Foundations of Quantum Programming

Mingsheng Ying

University of Technology Sydney, Australia
Institute of Software, Chinese Academy of Sciences
Tsinghua University, China
Outline

Introduction

Syntax of Quantum Programs

Operational Semantics

Denotational Semantics

(Floyd-)Hoare Logic for Quantum Programs

Research Problems
Outline

Introduction

Syntax of Quantum Programs

Operational Semantics

Denotational Semantics

(Floyd-)Hoare Logic for Quantum Programs

Research Problems
How to program quantum computers?

- Quantum algorithms:
  
  Deutsch-Josza, Grover, Shor, HHL, ...
  
  Quantum algorithm zoo (http://math.nist.gov/quantum/zoo/)
How to program quantum computers?

- **Quantum algorithms:**
  - Deutsch-Josza, Grover, Shor, HHL, ...
  - Quantum algorithm zoo (http://math.nist.gov/quantum/zoo/)

- **Quantum computers:**
  - IBM Q (20 qubits — Nov 2017; 50 qubits — 2018)
  - Google (49 qubits — 2018)
  - Intel (17 qubits — Oct 2017)
Quantum programming languages

- Q# @Microsoft
Quantum programming languages

- Q# @Microsoft
- Open Fermion @Google
Quantum programming languages

- Q# @Microsoft
- Open Fermion @Google
- Quil @Riggeti
Quantum programming languages

- Q# @Microsoft
- Open Fermion @Google
- Quil @Riggeti
- Quipper @Dalhousie
Quantum programming languages

- Q# @Microsoft
- Open Fermion @Google
- Quil @Riggeti
- Quipper @Dalhousie
- Scaffold @Princeton
Quantum programming languages

- Q# @Microsoft
- Open Fermion @Google
- Quil @Riggeti
- Quipper @Dalhousie
- Scaffold @Princeton
- Project Q @ETH
Quantum programming languages

- Q@Microsoft
- Open Fermion @Google
- Quil @Riggeti
- Quipper @Dalhousie
- Scaffold @Princeton
- Project Q @ETH
- QWire @UPenn
Quantum programming languages

- Q# @Microsoft
- Open Fermion @Google
- Quil @Riggeti
- Quipper @Dalhousie
- Scaffold @Princeton
- Project Q @ETH
- QWire @UPenn
- Q|SI> @UTS
This lecture

- **Principles** underlying *all* of the quantum programming languages
This lecture

- **Principles** underlying *all* of the quantum programming languages
- **Not** the languages themselves.

This lecture

- **Principles** underlying *all* of the quantum programming languages
- **Not** the languages themselves.

- **Reference:**
Programming languages and tools

Programming languages are notations used for specifying, organising and reasoning about computations.

[R. Sethi, *Programming languages: Concepts and Constructs*]

- Semantics
Programming languages and tools

Programming languages are notations used for specifying, organising and reasoning about computations.

[R. Sethi, Programming languages: Concepts and Constructs]

- Semantics
- Turing-complete?
Programming languages and tools

Programming languages are notations used for specifying, organising and reasoning about computations.

[R. Sethi, *Programming languages: Concepts and Constructs*]

- Semantics
- Turing-complete?
- Compilers
Programming languages and tools

Programming languages are notations used for specifying, organising and reasoning about computations.

[R. Sethi, Programming languages: Concepts and Constructs]

- Semantics
- Turing-complete?
- Compilers
- Program analysis: Termination, ...
Programming languages and tools

Programming languages are notations used for specifying, organising and reasoning about computations.

[R. Sethi, Programming languages: Concepts and Constructs]

- Semantics
- Turing-complete?
- Comiplers
- Program analysis: Termination, ...
- How to verify correctness of your programs?
Programming languages and tools

Programming languages are notations used for specifying, organising and reasoning about computations.

[R. Sethi, Programming languages: Concepts and Constructs]

- Semantics
- Turing-complete?
- Compilers
- Program analysis: Termination, ...
- How to verify correctness of your programs?
- ......
Quantum circuit vs Quantum programs

**Example:** Quantum walk on a circle with an absorbing boundary.

- $H_d = \text{span}\{|L\rangle, |R\rangle\}$ — the *direction* space, $|L\rangle$ and $|R\rangle$ indicate directions left and right, respectively.
Quantum circuit vs Quantum programs

Example: Quantum walk on a circle with an absorbing boundary.

- $H_d = \text{span}\{|L\rangle, |R\rangle\}$ — the direction space, $|L\rangle$ and $|R\rangle$ indicate directions left and right, respectively.
- $H_p = \text{span}\{|0\rangle, |1\rangle, ..., |n - 1\rangle\}$ — the position space with orthonormal basis states, the vector $|i\rangle$ denotes position $i$ for each $0 \leq i \leq n - 1$. 
Quantum circuit vs Quantum programs

Example: Quantum walk on a circle with an absorbing boundary.

- $H_d = \text{span}\{ |L\rangle, |R\rangle \}$ — the direction space, $|L\rangle$ and $|R\rangle$ indicate directions left and right, respectively.

- $H_p = \text{span}\{ |0\rangle, |1\rangle, ..., |n − 1\rangle \}$ — the position space with orthonormal basis states, the vector $|i\rangle$ denotes position $i$ for each $0 \leq i \leq n − 1$.

- The state space of the walk — $H = H_p \otimes H_d$. 
Quantum circuit vs Quantum programs

Example: Quantum walk on a circle with an absorbing boundary.

