Quantum Recursion and Second Quantisation

Mingsheng Ying

University of Technology Sydney, Australia
and
Tsinghua University, China

Outline

Outline

Recursion is one of the central ideas of computer science!

Recursion is one of the central ideas of computer science!

Classical Recursion of Quantum Programs

- Recursive procedure in quantum programming language QPL [Selinger, *Mathematical Structures in Computer Science*'2004].
- Quantum while-loops [Ying, Feng, *Acta Informatica*'2010].

Recursion is one of the central ideas of computer science!

Classical Recursion of Quantum Programs

- Recursive procedure in quantum programming language QPL [Selinger, *Mathematical Structures in Computer Science*'2004].
- Quantum while-loops [Ying, Feng, *Acta Informatica*'2010].

- "Quantum data, classical control" [Selinger]:

  The control flow of quantum recursions is classical: branchings are determined by the outcomes of quantum measurements.

Recursion is one of the central ideas of computer science!

Classical Recursion of Quantum Programs

- Recursive procedure in quantum programming language QPL [Selinger, *Mathematical Structures in Computer Science*'2004].
- Quantum while-loops [Ying, Feng, *Acta Informatica*'2010].

- "Quantum data, classical control" [Selinger]:

  The control flow of quantum recursions is classical: branchings are determined by the outcomes of quantum measurements.
- Example:

$$\textbf{while } M[\overline{q}] = 1 \textbf{ do } S \textbf{ od}$$

Outline

How to define quantum control flow?

- "Quantum data, quantum control" [Altenkirch and Grattage, *LICS'2005*]

How to define quantum control flow?

- "Quantum data, quantum control" [Altenkirch and Grattage, *LICS'2005*]
- "Coined" quantum case statement [Ying, *Foundations of Quantum Programming*, Morgan Kaufmann 2016]

Quantum Case Statement

How to define quantum control flow?

- "Quantum data, quantum control" [Altenkirch and Grattage, *LICS'2005*]
- "Coined" quantum case statement [Ying, *Foundations of Quantum Programming*, Morgan Kaufmann 2016]

Quantum Case Statement

- Classical Case Statement:

$$\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}$$

How to define quantum control flow?

- "Quantum data, quantum control" [Altenkirch and Grattage, *LICS'2005*]
- "Coined" quantum case statement [Ying, *Foundations of Quantum Programming*, Morgan Kaufmann 2016]

Quantum Case Statement

- Classical Case Statement:

$$\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}$$

- Classical Case Statement in Quantum Programming:

$$\begin{array}{l} \textbf{if } M[\bar{q}] = 0 \to S_1 \\ \qquad \square \quad\ 1 \to S_2 \\ \textbf{fi} \end{array}$$

Quantum Case Statement

- Introduce external quantum coin c: $\mathcal{H}_c = \text{span}\{|0\rangle, |1\rangle\}$

Quantum Case Statement

- Introduce external quantum coin c: $\mathcal{H}_c = \text{span}\{|0\rangle, |1\rangle\}$
- U_0, U_1 unitary operators on quantum system q: $\mathcal{H}_q$

Quantum Case Statement

- Introduce external quantum coin c: $\mathcal{H}_c = \text{span}\{|0\rangle, |1\rangle\}$
- U_0, U_1 unitary operators on quantum system q: $\mathcal{H}_q$
- *Quantum case statement* employing coin c:

$$\begin{aligned} &\textbf{qif } [c]\ |0\rangle \rightarrow U_0[q] \\ &\qquad \square\ |1\rangle \rightarrow U_1[q] \\ &\textbf{fiq} \end{aligned}$$

Quantum Case Statement

- Introduce external quantum coin c: $\mathcal{H}_c = \text{span}\{|0\rangle, |1\rangle\}$
- U_0, U_1 unitary operators on quantum system q: $\mathcal{H}_q$
- *Quantum case statement* employing coin c:

$$\begin{aligned} &\textbf{qif } [c]\ |0\rangle \to U_0[q] \\ &\qquad \square\ |1\rangle \to U_1[q] \\ &\textbf{fiq} \end{aligned}$$

- Semantics: unitary operator U in $\mathcal{H}_c \otimes \mathcal{H}_q$:

$$U|0,\psi\rangle = |0\rangle U_0|\psi\rangle, \quad U|1,\psi\rangle = |1\rangle U_1|\psi\rangle$$

Quantum Choice

- V a unitary operator in $\mathcal{H}_c$.

Quantum Choice

- V a unitary operator in $\mathcal{H}_c$.
- *Quantum choice* of $U_0[q], U_1[q]$ with coin-tossing $V[c]$:

$$U_0[q] \oplus_{V[c]} U_1[q] \stackrel{\text{def}}{=} V[c]; \textbf{qif } [c]\ |0\rangle \to U_0[q] \square\ |1\rangle \to U_1[q] \textbf{ fiq}$$

External Quantum Coin

- Superpositions of time evolutions of a quantum system [Aharonov, Anandan, Popescu, Vaidman, *Physical Review Letters* 1990].

External Quantum Coin

- Superpositions of time evolutions of a quantum system [Aharonov, Anandan, Popescu, Vaidman, *Physical Review Letters* 1990].
- Quantum walks [Ambainis, Bach, Nayak, Vishwanath, Watrous, *STOC*'2001; Aharonov, Ambainis, Kempe, Vazirani, *STOC*'2001].

Quantum walk

External Quantum Coin

- Superpositions of time evolutions of a quantum system [Aharonov, Anandan, Popescu, Vaidman, *Physical Review Letters* 1990].
- Quantum walks [Ambainis, Bach, Nayak, Vishwanath, Watrous, *STOC'*2001; Aharonov, Ambainis, Kempe, Vazirani, *STOC'*2001].

Quantum walk

- One-dimensional random walk - a particle moves on a line marked by integers $\mathbb{Z}$; at each step it moves one position left or right, depending on the flip of a fair coin.

External Quantum Coin

- Superpositions of time evolutions of a quantum system [Aharonov, Anandan, Popescu, Vaidman, *Physical Review Letters* 1990].
- Quantum walks [Ambainis, Bach, Nayak, Vishwanath, Watrous, *STOC'*2001; Aharonov, Ambainis, Kempe, Vazirani, *STOC'*2001].

Quantum walk

- One-dimensional random walk - a particle moves on a line marked by integers $\mathbb{Z}$; at each step it moves one position left or right, depending on the flip of a fair coin.
- Hadamard walk - a quantum variant.

Quantum walk

- Hilbert space $\mathcal{H}_d \otimes \mathcal{H}_p$.

Quantum walk

- Hilbert space $\mathcal{H}_d \otimes \mathcal{H}_p$.
- $\mathcal{H}_d = \text{span}\{|L\rangle, |R\rangle\}$, L, R — direction Left and Right.

