorem 4.29: Sound and adequate denotational semantics.

Concrete model: partial injective functions.

Theorem 4.61: Any *computable* partial injection is representable in the language.

However!

Contents


Effects
$|\phi\rangle$
$|\psi\rangle$
U
Centrality
Chapter 2
Quantum Control
Chapter 3
Reversible Recursion
Chapter 4
Quantum Recursion
Chapter 5

Quantum Recursion?

The nice structure (inherited from partial injections) is not preserved.

Quantum Recursion?

The nice structure (inherited from partial injections) is not preserved.

Assumptions:

- Partial identity smaller than the identity.
- Order preserved by composition.

Quantum Recursion?

The nice structure (inherited from partial injections) is not preserved.

Assumptions:

- Partial identity smaller than the identity.
- Order preserved by composition.

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \leq \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

Quantum Recursion?

The nice structure (inherited from partial injections) is not preserved.

Assumptions:

- Partial identity smaller than the identity.
- Order preserved by composition.

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \leq \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

thus

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \circ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \circ \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \leq \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \circ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \circ \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix},$$

Quantum Recursion?

The nice structure (inherited from partial injections) is not preserved.

Assumptions:

- Partial identity smaller than the identity.
- Order preserved by composition.

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \leq \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

thus
$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \circ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \circ \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \leq \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \circ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \circ \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix},$$

i.e.
$$\frac{1}{2} \leq 0.$$

An attempt: Guarded Recursion

Addition of:

- New types $\blacktriangleright A$.
- New constructor `next`: pause the computation.

An attempt: Guarded Recursion

Addition of:

- New types $\blacktriangleright A$.
- New constructor `next`: pause the computation.

Semantics of guarded recursion [BMSS12]. $X_0 \leftarrow X_1 \leftarrow X_2 \leftarrow X_3 \leftarrow \dots$

The topos of trees $\mathbf{Set}^{\mathbb{N}^{\mathrm{op}}}$

An attempt: Guarded Recursion

Addition of:

- New types $\blacktriangleright A$.
- New constructor `next`: pause the computation.

Semantics of guarded recursion [BMSS12]. $X_0 \leftarrow X_1 \leftarrow X_2 \leftarrow X_3 \leftarrow \dots$

The topos of trees $\mathbf{Set}^{\mathbb{N}^{\mathrm{op}}}$ (CCC $\leftrightarrow$ λ-calculus).

Results:

An attempt: Guarded Recursion

Addition of:

- New types $\blacktriangleright A$.
- New constructor `next`: pause the computation.

Semantics of guarded recursion [BMSS12]. $X_0 \leftarrow X_1 \leftarrow X_2 \leftarrow X_3 \leftarrow \dots$

The topos of trees $\mathbf{Set}^{\mathbb{N}^{\mathrm{op}}}$ (CCC $\leftrightarrow$ λ-calculus).

Results:

- Defined a (non-cartesian) category enriched in $\mathbf{Set}^{\mathbb{N}^{\mathrm{op}}}$ (Lemma 5.5).

An attempt: Guarded Recursion

Addition of:

- New types $\blacktriangleright A$.
- New constructor `next`: pause the computation.

Semantics of guarded recursion [BMSS12]. $X_0 \leftarrow X_1 \leftarrow X_2 \leftarrow X_3 \leftarrow \dots$

The topos of trees $\mathbf{Set}^{\mathbb{N}^{\mathrm{op}}}$ (CCC $\leftrightarrow$ λ-calculus).

Results:

- Defined a (non-cartesian) category enriched in $\mathbf{Set}^{\mathbb{N}^{\mathrm{op}}}$ (Lemma 5.5).
- Models inifinite data types and fixed point operators (Theorems 5.29 & 5.32).

Conclusion

Centrality and Monads, LICS'23

- With: Titouan Carette and Vladimir Zamdzhiev.
- What: Categorical study and computational interpretation of the centrality of effects.
- Why: A more refined view on the (non-)commutativity of effects.

Conclusion

Centrality and Monads, LICS'23

- With: Titouan Carette and Vladimir Zamdzhiev.
- What: Categorical study and computational interpretation of the centrality of effects.
- Why: A more refined view on the (non-)commutativity of effects.

Syntax for Quantum Control, Submitted

- With: Vladimir Zamdzhiev.
- What: Syntax and semantics for pure quantum computation.
- Why: Go beyond quantum circuits, master quantum control.

Conclusion

Centrality and Monads, LICS'23

- With: Titouan Carette and Vladimir Zamdzhiev.
- What: Categorical study and computational interpretation of the centrality of effects.
- Why: A more refined view on the (non-)commutativity of effects.

Syntax for Quantum Control, Submitted

- With: Vladimir Zamdzhiev.
- What: Syntax and semantics for pure quantum computation.
- Why: Go beyond quantum circuits, master quantum control.

Reversible Recursion, FSCD'24

- With: Kostia Chardonnet, Benoît Valiron.
- What: Semantics for a Turing-complete reversible language.
- Why: Develop reversible languages and make a step towards quantum recursion.