- $H_d = \text{span}\{ |L\rangle, |R\rangle \}$ — the direction space, $|L\rangle$ and $|R\rangle$ indicate directions left and right, respectively.
- $H_p = \text{span}\{ |0\rangle, |1\rangle, ..., |n-1\rangle \}$ — the position space with orthonormal basis states, the vector $|i\rangle$ denotes position $i$ for each $0 \leq i \leq n-1$.
- The state space of the walk — $H = H_p \otimes H_d$.
- The initial state — $|0\rangle_p |L\rangle_d$. 
Quantum circuit vs Quantum programs

- Each step of the walk:
  
  1. Measure the system to see whether the current position is 1 (absorbing boundary). If “yes”, terminates; otherwise, continues:
     
     \[ M = \begin{cases} M_{\text{yes}} = |1\rangle \langle 1| \otimes I_d, \\ M_{\text{no}} = I - M_{\text{yes}} \end{cases} ; \]
  
  2. A “coin-tossing” operator \( C = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \) is applied on the direction space;
  
  3. A shift operator \( S = n - 1 \sum_{i=0}^{n-1} |i\rangle \langle i| \otimes |L\rangle \langle L| + n - 1 \sum_{i=0}^{n-1} |i\rangle \langle i| \otimes |R\rangle \langle R| \) is performed on the state space \( H \).

- Question: How to specify it in the circuit language?
Quantum circuit vs Quantum programs

▶ Each step of the walk:

1. Measure the system to see whether the current position is 1 (absorbing boundary). If “yes”, terminates; otherwise, continues:

\[ \mathcal{M} = \{M_{yes} = |1\rangle\langle 1| \otimes I, M_{no} = I - M_{yes}\}; \]
Quantum circuit vs Quantum programs

- Each step of the walk:
  1. Measure the system to see whether the current position is 1 (absorbing boundary). If “yes”, terminates; otherwise, continues:

\[
\mathcal{M} = \{M_{yes} = |1\rangle \langle 1| \otimes I_d, M_{no} = I - M_{yes}\};
\]

  2. A “coin-tossing” operator

\[
C = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}
\]

is applied on the direction space;
Quantum circuit vs Quantum programs

- Each step of the walk:
  1. Measure the system to see whether the current position is 1 (absorbing boundary). If “yes”, terminates; otherwise, continues:

\[
\mathcal{M} = \{M_{yes} = |1\rangle\langle 1| \otimes I_d, M_{no} = I - M_{yes}\};
\]

  2. A “coin-tossing” operator

\[
C = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}
\]

is applied on the direction space;

  3. A shift operator

\[
S = \sum_{i=0}^{n-1} |i \oplus 1\rangle \langle i| \otimes |L\rangle \langle L| + \sum_{i=0}^{n-1} |i \oplus 1\rangle \langle i| \otimes |R\rangle \langle R|
\]

is performed on the state space \(H\).
Quantum circuit vs Quantum programs

- Each step of the walk:
  1. Measure the system to see whether the current position is 1 (absorbing boundary). If “yes”, terminates; otherwise, continues:

\[ \mathcal{M} = \{ M_{yes} = |1\rangle \langle 1| \otimes I_d, M_{no} = I - M_{yes} \} ; \]

  2. A “coin-tossing” operator

\[ C = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \]

is applied on the direction space;

  3. A shift operator

\[ S = \sum_{i=0}^{n-1} |i \oplus 1\rangle \langle i| \otimes |L\rangle \langle L| + \sum_{i=0}^{n-1} |i \oplus 1\rangle \langle i| \otimes |R\rangle \langle R| \]

is performed on the state space \( H \).

- **Question:** How to specify it in the circuit language?
Outline

Introduction

Syntax of Quantum Programs

Operational Semantics

Denotational Semantics

(Floyd-)Hoare Logic for Quantum Programs

Research Problems
Classical **while**-Language

\[
S ::= \text{skip} \mid u := t \mid S_1; S_2 \mid \text{if } b \text{ then } S_1 \text{ else } S_2 \text{ fi} \\
\mid \text{while } b \text{ do } S \text{ od.}
\]

- Conditional statement can be generalised to case statement:

\[
\text{if } G_1 \rightarrow S_1 \\
\square G_2 \rightarrow S_2 \\
\cdots \\
\square G_n \rightarrow S_n \\
\text{fi}
\]

or more compactly:

\[
\text{if } (\square i \cdot G_i \rightarrow S_i) \text{ fi}
\]
Quantum **while**-Language

- **Alphabet**: a countably infinite set $Var$ of quantum variables $q, q', q_0, q_1, q_2, \ldots$. 
Quantum \textbf{while}-Language

- **Alphabet**: a countably infinite set $\text{Var}$ of quantum variables $q, q', q_0, q_1, q_2$, ....
- Each quantum variable $q \in \text{Var}$ has a type $\mathcal{H}_q$ (a Hilbert space), e.g. $\text{Boolean} = \mathcal{H}_2$, $\text{integer} = \mathcal{H}_\infty$. 
Quantum \textbf{while-}Language

\begin{itemize}
  \item \textbf{Alphabet}: a countably infinite set $\text{Var}$ of quantum variables $q, q', q_0, q_1, q_2, \ldots$.  \\
  \item Each quantum variable $q \in \text{Var}$ has a \textbf{type} $\mathcal{H}_q$ (a Hilbert space), e.g.  \\ 
  \begin{align*}
    \text{Boolean} &= \mathcal{H}_2, \quad \text{integer} = \mathcal{H}_\infty.
  \end{align*}
  \\
  \item A quantum \textbf{register} is a finite sequence $\bar{q} = q_1, \ldots, q_n$ of distinct quantum variables. State Hilbert space:
  
  \begin{equation}
    \mathcal{H}_{\bar{q}} = \bigotimes_{i=1}^{n} \mathcal{H}_{q_i}.
  \end{equation}
\end{itemize}
Syntax of Quantum Programs

\[
S ::= \text{skip} | q ::= |0\rangle | q ::= U[\bar{q}] | S_1 ; S_2 \\
    | \text{if} (\square m \cdot M[\bar{q}] = m \rightarrow S_m) \text{ fi} \\
    | \text{while } M[\bar{q}] = 1 \text{ do } S \text{ od}.
\]
Syntax of Quantum Programs

\[ S ::= \text{skip} \mid q := \lvert 0 \rangle \mid \overline{q} := U[\overline{q}] \mid S_1; S_2 \]
\[ \mid \text{if} (\Box m \cdot M[\overline{q}] = m \rightarrow S_m) \text{ fi} \]
\[ \mid \text{while} M[\overline{q}] = 1 \text{ do } S \text{ od}. \]