Quantum walk

- Hilbert space $\mathcal{H}_d \otimes \mathcal{H}_p$.
- $\mathcal{H}_d = \text{span}\{|L\rangle, |R\rangle\}$, L, R — direction Left and Right.
- $\mathcal{H}_p = \text{span}\{|n\rangle : n \in \mathbb{Z}\}$, n — the position marked by integer n.

Quantum walk

- Hilbert space $\mathcal{H}_d \otimes \mathcal{H}_p$.
- $\mathcal{H}_d = \text{span}\{|L\rangle, |R\rangle\}$, L, R — direction Left and Right.
- $\mathcal{H}_p = \text{span}\{|n\rangle : n \in \mathbb{Z}\}$, n — the position marked by integer n.
- One step of Hadamard walk:

$$W = T(H \otimes I)$$

Quantum walk

- Hilbert space $\mathcal{H}_d \otimes \mathcal{H}_p$.
- $\mathcal{H}_d = \text{span}\{|L\rangle, |R\rangle\}$, L, R — direction Left and Right.
- $\mathcal{H}_p = \text{span}\{|n\rangle : n \in \mathbb{Z}\}$, n — the position marked by integer n.
- One step of Hadamard walk:

$$W = T(H \otimes I)$$

  - Translation T: unitary operator in $\mathcal{H}_d \otimes \mathcal{H}_p$

$$T|L, n\rangle = |L, n-1\rangle, \quad T|R, n\rangle = |R, n+1\rangle$$

Quantum walk

- Hilbert space $\mathcal{H}_d \otimes \mathcal{H}_p$.
- $\mathcal{H}_d = \text{span}\{|L\rangle, |R\rangle\}$, L, R — direction Left and Right.
- $\mathcal{H}_p = \text{span}\{|n\rangle : n \in \mathbb{Z}\}$, n — the position marked by integer n.
- One step of Hadamard walk:

$$W = T(H \otimes I)$$

  - Translation T: unitary operator in $\mathcal{H}_d \otimes \mathcal{H}_p$

$$T|L, n\rangle = |L, n-1\rangle, \quad T|R, n\rangle = |R, n+1\rangle$$

  - Hadamard transform $\mathcal{H}_d$:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Quantum walk

- Hilbert space $\mathcal{H}_d \otimes \mathcal{H}_p$.
- $\mathcal{H}_d = \text{span}\{|L\rangle, |R\rangle\}$, L, R — direction Left and Right.
- $\mathcal{H}_p = \text{span}\{|n\rangle : n \in \mathbb{Z}\}$, n — the position marked by integer n.
- One step of Hadamard walk:

$$W = T(H \otimes I)$$

  - Translation T: unitary operator in $\mathcal{H}_d \otimes \mathcal{H}_p$

$$T|L, n\rangle = |L, n-1\rangle, \quad T|R, n\rangle = |R, n+1\rangle$$

  - Hadamard transform $\mathcal{H}_d$:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

- Hadamard walk — repeated applications of operator W.

Look quantum walk in a new way!

- Define left and right translation T_L and T_R in space $\mathcal{H}_p$:

$$T_L|n\rangle = |n-1\rangle, \quad T_R|n\rangle = |n+1\rangle$$

Look quantum walk in a new way!

- Define left and right translation T_L and T_R in space $\mathcal{H}_p$:

$$T_L|n\rangle = |n-1\rangle, \quad T_R|n\rangle = |n+1\rangle$$

- Translation operator T is a quantum case statement:

$$\begin{aligned} T = \textbf{qif } [d] \ |L\rangle &\to T_L[p] \\ \square \ |R\rangle &\to T_R[p] \\ \textbf{fiq} & \end{aligned}$$

Look quantum walk in a new way!

- Define left and right translation T_L and T_R in space $\mathcal{H}_p$:

$$T_L|n\rangle = |n-1\rangle, \quad T_R|n\rangle = |n+1\rangle$$

- Translation operator T is a quantum case statement:

$$\begin{aligned} T = \textbf{qif } [d]\ |L\rangle &\rightarrow T_L[p] \\ \square\ |R\rangle &\rightarrow T_R[p] \\ \textbf{fiq} & \end{aligned}$$

- Single-step walk operator W is a quantum choice:

$$T_L[p] \oplus_{H[d]} T_R[p]$$

Outline

Quantum recursion with quantum control flow can be defined based on quantum case statement

Quantum recursion with quantum control flow can be defined based on quantum case statement

A Quantum Programming Language

Alphabet:

- Two sets of quantum variables:
  1. principal system variables $p, q, \ldots$;

Quantum recursion with quantum control flow can be defined based on quantum case statement

A Quantum Programming Language

Alphabet:

- Two sets of quantum variables:
  1. principal system variables $p, q, ...$;
  2. coin variables $c, d, ...$

Syntax

$$P ::= X \mid \textbf{abort} \mid \textbf{skip} \mid U[\overline{q}, \overline{c}] \mid P_1; P_2 \mid \textbf{qif}\ [c](\square i \cdot |i\rangle \rightarrow P_i)\ \textbf{fiq}$$

Quantum recursion with quantum control flow can be defined based on quantum case statement

A Quantum Programming Language

Alphabet:

- Two sets of quantum variables:
  1. principal system variables $p, q, ...$;
  2. coin variables $c, d, ...$
- A set of procedure identifiers $X, X_1, X_2, ...$

Syntax

$$P ::= X \mid \textbf{abort} \mid \textbf{skip} \mid U[\overline{q}, \overline{c}] \mid P_1; P_2 \mid \textbf{qif}\ [c](\square i \cdot |i\rangle \to P_i)\ \textbf{fiq}$$

Quantum choice

$$[P(c)] \bigoplus_i (|i\rangle \to P_i) \stackrel{\triangle}{=} P; \textbf{qif } [c] \, (\square i \cdot |i\rangle \to P_i) \textbf{ end}.$$

Quantum choice

$$[P(c)] \bigoplus_i (|i\rangle \to P_i) \stackrel{\triangle}{=} P; \textbf{qif } [c] \, (\square i \cdot |i\rangle \to P_i) \textbf{ end}.$$

"Superposition of Programs"

- Quantum choice first runs a coin-tossing subprogram P followed by an alternation of a family of subprograms $P_0, P_1, ...$
- Coin-tossing subprogram P creates a superposition of the execution paths of $P_0, P_1, ...,$

Quantum choice

$$[P(c)] \bigoplus_i (|i\rangle \to P_i) \stackrel{\triangle}{=} P; \textbf{qif } [c] \, (\square i \cdot |i\rangle \to P_i) \textbf{ end}.$$

"Superposition of Programs"

- Quantum choice first runs a coin-tossing subprogram P followed by an alternation of a family of subprograms $P_0, P_1, ...$
- Coin-tossing subprogram P creates a superposition of the execution paths of $P_0, P_1, ...,$
- During the execution of the alternation, each P_i is running along its own path, but the whole program is executed in a superposition of execution paths of $P_0, P_1, ...$

Semantics of quantum programs without recursion

- $\mathcal{H}$ — Hilbert space of the principal system.