Exercise 1

Write quantum walk as a program in quantum while-language.
Outline

Introduction

Syntax of Quantum Programs

Operational Semantics

Denotational Semantics

(Floyd-)Hoare Logic for Quantum Programs

Research Problems
Notations

- **partial density operator**: positive operator $\rho$, $tr(\rho) \leq 1$. 

$D(H)$ — the set of partial density operators in $H$.

State Hilbert space of all quantum variables: $H_{all} = \bigotimes_{q \in \text{Var}} H_q$.

$E$ — empty program; i.e. termination.

Configuration: pair $\langle S, \rho \rangle$, where:

1. $S$ is a quantum program or the empty program $E$;
2. $\rho \in D(H_{all})$, denoting the (global) state of quantum variables.

Transition: $\langle S, \rho \rangle \rightarrow \langle S', \rho' \rangle$.
Notations

- **partial density operator**: positive operator \( \rho, \text{tr}(\rho) \leq 1 \).
- \( \mathcal{D}(\mathcal{H}) \) — the set of partial density operators in \( \mathcal{H} \).
Notations

- **partial density operator**: positive operator $\rho$, $tr(\rho) \leq 1$.
- $D(\mathcal{H})$ — the set of partial density operators in $\mathcal{H}$.
- State Hilbert space of all quantum variables:
  \[
  \mathcal{H}_{all} = \bigotimes_{q \in \text{Var}} \mathcal{H}_q.
  \]
Notations

- **partial density operator**: positive operator $\rho$, $tr(\rho) \leq 1$.
- $D(\mathcal{H})$ — the set of partial density operators in $\mathcal{H}$.
- State Hilbert space of all quantum variables:
  \[ \mathcal{H}_{all} = \bigotimes_{q \in \text{Var}} \mathcal{H}_q. \]
- $E$ — empty program; i.e. termination.
Notations

- **partial density operator**: positive operator $\rho$, $tr(\rho) \leq 1$.
- $\mathcal{D}(\mathcal{H})$ — the set of partial density operators in $\mathcal{H}$.
- State Hilbert space of all quantum variables:
  \[ \mathcal{H}_{all} = \bigotimes_{q \in \text{Var}} \mathcal{H}_q. \]
- $E$ — empty program; i.e. termination.
- **Configuration**: pair $\langle S, \rho \rangle$, where:
Notations

▶ partial density operator: positive operator $\rho$, $tr(\rho) \leq 1$.
▶ $\mathcal{D}(\mathcal{H})$ — the set of partial density operators in $\mathcal{H}$.
▶ State Hilbert space of all quantum variables:

$$\mathcal{H}_{all} = \bigotimes_{q \in \text{Var}} \mathcal{H}_q.$$  

▶ $E$ — empty program; i.e. termination.
▶ Configuration: pair $\langle S, \rho \rangle$, where:
  1. $S$ is a quantum program or the empty program $E$;
Notations

- partial density operator: positive operator $\rho$, $\text{tr}(\rho) \leq 1$.
- $\mathcal{D}(\mathcal{H})$ — the set of partial density operators in $\mathcal{H}$.
- State Hilbert space of all quantum variables:
  \[ \mathcal{H}_{all} = \bigotimes_{q \in \text{Var}} \mathcal{H}_q. \]
- $E$ — empty program; i.e. termination.
- Configuration: pair $\langle S, \rho \rangle$, where:
  1. $S$ is a quantum program or the empty program $E$;
  2. $\rho \in \mathcal{D}(\mathcal{H}_{all})$, denoting the (global) state of quantum variables.
Notations

- **partial density operator**: positive operator $\rho$, $tr(\rho) \leq 1$.
- $\mathcal{D}(\mathcal{H})$ — the set of partial density operators in $\mathcal{H}$.
- State Hilbert space of all quantum variables:

$$\mathcal{H}_{\text{all}} = \bigotimes_{q \in \text{Var}} \mathcal{H}_q.$$  

- $E$ — empty program; i.e. termination.
- **Configuration**: pair $\langle S, \rho \rangle$, where:
  1. $S$ is a quantum program or the empty program $E$;
  2. $\rho \in \mathcal{D}(\mathcal{H}_{\text{all}})$, denoting the (global) state of quantum variables.
- **Transition**:

$$\langle S, \rho \rangle \rightarrow \langle S', \rho' \rangle$$
Operational Semantics

**(SK)** \[ \langle \text{skip}, \rho \rangle \rightarrow \langle E, \rho \rangle \]

**(IN)** \[ \langle \tilde{q} := |0\rangle, \rho \rangle \rightarrow \langle E, \rho_q^0 \rangle \]

where: \( \rho_q^0 = \sum_i |0\rangle_q \langle i| \rho \langle i \rangle_q |0\rangle \)

**(UT)** \[ \langle \tilde{q} := U[\tilde{q}], \rho \rangle \rightarrow \langle E, U\rho U^\dagger \rangle \]

**(SC)** \[ \langle S_1, \rho \rangle \rightarrow \langle S'_1, \rho' \rangle \]
\[ \langle S_1; S_2, \rho \rangle \rightarrow \langle S'_1; S_2, \rho' \rangle \]

where: \( E; S_2 = S_2 \).
Operational Semantics

\[(\text{IF})\]
\[
\langle \text{if } (\Box m \cdot M[q] = m \rightarrow S_m) \text{ fi}, \rho \rangle \rightarrow \langle S_m, M_m \rho M_m^\dagger \rangle
\]

for each possible outcome \(m\) of measurement \(M = \{M_m\}\).

\[(L0)\]
\[
\langle \text{while } M[q] = 1 \text{ do } S \text{ od}, \rho \rangle \rightarrow \langle E, M_0 \rho M_0^\dagger \rangle
\]

\[(L1)\]
\[
\langle \text{while } M[q] = 1 \text{ do } S \text{ od}, \rho \rangle \rightarrow \langle S; \text{while } M[q] = 1 \text{ do } S \text{ od}, M_1 \rho M_1^\dagger \rangle
\]
Computation of Programs

1. A (finite or infinite) transition sequence of program $S$ with input $\rho \in D(\mathcal{H}_{all})$:

$$\langle S, \rho \rangle \rightarrow \langle S_1, \rho_1 \rangle \rightarrow ... \rightarrow \langle S_n, \rho_n \rangle \rightarrow \langle S_{n+1}, \rho_{n+1} \rangle \rightarrow ...$$

such that $\rho_n \neq 0$ for all $n$ (except the last $n$ in the case of a finite sequence).
Computation of Programs

1. A (finite or infinite) transition sequence of program $S$ with input $\rho \in D(\mathcal{H}_{all})$:

   $$\langle S, \rho \rangle \rightarrow \langle S_1, \rho_1 \rangle \rightarrow \ldots \rightarrow \langle S_n, \rho_n \rangle \rightarrow \langle S_{n+1}, \rho_{n+1} \rangle \rightarrow \ldots$$

   such that $\rho_n \neq 0$ for all $n$ (except the last $n$ in the case of a finite sequence).

2. If this sequence cannot be extended, it is called a computation.
Computation of Programs

1. A (finite or infinite) transition sequence of program $S$ with input $\rho \in D(\mathcal{H}_{all})$:

$$\langle S, \rho \rangle \rightarrow \langle S_1, \rho_1 \rangle \rightarrow ... \rightarrow \langle S_n, \rho_n \rangle \rightarrow \langle S_{n+1}, \rho_{n+1} \rangle \rightarrow ...$$

such that $\rho_n \neq 0$ for all $n$ (except the last $n$ in the case of a finite sequence).

2. If this sequence cannot be extended, it is called a computation.
   - If it is finite and its last configuration is $\langle E, \rho' \rangle$, we say it terminates in $\rho'$. 

▶ If it is infinite, we say it diverges.
Computation of Programs

1. A (finite or infinite) transition sequence of program $S$ with input $\rho \in D(H_{all})$:

$$\langle S, \rho \rangle \rightarrow \langle S_1, \rho_1 \rangle \rightarrow ... \rightarrow \langle S_n, \rho_n \rangle \rightarrow \langle S_{n+1}, \rho_{n+1} \rangle \rightarrow ...$$

such that $\rho_n \neq 0$ for all $n$ (except the last $n$ in the case of a finite sequence).

2. If this sequence cannot be extended, it is called a computation.
   - If it is finite and its last configuration is $\langle E, \rho' \rangle$, we say it terminates in $\rho'$.
   - If it is infinite, we say it diverges.
Notation

- Write:

\[
\langle S, \rho \rangle \rightarrow^n \langle S', \rho' \rangle
\]

if there are configurations \(\langle S_1, \rho_1 \rangle, \ldots, \langle S_{n-1}, \rho_{n-1} \rangle\) such that

\[
\langle S, \rho \rangle \rightarrow \langle S_1, \rho_1 \rangle \rightarrow \ldots \rightarrow \langle S_{n-1}, \rho_{n-1} \rangle \rightarrow \langle S', \rho' \rangle,
\]
Notation

- Write:
  \[ \langle S, \rho \rangle \rightarrow^n \langle S', \rho' \rangle \]
  if there are configurations \( \langle S_1, \rho_1 \rangle, \ldots, \langle S_{n-1}, \rho_{n-1} \rangle \) such that
  \[ \langle S, \rho \rangle \rightarrow \langle S_1, \rho_1 \rangle \rightarrow \ldots \rightarrow \langle S_{n-1}, \rho_{n-1} \rangle \rightarrow \langle S', \rho' \rangle, \]

- Write \( \rightarrow^* \) for the reflexive and transitive closures of \( \rightarrow \):
  \[ \langle S, \rho \rangle \rightarrow^* \langle S', \rho' \rangle \]
  if and only if \( \langle S, \rho \rangle \rightarrow^n \langle S', \rho' \rangle \) for some \( n \geq 0 \).
Outline

Introduction

Syntax of Quantum Programs

Operational Semantics

Denotational Semantics

(Floyd-)Hoare Logic for Quantum Programs

Research Problems
Semantic Function

Semantic function of program $S$:

$$\llbracket S \rrbracket : \mathcal{D} (\mathcal{H}_{all}) \rightarrow \mathcal{D} (\mathcal{H}_{all})$$

$$\llbracket S \rrbracket (\rho) = \sum \{|\rho' : \langle S, \rho \rangle \rightarrow^* \langle E, \rho' \rangle|\}$$

Exercise 2
Try to compute semantic function of your quantum walk program.
Semantic Function