Semantics of quantum programs without recursion

- $\mathcal{H}$ — Hilbert space of the principal system.
- Semantics $[\![P]\!]$ of a program P without procedure identifiers:

Semantics of quantum programs without recursion

- $\mathcal{H}$ — Hilbert space of the principal system.
- Semantics $[\![P]\!]$ of a program P without procedure identifiers:
  1. If $P =$ **abort**, then $[\![P]\!] = 0$ (the zero operator in $\mathcal{H}$);

Semantics of quantum programs without recursion

- $\mathcal{H}$ — Hilbert space of the principal system.
- Semantics $[\![P]\!]$ of a program P without procedure identifiers:
  1. If $P = \textbf{abort}$, then $[\![P]\!] = 0$ (the zero operator in $\mathcal{H}$);
  2. If $P = \textbf{skip}$, then $[\![P]\!] = I$ (the identity operator in $\mathcal{H}$);

Semantics of quantum programs without recursion

- $\mathcal{H}$ — Hilbert space of the principal system.
- Semantics $[\![P]\!]$ of a program P without procedure identifiers:
  1. If $P =$ **abort**, then $[\![P]\!] = 0$ (the zero operator in $\mathcal{H}$);
  2. If $P =$ **skip**, then $[\![P]\!] = I$ (the identity operator in $\mathcal{H}$);
  3. If $P = U[\overline{q}, \overline{c}]$, then $[\![P]\!] = U$;

Semantics of quantum programs without recursion

- $\mathcal{H}$ — Hilbert space of the principal system.
- Semantics $[\![P]\!]$ of a program P without procedure identifiers:
  1. If $P =$ **abort**, then $[\![P]\!] = 0$ (the zero operator in $\mathcal{H}$);
  2. If $P =$ **skip**, then $[\![P]\!] = I$ (the identity operator in $\mathcal{H}$);
  3. If $P = U[\overline{q}, \overline{c}]$, then $[\![P]\!] = U$;
  4. If $P = P_1; P_2$, then $[\![P]\!] = [\![P_2]\!] \cdot [\![P_1]\!]$;

Semantics of quantum programs without recursion

- $\mathcal{H}$ — Hilbert space of the principal system.
- Semantics $[\![P]\!]$ of a program P without procedure identifiers:
  1. If $P =$ **abort**, then $[\![P]\!] = 0$ (the zero operator in $\mathcal{H}$);
  2. If $P =$ **skip**, then $[\![P]\!] = I$ (the identity operator in $\mathcal{H}$);
  3. If $P = U[\overline{q}, \overline{c}]$, then $[\![P]\!] = U$;
  4. If $P = P_1; P_2$, then $[\![P]\!] = [\![P_2]\!] \cdot [\![P_1]\!]$;
  5. If $P =$ **qif** $[c](\square i \cdot |i\rangle \to P_i)$ **fiq**, then

$$[\![P]\!] = \square_i (|i\rangle \to [\![P_i]\!]) = \sum_i (|i\rangle\langle i| \otimes [\![P_i]\!]) .$$

Quantum recursive programs

- Let $X_1, ..., X_m$ be different procedure identifiers. A declaration for $X_1, ..., X_m$ is a system of equations:

$$S : \begin{cases} X_1 \Leftarrow P_1, \\ \quad ..... \\ X_m \Leftarrow P_m, \end{cases}$$

where $P_i = P_i[X_1, ..., X_m]$ contains at most procedure identifiers $X_1, ..., X_m$.

Quantum recursive programs

- Let $X_1, ..., X_m$ be different procedure identifiers. A declaration for $X_1, ..., X_m$ is a system of equations:

$$S : \begin{cases} X_1 \Leftarrow P_1, \\ \quad ..... \\ X_m \Leftarrow P_m, \end{cases}$$

  where $P_i = P_i[X_1, ..., X_m]$ contains at most procedure identifiers $X_1, ..., X_m$.

- A recursive program consists of a main statement $P = P[X_1, ..., X_m]$ and a declaration S for $X_1, ..., X_m$.

Question: How to define semantics of quantum recursive programs?

Outline

Recursive Hadamard Walk

- The walk first runs coin-tossing operator $H[d]$, and then a quantum case statement:

Recursive Hadamard Walk

- The walk first runs coin-tossing operator $H[d]$, and then a quantum case statement:
  - if coin d is in state $|L\rangle$ then the walker moves one position left;

Recursive Hadamard Walk

- The walk first runs coin-tossing operator $H[d]$, and then a quantum case statement:
  - if coin d is in state $|L\rangle$ then the walker moves one position left;
  - if d is in state $|R\rangle$ then it moves one position right, followed by a procedure behaving as the recursive walk itself.

Recursive Hadamard Walk

- The walk first runs coin-tossing operator $H[d]$, and then a quantum case statement:
  - if coin d is in state $|L\rangle$ then the walker moves one position left;
  - if d is in state $|R\rangle$ then it moves one position right, followed by a procedure behaving as the recursive walk itself.
- Program X declared by recursive equation:

$$X \Leftarrow T_L[p] \oplus_{H[d]} (T_R[p]; X)$$

Recursive Hadamard Walk

- The walk first runs coin-tossing operator $H[d]$ and then a quantum case statement:

Recursive Hadamard Walk

- The walk first runs coin-tossing operator $H[d]$ and then a quantum case statement:
  - if coin d is in state $|L\rangle$ then the walker moves one position left, followed by a procedure behaving as the recursive walk itself;

Recursive Hadamard Walk

- The walk first runs coin-tossing operator $H[d]$ and then a quantum case statement:
  - if coin d is in state $|L\rangle$ then the walker moves one position left, followed by a procedure behaving as the recursive walk itself;
  - if d is in state $|R\rangle$ then it moves one position right, also followed by a procedure behaving as the recursive walk itself.