Semantic function of program $S$:

$$\llbracket S \rrbracket : \mathcal{D}(\mathcal{H}_{all}) \to \mathcal{D}(\mathcal{H}_{all})$$

$$\llbracket S \rrbracket (\rho) = \sum \{ |\rho' : \langle S, \rho \rangle \rightarrow^* \langle E, \rho' \rangle | \}$$

Exercise 2

Try to compute semantic function of your quantum walk program.
Structural Representation

1. $[[\text{skip}]](\rho) = \rho.$
Structural Representation

1. $\llbracket \text{skip} \rrbracket(\rho) = \rho$.
2. $\llbracket q := \langle 0 \rangle \rrbracket(\rho) = \sum_i \langle 0 \rangle_q \langle i \rangle_{\rho} \langle i \rangle_q \langle 0 \rangle$. 
Structural Representation

1. $\llbracket \text{skip} \rrbracket(\rho) = \rho$.
2. $\llbracket q := |0\rangle \rrbracket(\rho) = \sum_i |0\rangle_q \langle i | \rho | i \rangle_q \langle 0 |$.
3. $\llbracket \bar{q} := U[\bar{q}] \rrbracket(\rho) = U \rho U^\dagger$. 
Structural Representation

1. $\langle \text{skip} \rangle(\rho) = \rho$.
2. $\langle q := |0\rangle \rangle(\rho) = \sum_i |0\rangle_q \langle i| \rho |i\rangle_q \langle 0|$.
3. $\langle \bar{q} := U[\bar{q}] \rangle(\rho) = U\rho U^\dagger$.
4. $\langle S_1; S_2 \rangle(\rho) = \langle S_2 \rangle(\langle S_1 \rangle(\rho))$. 
Structural Representation

1. \([\text{skip}]\)(\(\rho\)) = \(\rho\).

2. \(\langle q := |0\rangle \rangle(\rho) = \sum_i |0\rangle_q \langle i| \rho \langle i|_q \langle 0|\).

3. \(\langle \overline{q} := U[\overline{q}] \rangle(\rho) = U\rho U^\dagger\).

4. \(\langle S_1; S_2 \rangle(\rho) = \langle S_2 \rangle(\langle S_1 \rangle(\rho))\).

5. \(\langle \text{if } (\Box m \cdot M[\overline{q}] = m \rightarrow S_m) \text{ fi} \rangle(\rho) = \sum_m \langle S_m \rangle(M_m \rho M_m^\dagger)\).
Structural Representation

1. $[[\text{skip}])(\rho) = \rho.$
2. $[[q := |0\rangle](\rho) = \sum_i |0\rangle_q \langle i| \rho |i\rangle_q \langle 0|.$
3. $[[\overline{q} := U[\overline{q}])(\rho) = UpU^\dagger.$
4. $[[S_1; S_2]](\rho) = [[S_2]]([[S_1]](\rho)).$
5. $[[\text{if } (\Box m \cdot M[\overline{q}] = m \rightarrow S_m) \text{ fi}}(\rho) = \sum_m [[S_m]](M_m \rho M_m^\dagger).$
6. $[[\text{while } M[\overline{q}] = 1 \text{ do S od}}](\rho) = ???$
Basic Lattice Theory

- A partial order $(L, \sqsubseteq)$: $L$ is a nonempty set, $\sqsubseteq$ is a binary relation on $L$ satisfying:
  1. Reflexivity: $x \sqsubseteq x$ for all $x \in L$;
  2. Antisymmetry: $x \sqsubseteq y$ and $y \sqsubseteq x$ imply $x = y$ for all $x, y \in L$;
  3. Transitivity: $x \sqsubseteq y$ and $y \sqsubseteq z$ imply $x \sqsubseteq z$ for all $x, y, z \in L$.

$x \in L$ is called the least element when $x \sqsubseteq y$ for all $y \in L$.

$x \in L$ is called an upper bound of a subset $X \subseteq L$ if $y \sqsubseteq x$ for all $x \in X$.

$x$ is called the least upper bound of $X$, written $x = \bigvee X$, if $x$ is an upper bound of $X$;
for any upper bound $y$ of $X$, $x \sqsubseteq y$.
Basic Lattice Theory

- A partial order \((L, \sqsubseteq)\): \(L\) is a nonempty set, \(\sqsubseteq\) is a binary relation on \(L\) satisfying:
  1. Reflexivity: \(x \sqsubseteq x\) for all \(x \in L\);
Basic Lattice Theory

- A partial order \((L, \sqsubseteq)\): \(L\) is a nonempty set, \(\sqsubseteq\) is a binary relation on \(L\) satisfying:
  1. Reflexivity: \(x \sqsubseteq x\) for all \(x \in L\);
  2. Antisymmetry: \(x \sqsubseteq y\) and \(y \sqsubseteq x\) imply \(x = y\) for all \(x, y \in L\);
Basic Lattice Theory

- A partial order \((L, \sqsubseteq)\): \(L\) is a nonempty set, \(\sqsubseteq\) is a binary relation on \(L\) satisfying:
  1. Reflexivity: \(x \sqsubseteq x\) for all \(x \in L\);
  2. Antisymmetry: \(x \sqsubseteq y\) and \(y \sqsubseteq x\) imply \(x = y\) for all \(x, y \in L\);
  3. Transitivity: \(x \sqsubseteq y\) and \(y \sqsubseteq z\) imply \(x \sqsubseteq z\) for all \(x, y, z \in L\).
Basic Lattice Theory

- A **partial order** \((L, \sqsubseteq)\): \(L\) is a nonempty set, \(\sqsubseteq\) is a binary relation on \(L\) satisfying:
  1. Reflexivity: \(x \sqsubseteq x\) for all \(x \in L\);
  2. Antisymmetry: \(x \sqsubseteq y\) and \(y \sqsubseteq x\) imply \(x = y\) for all \(x, y \in L\);
  3. Transitivity: \(x \sqsubseteq y\) and \(y \sqsubseteq z\) imply \(x \sqsubseteq z\) for all \(x, y, z \in L\).