Recursive Hadamard Walk

- The walk first runs coin-tossing operator $H[d]$ and then a quantum case statement:
  - if coin d is in state $|L\rangle$ then the walker moves one position left, followed by a procedure behaving as the recursive walk itself;
  - if d is in state $|R\rangle$ then it moves one position right, also followed by a procedure behaving as the recursive walk itself.
- Program X declared by the recursive equation:

$$X \Leftarrow (T_L[p]; X) \oplus_{H[d]} (T_R[p]; X)$$

Recursive Hadamard Walk

- The walk first runs coin-tossing operator $H[d]$ and then a quantum case statement:
  - if coin d is in state $|L\rangle$ then the walker moves one position left, followed by a procedure behaving as the recursive walk itself;
  - if d is in state $|R\rangle$ then it moves one position right, also followed by a procedure behaving as the recursive walk itself.
- Program X declared by the recursive equation:

$$X \Leftarrow (T_L[p]; X) \oplus_{H[d]} (T_R[p]; X)$$

- A variant declared by system of recursive equations:

$$\begin{cases} X \Leftarrow T_L[p] \oplus_{H[d]} (T_R[p]; Y), \\ Y \Leftarrow (T_L[p]; X) \oplus_{H[d]} T_R[p] \end{cases}$$

More Interesting Recursive Hardamard Walk

- Let $n \geq 2$. Program declared by recursive equation:

$$X \Leftarrow ((T_L[p]; X) \oplus_{H[d]} (T_R[p]; X)); (T_L[p] \oplus_{H[d]} T_R[p])^n$$

$$\begin{aligned}(X, |L\rangle_d|0\rangle_p) \rightarrow \ldots \\ \rightarrow \frac{1}{2\sqrt{2}}[&(X, |L\rangle_d| - 3\rangle_p) + (X, |R\rangle_d| - 1\rangle_p) \\ &+ (X, |L\rangle_d| - 1\rangle_p) - (X, |R\rangle_d|1\rangle_p) \\ &+ (X, |L\rangle_d| - 1\rangle_p) + (X, |R\rangle_d|1\rangle_p) \\ &- (X, |L\rangle_d|1\rangle_p) + (X, |R\rangle_d|3\rangle_p)]\end{aligned}$$

More Interesting Recursive Hardamard Walk

- Let $n \geq 2$. Program declared by recursive equation:

$$X \Leftarrow ((T_L[p]; X) \oplus_{H[d]} (T_R[p]; X)); (T_L[p] \oplus_{H[d]} T_R[p])^n$$

$$(X, |L\rangle_d|0\rangle_p) \to ...$$

$$\to \frac{1}{2\sqrt{2}}[(X, |L\rangle_d| - 3\rangle_p) + (X, |R\rangle_d| - 1\rangle_p)$$

$$+ (X, |L\rangle_d| - 1\rangle_p) - (X, |R\rangle_d|1\rangle_p)$$

$$+ (X, |L\rangle_d| - 1\rangle_p) + (X, |R\rangle_d|1\rangle_p)$$

$$- (X, |L\rangle_d|1\rangle_p) + (X, |R\rangle_d|3\rangle_p)]$$

- Configurations $-(X, |R\rangle_d|1\rangle_p)$ and $(X, |R\rangle_d|1\rangle_p)$ cancel one another.

Question: how to solve quantum recursive equations?

Syntactic Approximation

- Recursive program X declared by equation

$$X \Leftarrow F(X)$$

Syntactic Approximation

- Recursive program X declared by equation

$$X \Leftarrow F(X)$$

- Syntactic approximations:

$$\begin{cases} X^{(0)} = \textbf{Abort}, \\ X^{(n+1)} = F[X^{(n)}/X] \text{ for } n \geq 0. \end{cases}$$

$X^{(n)}$ is the nth syntactic approximation of X.

Syntactic Approximation

- Recursive program X declared by equation

$$X \Leftarrow F(X)$$

- Syntactic approximations:

$$\begin{cases} X^{(0)} = \textbf{Abort}, \\ X^{(n+1)} = F[X^{(n)}/X] \text{ for } n \geq 0. \end{cases}$$

  $X^{(n)}$ is the nth syntactic approximation of X.

- Semantics $[\![X]\!]$ of X is the limit:

$$[\![X]\!] = \lim_{n \to \infty} [\![X^{(n)}]\!]$$

Example - Recursive Hadamard Walk

$$X^{(0)} = \textbf{abort},$$

$$X^{(1)} = T_L[p] \oplus_{H[d]} (T_R[p]; \textbf{abort}),$$

$$X^{(2)} = T_L[p] \oplus_{H[d]} (T_R[p]; T_L[p] \oplus_{H[d_1]} (T_R[p]; \textbf{abort})),$$

$$X^{(3)} = T_L[p] \oplus_{H[d]} (T_R[p]; T_L[p] \oplus_{H[d_1]} (T_R[p]; T_L[p] \oplus_{H[d_2]} (T_R[p]; \textbf{abort}))),$$

.....

Example - Recursive Hadamard Walk

$$X^{(0)} = \mathbf{abort},$$
$$X^{(1)} = T_L[p] \oplus_{H[d]} (T_R[p]; \mathbf{abort}),$$
$$X^{(2)} = T_L[p] \oplus_{H[d]} (T_R[p]; T_L[p] \oplus_{H[d_1]} (T_R[p]; \mathbf{abort})),$$
$$X^{(3)} = T_L[p] \oplus_{H[d]} (T_R[p]; T_L[p] \oplus_{H[d_1]} (T_R[p]; T_L[p] \oplus_{H[d_2]} (T_R[p]; \mathbf{abort}))),$$
.....

Observation

- Continuously introduce new "coin" to avoid variable conflict.
- Variables $d, d_1, d_2, ...$ denote identical particles.

Example - Recursive Hadamard Walk

$$X^{(0)} = \mathbf{abort},$$
$$X^{(1)} = T_L[p] \oplus_{H[d]} (T_R[p]; \mathbf{abort}),$$
$$X^{(2)} = T_L[p] \oplus_{H[d]} (T_R[p]; T_L[p] \oplus_{H[d_1]} (T_R[p]; \mathbf{abort})),$$
$$X^{(3)} = T_L[p] \oplus_{H[d]} (T_R[p]; T_L[p] \oplus_{H[d_1]} (T_R[p]; T_L[p] \oplus_{H[d_2]} (T_R[p]; \mathbf{abort}))),$$
.....

Observation

- Continuously introduce new "coin" to avoid variable conflict.
- Variables $d, d_1, d_2, ...$ denote identical particles.
- The number of coin particles needed in running recursive walk is unknown beforehand.

Example - Recursive Hadamard Walk

$$X^{(0)} = \textbf{abort},$$

$$X^{(1)} = T_L[p] \oplus_{H[d]} (T_R[p]; \textbf{abort}),$$

$$X^{(2)} = T_L[p] \oplus_{H[d]} (T_R[p]; T_L[p] \oplus_{H[d_1]} (T_R[p]; \textbf{abort})),$$

$$X^{(3)} = T_L[p] \oplus_{H[d]} (T_R[p]; T_L[p] \oplus_{H[d_1]} (T_R[p]; T_L[p] \oplus_{H[d_2]} (T_R[p]; \textbf{abort}))),$$

.....

Observation

- Continuously introduce new "coin" to avoid variable conflict.
- Variables $d, d_1, d_2, ...$ denote identical particles.
- The number of coin particles needed in running recursive walk is unknown beforehand.
- We need to deal with *quantum systems where the number of particles of the same type may vary*.