- \(x \in L\) is called the **least element** when \(x \sqsubseteq y\) for all \(y \in L\).
Basic Lattice Theory

- A partial order \((L, \sqsubseteq)\): \(L\) is a nonempty set, \(\sqsubseteq\) is a binary relation on \(L\) satisfying:
  1. Reflexivity: \(x \sqsubseteq x\) for all \(x \in L\);
  2. Antisymmetry: \(x \sqsubseteq y\) and \(y \sqsubseteq x\) imply \(x = y\) for all \(x, y \in L\);
  3. Transitivity: \(x \sqsubseteq y\) and \(y \sqsubseteq z\) imply \(x \sqsubseteq z\) for all \(x, y, z \in L\).

- \(x \in L\) is called the least element when \(x \sqsubseteq y\) for all \(y \in L\).

- \(x \in L\) is called an upper bound of a subset \(X \subseteq L\) if \(y \sqsubseteq x\) for all \(x \in X\).
Basic Lattice Theory

- A partial order \((L, \sqsubseteq)\): \(L\) is a nonempty set, \(\sqsubseteq\) is a binary relation on \(L\) satisfying:
  1. Reflexivity: \(x \sqsubseteq x\) for all \(x \in L\);
  2. Antisymmetry: \(x \sqsubseteq y\) and \(y \sqsubseteq x\) imply \(x = y\) for all \(x, y \in L\);
  3. Transitivity: \(x \sqsubseteq y\) and \(y \sqsubseteq z\) imply \(x \sqsubseteq z\) for all \(x, y, z \in L\).

- \(x \in L\) is called the **least element** when \(x \sqsubseteq y\) for all \(y \in L\).
- \(x \in L\) is called an upper bound of a subset \(X \subseteq L\) if \(y \sqsubseteq x\) for all \(x \in X\).
- \(x\) is called the least upper bound of \(X\), written \(x = \bigcup X\), if
Basic Lattice Theory

- A partial order \((L, \sqsubseteq)\): \(L\) is a nonempty set, \(\sqsubseteq\) is a binary relation on \(L\) satisfying:
  1. Reflexivity: \(x \sqsubseteq x\) for all \(x \in L\);
  2. Antisymmetry: \(x \sqsubseteq y\) and \(y \sqsubseteq x\) imply \(x = y\) for all \(x, y \in L\);
  3. Transitivity: \(x \sqsubseteq y\) and \(y \sqsubseteq z\) imply \(x \sqsubseteq z\) for all \(x, y, z \in L\).

- \(x \in L\) is called the least element when \(x \sqsubseteq y\) for all \(y \in L\).

- \(x \in L\) is called an upper bound of a subset \(X \subseteq L\) if \(y \sqsubseteq x\) for all \(x \in X\).

- \(x\) is called the least upper bound of \(X\), written \(x = \bigvee X\), if
  - \(x\) is an upper bound of \(X\);
Basic Lattice Theory

- A partial order \((L, \sqsubseteq)\): \(L\) is a nonempty set, \(\sqsubseteq\) is a binary relation on \(L\) satisfying:
  1. Reflexivity: \(x \sqsubseteq x\) for all \(x \in L\);
  2. Antisymmetry: \(x \sqsubseteq y\) and \(y \sqsubseteq x\) imply \(x = y\) for all \(x, y \in L\);
  3. Transitivity: \(x \sqsubseteq y\) and \(y \sqsubseteq z\) imply \(x \sqsubseteq z\) for all \(x, y, z \in L\).

- \(x \in L\) is called the least element when \(x \sqsubseteq y\) for all \(y \in L\).

- \(x \in L\) is called an upper bound of a subset \(X \subseteq L\) if \(y \sqsubseteq x\) for all \(x \in X\).

- \(x\) is called the least upper bound of \(X\), written \(x = \bigvee X\), if
  - \(x\) is an upper bound of \(X\);
  - for any upper bound \(y\) of \(X\), \(x \sqsubseteq y\).
Basic Lattice Theory

- A complete partial order (CPO) is a partial order \((L, \sqsubseteq)\):
  
  1. It has the least element 0;
  2. \(\sqcup\) exists for any increasing sequence \(\{x_n\}\):
     
     \[ x_0 \sqsubseteq \ldots \sqsubseteq x_n \sqsubseteq \ldots \]

A function \(f\) from \(L\) into itself is continuous if

\[ f(\sqcup_n x_n) = \sqcup_n f(x_n) \]

for any increasing sequence \(\{x_n\}\) in \(L\).
Basic Lattice Theory

- A complete partial order (CPO) is a partial order \((L, \sqsubseteq)\):
  1. it has the least element \(0\);
Basic Lattice Theory

- A complete partial order (CPO) is a partial order \((L, \sqsubseteq)\):
  1. it has the least element 0;
  2. \(\bigsqcup_{n=0}^{\infty} x_n\) exists for any increasing sequence \(\{x_n\}\):

\[ x_0 \sqsubseteq \ldots \sqsubseteq x_n \sqsubseteq x_{n+1} \sqsubseteq \ldots \]
Basic Lattice Theory

- A complete partial order (CPO) is a partial order \((L, \sqsubseteq)\):
  1. it has the least element 0;
  2. \(\bigsqcup_{n=0}^{\infty} x_n\) exists for any increasing sequence \(\{x_n\}\):

\[
x_0 \sqsubseteq ... \sqsubseteq x_n \sqsubseteq x_{n+1} \sqsubseteq ....
\]