Outline

Fock Spaces

- ***The principle of symmetrisation***: the states of n identical particles are either completely symmetric or completely antisymmetric with respect to the permutations of the particles. [bosons - *symmetric*; fermions - *antisymmetric*]

Fock Spaces

- ***The principle of symmetrisation***: the states of n identical particles are either completely symmetric or completely antisymmetric with respect to the permutations of the particles. [bosons - *symmetric*; fermions - *antisymmetric*]
- $\mathcal{H}$ — the Hilbert space of one particle.

Fock Spaces

- ***The principle of symmetrisation***: the states of n identical particles are either completely symmetric or completely antisymmetric with respect to the permutations of the particles. [bosons - *symmetric*; fermions - *antisymmetric*]
- $\mathcal{H}$ — the Hilbert space of one particle.
- For each permutation π of $1, ..., n$, define operator P_π in $\mathcal{H}^{\otimes n}$:

$$P_\pi |\psi_1 \otimes ... \otimes \psi_n\rangle = |\psi_{\pi(1)} \otimes ... \otimes \psi_{\pi(n)}\rangle$$

Fock Spaces

- ***The principle of symmetrisation***: the states of n identical particles are either completely symmetric or completely antisymmetric with respect to the permutations of the particles. [bosons - *symmetric*; fermions - *antisymmetric*]
- $\mathcal{H}$ — the Hilbert space of one particle.
- For each permutation π of $1, ..., n$, define operator P_π in $\mathcal{H}^{\otimes n}$:

$$P_\pi |\psi_1 \otimes ... \otimes \psi_n\rangle = |\psi_{\pi(1)} \otimes ... \otimes \psi_{\pi(n)}\rangle$$

- Define symmetrisation and antisymmetrisation operators in $\mathcal{H}^{\otimes n}$:

$$S_+ = \frac{1}{n!}\sum_\pi P_\pi, \qquad S_- = \frac{1}{n!}\sum_\pi (-1)^\pi P_\pi$$

Fock Spaces

$v = +$ for bosons, $v = -$ for fermions.

- Write

$$|\psi_1, ..., \psi_n\rangle_v = S_v |\psi_1 \otimes ... \otimes \psi_n\rangle.$$

Fock Spaces

$v = +$ for bosons, $v = -$ for fermions.

- Write
$$|\psi_1, ..., \psi_n\rangle_v = S_v|\psi_1 \otimes ... \otimes \psi_n\rangle.$$
- State spaces of n bosons and fermions:
$$\mathcal{H}_v^{\otimes n} = S_v\mathcal{H}^{\otimes n} = \text{span}\{|\psi_1, ..., \psi_n\rangle_v : |\psi_1\rangle, ..., |\psi_n\rangle \text{ are in } \mathcal{H}\}$$

Fock Spaces

$v = +$ for bosons, $v = -$ for fermions.

- Write
$$|\psi_1, ..., \psi_n\rangle_v = S_v |\psi_1 \otimes ... \otimes \psi_n\rangle.$$
- State spaces of n bosons and fermions:
$$\mathcal{H}_v^{\otimes n} = S_v \mathcal{H}^{\otimes n} = \text{span}\{|\psi_1, ..., \psi_n\rangle_v : |\psi_1\rangle, ..., |\psi_n\rangle \text{ are in } \mathcal{H}\}$$
- Vacuum state $|\mathbf{0}\rangle$
$$\mathcal{H}_v^{\otimes 0} = \mathcal{H}^{\otimes 0} = \text{span}\{|\mathbf{0}\rangle\}$$

Fock Spaces

$v = +$ for bosons, $v = -$ for fermions.

- Write
$$|\psi_1, ..., \psi_n\rangle_v = S_v |\psi_1 \otimes ... \otimes \psi_n\rangle.$$
- State spaces of n bosons and fermions:
$$\mathcal{H}_v^{\otimes n} = S_v \mathcal{H}^{\otimes n} = \text{span}\{|\psi_1, ..., \psi_n\rangle_v : |\psi_1\rangle, ..., |\psi_n\rangle \text{ are in } \mathcal{H}\}$$
- Vacuum state $|\mathbf{0}\rangle$
$$\mathcal{H}_v^{\otimes 0} = \mathcal{H}^{\otimes 0} = \text{span}\{|\mathbf{0}\rangle\}$$
- Space of the states of variable particle number is Fock space:
$$\mathcal{F}_v(\mathcal{H}) = \sum_{n=0}^{\infty} \mathcal{H}_v^{\otimes n}$$

Fock Spaces

$v = +$ for bosons, $v = -$ for fermions.

- Write
$$|\psi_1, ..., \psi_n\rangle_v = S_v |\psi_1 \otimes ... \otimes \psi_n\rangle.$$
- State spaces of n bosons and fermions:
$$\mathcal{H}_v^{\otimes n} = S_v \mathcal{H}^{\otimes n} = \text{span}\{|\psi_1, ..., \psi_n\rangle_v : |\psi_1\rangle, ..., |\psi_n\rangle \text{ are in } \mathcal{H}\}$$
- Vacuum state $|\mathbf{0}\rangle$
$$\mathcal{H}_v^{\otimes 0} = \mathcal{H}^{\otimes 0} = \text{span}\{|\mathbf{0}\rangle\}$$
- Space of the states of variable particle number is Fock space:
$$\mathcal{F}_v(\mathcal{H}) = \sum_{n=0}^{\infty} \mathcal{H}_v^{\otimes n}$$
- Free Fock space:
$$\mathcal{F}(\mathcal{H}) = \sum_{n=0}^{\infty} \mathcal{H}^{\otimes n}$$

Evolution Fock Spaces

- Evolution of one particle: unitary operator U.

Evolution Fock Spaces

- Evolution of one particle: unitary operator U.
- Evolution of n particles without mutual interactions: operator $\mathbf{U}$ in $\mathcal{H}^{\otimes n}$:

$$\mathbf{U}|\psi_1 \otimes ... \otimes \psi_n\rangle = |U\psi_1 \otimes ... \otimes U\psi_n\rangle$$

Evolution Fock Spaces

- Evolution of one particle: unitary operator U.
- Evolution of n particles without mutual interactions: operator $\mathbf{U}$ in $\mathcal{H}^{\otimes n}$:

$$\mathbf{U}|\psi_1 \otimes ... \otimes \psi_n\rangle = |U\psi_1 \otimes ... \otimes U\psi_n\rangle$$

- Symmetrisation:

$$\mathbf{U}|\psi_1, ..., \psi_n\rangle_v = |U\psi_1, ...U\psi_n\rangle_v.$$

Evolution Fock Spaces

- Evolution of one particle: unitary operator U.
- Evolution of n particles without mutual interactions: operator $\mathbf{U}$ in $\mathcal{H}^{\otimes n}$:

$$\mathbf{U}|\psi_1 \otimes ... \otimes \psi_n\rangle = |U\psi_1 \otimes ... \otimes U\psi_n\rangle$$

- Symmetrisation:

$$\mathbf{U}|\psi_1, ..., \psi_n\rangle_v = |U\psi_1, ...U\psi_n\rangle_v.$$

- Extend to Fock spaces $\mathcal{F}_v(\mathcal{H})$ and $\mathcal{F}(\mathcal{H})$:

$$\mathbf{U}\left(\sum_{n=0}^{\infty} |\Psi(n)\rangle\right) = \sum_{n=0}^{\infty} \mathbf{U}|\Psi(n)\rangle$$

Creation and Annihilation of Particles

- Transitions between states of different particle numbers.

Creation and Annihilation of Particles

- Transitions between states of different particle numbers.
- Creation operator $a^*(\psi)$ in $\mathcal{F}_v(\mathcal{H})$:

$$a^*(\psi)|\psi_1, ..., \psi_n\rangle_v = \sqrt{n+1}|\psi, \psi_1, ..., \psi_n\rangle_v$$

Add a particle in individual state $|\psi\rangle$ to the system of n particles without modifying their respective states.

Creation and Annihilation of Particles

- Transitions between states of different particle numbers.
- Creation operator $a^*(\psi)$ in $\mathcal{F}_v(\mathcal{H})$:

$$a^*(\psi)|\psi_1, ..., \psi_n\rangle_v = \sqrt{n+1}|\psi, \psi_1, ..., \psi_n\rangle_v$$

Add a particle in individual state $|\psi\rangle$ to the system of n particles without modifying their respective states.

- Annihilation operator $a(\psi)$ — Hermitian conjugate of $a^*(\psi)$:

$$a(\psi)|\mathbf{0}\rangle = 0,$$

$$a(\psi)|\psi_1, ..., \psi_n\rangle_v = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} (v)^{i-1} \langle\psi|\psi_i\rangle |\psi_1, ..., \psi_{i-1}, \psi_{i+1}, ..., \psi_n\rangle_v$$

Decrease the number of particles by one unit, while preserving the symmetry of state.

Outline

Second quantisation provides us with necessary tools for defining semantics of quantum recursions!

Second quantisation provides us with necessary tools for defining semantics of quantum recursions!

A domain of operators in free Fock space

- $\mathcal{H}$ and $\mathcal{K}$ — two Hilbert spaces
- $\mathcal{F}(\mathcal{H})$ — free Fock space over $\mathcal{H}$.

Second quantisation provides us with necessary tools for defining semantics of quantum recursions!

A domain of operators in free Fock space

- $\mathcal{H}$ and $\mathcal{K}$ — two Hilbert spaces
- $\mathcal{F}(\mathcal{H})$ — free Fock space over $\mathcal{H}$.
- $\mathcal{O}(\mathcal{F}(\mathcal{H}) \otimes \mathcal{K})$ — the set of all operators of the form

$$\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n),$$

where $\mathbf{A}(n)$ is an operator in $\mathcal{H}^{\otimes n} \otimes \mathcal{K}$.

Order and Operations on $\mathcal{O}(\mathcal{F}(\mathcal{H}) \otimes \mathcal{K})$

- Flat order: $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n), \mathbf{B} = \sum_{n=0}^{\infty} \mathbf{B}(n)$

Order and Operations on $\mathcal{O}(\mathcal{F}(\mathcal{H}) \otimes \mathcal{K})$

- Flat order: $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n), \mathbf{B} = \sum_{n=0}^{\infty} \mathbf{B}(n)$
  - $\mathbf{A} \sqsubseteq \mathbf{B}$ if and only if

Order and Operations on $\mathcal{O}(\mathcal{F}(\mathcal{H}) \otimes \mathcal{K})$

- Flat order: $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n), \mathbf{B} = \sum_{n=0}^{\infty} \mathbf{B}(n)$
  - $\mathbf{A} \sqsubseteq \mathbf{B}$ if and only if
    - either for all $n \geq 0$, $\mathbf{A}(n) = \mathbf{B}(n)$, or

Order and Operations on $\mathcal{O}(\mathcal{F}(\mathcal{H}) \otimes \mathcal{K})$

- Flat order: $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n), \mathbf{B} = \sum_{n=0}^{\infty} \mathbf{B}(n)$
  - $\mathbf{A} \sqsubseteq \mathbf{B}$ if and only if
    - either for all $n \geq 0$, $\mathbf{A}(n) = \mathbf{B}(n)$, or
    - there exists an integer n_0 such that $\mathbf{A}(n) = \mathbf{B}(n)$ for all $0 \leq n \leq n_0$ and $\mathbf{A}(n) = 0$ for all $n > n_0$.

Order and Operations on $\mathcal{O}(\mathcal{F}(\mathcal{H}) \otimes \mathcal{K})$

- Flat order: $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n), \mathbf{B} = \sum_{n=0}^{\infty} \mathbf{B}(n)$
  - $\mathbf{A} \sqsubseteq \mathbf{B}$ if and only if
    - either for all $n \geq 0$, $\mathbf{A}(n) = \mathbf{B}(n)$, or
    - there exists an integer n_0 such that $\mathbf{A}(n) = \mathbf{B}(n)$ for all $0 \leq n \leq n_0$ and $\mathbf{A}(n) = 0$ for all $n > n_0$.
- Product:

$$\mathbf{A} \cdot \mathbf{B} = \sum_{n=0}^{\infty} \left(\mathbf{A}(n) \cdot \mathbf{B}(n) \right).$$

Order and Operations on $\mathcal{O}(\mathcal{F}(\mathcal{H}) \otimes \mathcal{K})$

- Flat order: $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n), \mathbf{B} = \sum_{n=0}^{\infty} \mathbf{B}(n)$
  - $\mathbf{A} \sqsubseteq \mathbf{B}$ if and only if
    - either for all $n \geq 0$, $\mathbf{A}(n) = \mathbf{B}(n)$, or
    - there exists an integer n_0 such that $\mathbf{A}(n) = \mathbf{B}(n)$ for all $0 \leq n \leq n_0$ and $\mathbf{A}(n) = 0$ for all $n > n_0$.
- Product:

$$\mathbf{A} \cdot \mathbf{B} = \sum_{n=0}^{\infty} \left(\mathbf{A}(n) \cdot \mathbf{B}(n) \right).$$

- Guarded composition:

$$\square_i \left(|i\rangle \rightarrow \mathbf{A}_i \right) = \sum_{n=0}^{\infty} \left(\sum_i \left(|i\rangle \langle i| \otimes \mathbf{A}_i(n) \right) \right).$$

Notation

- C — the set of quantum coins in$P = P[X_1, ..., X_m]$.