- A function \(f\) from \(L\) into itself is continuous if

\[
f \left( \bigsqcup_n x_n \right) = \bigsqcup_n f(x_n)
\]

for any increasing sequence \(\{x_n\}\) in \(L\).
Knaster-Tarski Theorem

Let \((L, \sqsubseteq)\) be a CPO and function \(f : L \to L\) continuous. Then \(f\) has the least fixed point

\[
\mu f = \bigsqcup_{n=0}^{\infty} f^{(n)}(0)
\]

where

\[
\begin{align*}
  f^{(0)}(0) &= 0, \\
  f^{(n+1)}(0) &= f(f^{(n)}(0)) \text{ for } n \geq 0.
\end{align*}
\]
CPO of Partial Density Operators

- **Löwner order**: operators $A \sqsubseteq B \iff B - A$ is positive.
CPO of Partial Density Operators

- Löwner order: operators $A \sqsubseteq B \iff B - A$ is positive.
- $(\mathcal{D}(\mathcal{H}), \sqsubseteq)$ is a CPO with the zero operator $0_{\mathcal{H}}$ as its least element.

Exercise 3

Prove the above statement for finite-dimensional $\mathcal{H}$. 
CPO of Super-operators

- Each super-operator in $\mathcal{H}$ is a continuous function from $(\mathcal{D}(\mathcal{H}), \sqsubseteq)$ into itself.
CPO of Super-operators

- Each super-operator in $\mathcal{H}$ is a continuous function from $(\mathcal{D}(\mathcal{H}), \sqsubseteq)$ into itself.
- $\mathcal{QO}(\mathcal{H})$ — the set of superoperators in Hilbert space $\mathcal{H}$. 

Löwner order between operators can be lifted to a partial order between super-operators: 

$E \sqsubseteq F \iff E(\rho) \sqsubseteq F(\rho)$ for all $\rho \in \mathcal{D}(\mathcal{H})$. 

$(\mathcal{QO}(\mathcal{H}), \sqsubseteq)$ is a CPO.
CPO of Super-operators

- Each super-operator in $\mathcal{H}$ is a continuous function from $(\mathcal{D}(\mathcal{H}), \sqsubseteq)$ into itself.
- $QO(\mathcal{H})$ — the set of superoperators in Hilbert space $\mathcal{H}$.
- Löwner order between operators can be lifted to a partial order between super-operators:

$$\mathcal{E} \sqsubseteq \mathcal{F} \iff \mathcal{E}(\rho) \sqsubseteq \mathcal{F}(\rho) \text{ for all } \rho \in \mathcal{D}(\mathcal{H})$$
CPO of Super-operators

- Each super-operator in $\mathcal{H}$ is a continuous function from $(\mathcal{D}(\mathcal{H}), \sqsubseteq)$ into itself.
- $\mathcal{QO}(\mathcal{H})$ — the set of superoperators in Hilbert space $\mathcal{H}$.
- Löwner order between operators can be lifted to a partial order between super-operators:

$$\mathcal{E} \sqsubseteq \mathcal{F} \iff \mathcal{E}(\rho) \sqsubseteq \mathcal{F}(\rho) \text{ for all } \rho \in \mathcal{D}(\mathcal{H})$$

- $(\mathcal{QO}(\mathcal{H}), \sqsubseteq)$ is a CPO.
Syntactic Approximation

- **abort** denotes a program such that

\[
⟦\text{abort}⟧(ρ) = 0_H \text{ all for all } ρ \in D(ℋ).
\]
Syntactic Approximation

- **abort** denotes a program such that

\[
\llbracket \text{abort} \rrbracket (\rho) = 0_{H_{\text{all}}} \text{ for all } \rho \in D(H).
\]

- Write:

\[
\text{while } \equiv \text{while } M[\bar{q}] = 1 \text{ do } S \text{ od}.
\]
Syntactic Approximation

▷ abort denotes a program such that

\[
\llbracket \text{abort} \rrbracket (\rho) = 0_{\mathcal{H}_{\text{all}}} \text{ for all } \rho \in \mathcal{D}(\mathcal{H}).
\]

▷ Write:

\[
\text{while } \equiv \text{while } M[\bar{q}] = 1 \text{ do } S \text{ od.}
\]

▷ For integer \( k \geq 0 \), the \( k \)th syntactic approximation \( \text{while}^{(k)} \) of while:

\[
\begin{cases}
\text{while}^{(0)} & \equiv \text{abort}, \\
\text{while}^{(k+1)} & \equiv \text{if } M[\bar{q}] = 0 \rightarrow \text{skip} \\
& \quad \square 1 \rightarrow S; \text{while}^{(k)} \\
& \quad \text{fi}
\end{cases}
\]
Semantic Function of Loops

$$\llbracket \text{while} \rrbracket = \bigcup_{k=0}^{\infty} \llbracket \text{while}^{(k)} \rrbracket,$$

where $\sqcup$ stands for the least upper bound in CPO $(QO (\mathcal{H}_{all}), \sqsubseteq)$. 

Exercise 4
Prove the above equality.
Semantic Function of Loops

$$\llbracket \text{while} \rrbracket = \bigsqcup_{k=0}^{\infty} \llbracket \text{while}^{(k)} \rrbracket,$$

where $\bigsqcup$ stands for the least upper bound in CPO $(QO(H_{all}), \sqsubseteq)$.