Notation

- C — the set of quantum coins in$P = P[X_1, ..., X_m]$.
- $\mathcal{H}_C = \bigotimes_{c \in C} \mathcal{H}_c$, where $\mathcal{H}_c$ is the Hilbert space of coin c.

Notation

- C — the set of quantum coins in$P = P[X_1, ..., X_m]$.
- $\mathcal{H}_C = \bigotimes_{c \in C} \mathcal{H}_c$, where $\mathcal{H}_c$ is the Hilbert space of coin c.
- The principal system of P is the composition of the systems denoted by principal variables in P.

Notation

- C — the set of quantum coins in$P = P[X_1, ..., X_m]$.
- $\mathcal{H}_C = \bigotimes_{c \in C} \mathcal{H}_c$, where $\mathcal{H}_c$ is the Hilbert space of coin c.
- The principal system of P is the composition of the systems denoted by principal variables in P.
- $\mathcal{H}$ — the Hilbert space of the principal system.

Semantic functional

Semantic functional of program scheme P:

$$[\![P]\!] : \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})^m \to \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H}).$$

For any $\mathbf{A}_1, ..., \mathbf{A}_m \in \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})$,

$$[\![P]\!](\mathbf{A}_1, ..., \mathcal{A}_m)$$

is inductively defined:

- If $P =$ **abort**, then $[\![P]\!](\mathbf{A}_1, ..., \mathbf{A}_m)$ is the zero operator $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$ with $\mathbf{A}(n) = 0$ (the zero operator in $\mathcal{H}_C^{\otimes n} \otimes \mathcal{H}$);

Semantic functional

Semantic functional of program scheme P:

$$[\![P]\!] : \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})^m \to \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H}).$$

For any $\mathbf{A}_1, ..., \mathbf{A}_m \in \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})$,

$$[\![P]\!](\mathbf{A}_1, ..., \mathcal{A}_m)$$

is inductively defined:

- If $P =$ **abort**, then $[\![P]\!](\mathbf{A}_1, ..., \mathbf{A}_m)$ is the zero operator $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$ with $\mathbf{A}(n) = 0$ (the zero operator in $\mathcal{H}_C^{\otimes n} \otimes \mathcal{H}$);
- If $P =$ **skip**, then $[\![P]\!](\mathbf{A}_1, ..., \mathbf{A}_m)$ is the identity operator $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$ with $\mathbf{A}(n) = I$ (the identity operator in $\mathcal{H}_C^{\otimes n} \otimes \mathcal{H}$);

Semantic functional

Semantic functional of program scheme P:

$$[\![P]\!] : \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})^m \to \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H}).$$

For any $\mathbf{A}_1, ..., \mathbf{A}_m \in \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})$,

$$[\![P]\!](\mathbf{A}_1, ..., \mathcal{A}_m)$$

is inductively defined:

- If $P =$ **abort**, then $[\![P]\!](\mathbf{A}_1, ..., \mathbf{A}_m)$ is the zero operator $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$ with $\mathbf{A}(n) = 0$ (the zero operator in $\mathcal{H}_C^{\otimes n} \otimes \mathcal{H}$);
- If $P =$ **skip**, then $[\![P]\!](\mathbf{A}_1, ..., \mathbf{A}_m)$ is the identity operator $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$ with $\mathbf{A}(n) = I$ (the identity operator in $\mathcal{H}_C^{\otimes n} \otimes \mathcal{H}$);
- If $P = U[\overline{q}, \overline{c}]$, then $[\![P]\!](\mathbf{A}_1, ..., \mathbf{A}_m)$ is the cylindrical extension of U: $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$ with $\mathbf{A}(n) = U \otimes I_1 \otimes I_2 \otimes I_3$, where I_1 is the identity operator in the Hilbert space of those coins not in $\overline{c}$, I_2 is the identity operator in $\mathcal{H}_C^{\otimes(n-1)}$, and I_3 is the identity operator in the Hilbert space of those principal variables not in $\overline{q}$;

Semantic functional

- If $P = X_j$ $(1 \leq j \leq m)$, then $[\![P]\!](\mathbf{A}_1, ..., \mathbf{A}_m) = \mathbf{A}_j$;

  If $P = P_1; P_2$, then

  $$[\![P]\!](\mathbf{A}_1, ..., \mathbf{A}_m) = [\![P_2]\!](\mathbf{A}_1, ..., \mathbf{A}_m) \cdot [\![P_1]\!](A_1, ..., A_m);$$

  If $P = \mathbf{qif}\ [c](\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq}$, then

  $$[\![P]\!](\mathbf{A}_1, ..., \mathbf{A}_m) = \square_i \left(|i\rangle \to [\![P_i]\!](\mathbf{A}_1, ..., \mathbf{A}_m)\right).$$

Theorem — Continuity of Semantic Functionals

Semantic functional

$$[\![P]\!] : (\mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})^m, \sqsubseteq) \to (\mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$$

is continuous.

Creation functional

- Creation functional

$$C : \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H}) \to \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})$$

is defined: for any $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$,

$$C(\mathbf{A}) = \sum_{n=0}^{\infty} (I \otimes \mathbf{A}(n))$$

where I is the identity operator in $\mathcal{H}_C$.

Creation functional

- Creation functional

$$C : \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H}) \to \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})$$

  is defined: for any $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$,

$$C(\mathbf{A}) = \sum_{n=0}^{\infty} (I \otimes \mathbf{A}(n))$$

  where I is the identity operator in $\mathcal{H}_C$.

- *Intuition* — creation functional C moves all coins $c_0, c_1, c_2, \ldots$ one position to the right so that c_i becomes c_{i+1} for all $i = 0, 1, 2, \ldots$. Thus, a new position is created at the left end for a new coin c_0.

Lemma — Continuity of Creation Functional

Creation functional

$$C : (\mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq) \to (\mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$$

is continuous.

Corollary

$P = P[X_1, ..., X_m]$ a program scheme. Define:

$$(C^{\otimes m} \circ [\![P]\!])(\mathbf{A}_1, ..., \mathbf{A}_m) = [\![P]\!](C(\mathbf{A}_1), ..., C(\mathbf{A}_m)).$$

Then functional

$$C^{\otimes m} \circ [\![P]\!] : (\mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})^{\otimes m}, \sqsubseteq) \to (\mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$$

is continuous.

Corollary

$P = P[X_1, ..., X_m]$ a program scheme. Define:

$$(C^{\otimes m} \circ [\![P]\!])(\mathbf{A}_1, ..., \mathbf{A}_m) = [\![P]\!](C(\mathbf{A}_1), ..., C(\mathbf{A}_m)).$$

Then functional

$$C^{\otimes m} \circ [\![P]\!] : (\mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})^{\otimes m}, \sqsubseteq) \to (\mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$$

is continuous.

Notation

- Consider a recursive program P declared by system of recursive equations:

$$S : \begin{cases} X_1 \Leftarrow P_1, \\ \quad ..... \\ X_m \Leftarrow P_m, \end{cases}$$

Notation

- System S of recursive equations induces semantic functional:

$$[\![S]\!] : \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})^{\otimes m} \to \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})^m,$$
$$[\![S]\!](\mathbf{A}_1, ..., \mathbf{A}_m) = ((C^m \circ [\![P_1]\!])(\mathbf{A}_1, ..., \mathbf{A}_m), ..., (C^m \circ [\![P_m]\!])(\mathbf{A}_1, ..., \mathbf{A}_m))$$

Notation

- System S of recursive equations induces semantic functional:

$$[\![S]\!] : \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})^{\otimes m} \to \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})^m,$$
$$[\![S]\!](\mathbf{A}_1, ..., \mathbf{A}_m) = ((C^m \circ [\![P_1]\!])(\mathbf{A}_1, ..., \mathbf{A}_m), ..., (C^m \circ [\![P_m]\!])(\mathbf{A}_1, ..., \mathbf{A}_m))$$

- Semantical functional

$$[\![S]\!] : (\mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})^{\otimes m}, \sqsubseteq) \to (\mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})^m, \sqsubseteq)$$

is continuous.

Fixed point semantics

- Knaster-Tarski Fixed Point Theorem: $[\![S]\!]$ has the least fixed point $\mu[\![S]\!]$.

Fixed point semantics

- Knaster-Tarski Fixed Point Theorem: $[\![S]\!]$ has the least fixed point $\mu[\![S]\!]$.

Definition

The fixed point semantics of recursive program P declared by S:

$$[\![P]\!]_{fix} = [\![P]\!](\mathbf{A}_1^*, ..., \mathbf{A}_m^*) \in \mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})$$

where $\mu[\![S]\!] = (\mathbf{A}_1^*, ..., \mathbf{A}_m^*)$.

Theorem — Equivalence of Denotational Semantics and Operational Semantics

1. For each k, $\{[\![X_k^{(n)}]\!]\}_{n=0}^{\infty}$ is an increasing chain and

$$[\![X_k^{(\infty)}]\!] \stackrel{\triangle}{=} \lim_{n\to\infty} [\![X_k^{(n)}]\!] = \bigsqcup_{n=0}^{\infty} [\![X_k^{(n)}]\!]$$

exists in $\mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})$.

Theorem — Equivalence of Denotational Semantics and Operational Semantics

1. For each k, $\{[\![X_k^{(n)}]\!]\}_{n=0}^{\infty}$ is an increasing chain and

$$[\![X_k^{(\infty)}]\!] \stackrel{\triangle}{=} \lim_{n\to\infty} [\![X_k^{(n)}]\!] = \bigsqcup_{n=0}^{\infty} [\![X_k^{(n)}]\!]$$

   exists in $\mathcal{O}(\mathcal{F}(\mathcal{H}_C) \otimes \mathcal{H})$.

2. $([\![X_1^{(\infty)}]\!], ..., [\![X_m^{(\infty)}]\!]) = \mu[\![S]\!]$ is the least fixed point of semantic functional $[\![S]\!]$.

Outline

Solutions of recursive equations in Boson/Fermion Fock space

Symmetrisation/anti-symmetrisation of the solutions of recursive equations in free Fock space!

Solutions of recursive equations in Boson/Fermion Fock space

Symmetrisation/anti-symmetrisation of the solutions of recursive equations in free Fock space!

Principal System Semantics

- Each state $|\Psi\rangle$ in Fock space $\mathcal{F}_v(\mathcal{H}_d)$ induces mapping:

  $$[\![X, \Psi]\!]_p : \text{pure states} \to \text{partial density operators in } \mathcal{H}_p$$
  $$[\![X, \Psi]\!]_p(|\psi\rangle) = tr_{\mathcal{F}(\mathcal{H}_d)}(|\Phi\rangle\langle\Phi|)$$

  where $|\Phi\rangle = [\![X]\!](|\Psi\rangle \otimes |\psi\rangle)$

- Principal system semantics of X with coin initialisation $|\Psi\rangle$: mapping $[\![X, \Psi]\!]_p$.

Example — Recursive Quantum Walk

$$\begin{cases} X \Leftarrow T_L[p] \oplus_{H[d]} (T_R[p]; Y), \\ Y \Leftarrow (T_L[p]; X) \oplus_{H[d]} T_R[p] \end{cases}$$

- Coherent state of bosons in Boson Fock space $\mathcal{F}_+(\mathcal{H})$:

$$|\psi\rangle_{\text{coh}} = \exp\left(-\frac{1}{2}\langle\psi|\psi\rangle\right) \sum_{n=0}^{\infty} \frac{[a^*(\psi)]^n}{n!}|\mathbf{0}\rangle$$

Example — Recursive Quantum Walk

$$\begin{cases} X \Leftarrow T_L[p] \oplus_{H[d]} (T_R[p]; Y), \\ Y \Leftarrow (T_L[p]; X) \oplus_{H[d]} T_R[p] \end{cases}$$

- Coherent state of bosons in Boson Fock space $\mathcal{F}_+(\mathcal{H})$:

$$|\psi\rangle_{\text{coh}} = \exp\left(-\frac{1}{2}\langle\psi|\psi\rangle\right) \sum_{n=0}^{\infty} \frac{[a^*(\psi)]^n}{n!}|\mathbf{0}\rangle$$

- The walk starts from position 0 and the coins are initialised in the coherent states of bosons corresponding to $|L\rangle$:

$$\begin{aligned} [\![X, L_{\text{coh}}]\!]_p(|0\rangle) &= \frac{1}{\sqrt{e}}\left(\sum_{k=0}^{\infty} \frac{1}{2^{2k+1}}|-1\rangle\langle -1| + \sum_{k=0}^{\infty} \frac{1}{2^{2k+2}}|2\rangle\langle 2|\right) \\ &= \frac{1}{\sqrt{e}}\left(\frac{2}{3}|-1\rangle\langle -1| + \frac{1}{3}|2\rangle\langle 2|\right). \end{aligned}$$

Outline

Problems:

- What kind of problems can be solved more conveniently by using quantum recursion?

Problems:

- What kind of problems can be solved more conveniently by using quantum recursion?
- Hoare logic for quantum while-loops defined using quantum coins?

Problems:

- What kind of problems can be solved more conveniently by using quantum recursion?
- Hoare logic for quantum while-loops defined using quantum coins?
  - Hoare logic for quantum **while**-programs with classical controls [Ying, *TOPLAS'*2011]

Problems:

- What kind of problems can be solved more conveniently by using quantum recursion?
- Hoare logic for quantum while-loops defined using quantum coins?
  - Hoare logic for quantum **while**-programs with classical controls [Ying, *TOPLAS'*2011]
- What kind of physical systems can be used to implement quantum recursion?

Thank You!