Fixed Point Characterisation

For any $\rho \in D(H_{all})$:

$$\llbracket \text{while} \rrbracket(\rho) = M_0\rho M_0^+ + \llbracket \text{while} \rrbracket \left( \llbracket S \rrbracket \left( M_1\rho M_1^+ \right) \right).$$
Semantic Function of Loops

\[ \llbracket \text{while} \rrbracket = \bigsqcup_{k=0}^{\infty} \llbracket \text{while}^{(k)} \rrbracket , \]

where \( \bigsqcup \) stands for the least upper bound in CPO \((\mathcal{QO}(\mathcal{H}_{all}), \sqsubseteq)\).

Fixed Point Characterisation

For any \( \rho \in \mathcal{D}(\mathcal{H}_{all}) \):

\[ \llbracket \text{while} \rrbracket (\rho) = M_0 \rho M_0^+ + \llbracket \text{while} \rrbracket \left( \llbracket S \rrbracket \left( M_1 \rho M_1^+ \right) \right) . \]

Exercise 4

Prove the above equality.
Termination and Divergence Probabilities

- For any quantum program $S$ and $\rho \in D(\mathcal{H}_{all})$:

$$tr([S](\rho)) \leq tr(\rho).$$
Termination and Divergence Probabilities

- For any quantum program $S$ and $\rho \in \mathcal{D}(\mathcal{H}_{all})$:
  \[
  \text{tr}(\llbracket S \rrbracket(\rho)) \leq \text{tr}(\rho).
  \]

- $\text{tr}(\llbracket S \rrbracket(\rho))$ is the probability that program $S$ with input $\rho$ terminates.
Outline

Introduction

Syntax of Quantum Programs

Operational Semantics

Denotational Semantics

(Floyd-)Hoare Logic for Quantum Programs

Research Problems
Quantum predicates = Quantum effects

- A **quantum effect** is a (Hermitian) operator $0 \subseteq M \subseteq I$; that is, $0 \leq tr(M\rho) \leq 1$ for all density operators.
Quantum predicates = Quantum effects

- A **quantum effect** is a (Hermitian) operator $0 \subseteq M \subseteq I$; that is, $0 \leq \text{tr}(M\rho) \leq 1$ for all density operators.
- $\text{tr}(M\rho)$ may be interpreted as the expected degree to which quantum state $\rho$ satisfies quantum predicate $M$. 
Correctness Formulas

- A correctness formula (Hoare triple) is a statement of the form:

\[{P} \triangleright S \triangleright {Q}\]

where:

- $S$ is a quantum program;
- $P$, $Q$ are quantum predicates in $\mathcal{H}$.
- $P$ is called the precondition, $Q$ the postcondition.

Partial Correctness, Total Correctness

- Partial correctness: If an input to program $S$ satisfies precondition $P$, then either $S$ does not terminate, or it terminates in a state satisfying postcondition $Q$.
- Total correctness: If an input to program $S$ satisfies precondition $P$, then $S$ must terminate and it terminates in a state satisfying postcondition $Q$. 
Correctness Formulas

- A correctness formula (Hoare triple) is a statement of the form:

\[ \{P\} S \{Q\} \]

where:
- S is a quantum program;

Partial Correctness, Total Correctness
Correctness Formulas

- A correctness formula (Hoare triple) is a statement of the form:

\[ \{ P \} S \{ Q \} \]

where:
- \( S \) is a quantum program;
- \( P, Q \) are quantum predicates in \( \mathcal{H}_{all} \).

Partial Correctness, Total Correctness
Correctness Formulas

- A correctness formula (Hoare triple) is a statement of the form:

\[ \{ P \} S \{ Q \} \]

where:
- \( S \) is a quantum program;
- \( P, Q \) are quantum predicates in \( H_{all} \);
- \( P \) is called the precondition, \( Q \) the postcondition.

Partial Correctness, Total Correctness
Correctness Formulas

- A correctness formula (Hoare triple) is a statement of the form:

\[ \{ P \} S \{ Q \} \]

where:
- \( S \) is a quantum program;
- \( P, Q \) are quantum predicates in \( \mathcal{H}_{all} \);
- \( P \) is called the precondition, \( Q \) the postcondition.

Partial Correctness, Total Correctness

- **Partial correctness**: If an input to program \( S \) satisfies precondition \( P \), then either \( S \) does not terminate, or it terminates in a state satisfying postcondition \( Q \).
Correctness Formulas

- A correctness formula (Hoare triple) is a statement of the form:

\[ \{P\} S \{Q\} \]

where:
- \( S \) is a quantum program;
- \( P, Q \) are quantum predicates in \( \mathcal{H}_{all} \).
- \( P \) is called the precondition, \( Q \) the postcondition.

Partial Correctness, Total Correctness

- **Partial correctness**: If an input to program \( S \) satisfies precondition \( P \), then either \( S \) does not terminate, or it terminates in a state satisfying postcondition \( Q \).
- **Total correctness**: If an input to program \( S \) satisfies precondition \( P \), then \( S \) must terminate and it terminates in a state satisfying postcondition \( Q \).
Partial Correctness, Total Correctness (Continued)

- The correctness formula \( \{ P \} S \{ Q \} \) is true in the sense of total correctness, written

\[
\models_{tot} \{ P \} S \{ Q \},
\]

if:

\[
tr(P\rho) \leq tr(Q[\llbracket S \rrbracket](\rho))
\]

for all \( \rho \in D(\mathcal{H}_{all}) \), where \( \llbracket S \rrbracket \) is the semantic function of \( S \).
Partial Correctness, Total Correctness (Continued)

- The correctness formula \(\{P\} S \{Q\}\) is true in the sense of *total correctness*, written
  
  \[\models_{\text{tot}} \{P\} S \{Q\},\]

  if:
  
  \[tr(P\rho) \leq tr(Q[S](\rho))\]

  for all \(\rho \in D(\mathcal{H}_{all})\), where \([S]\) is the semantic function of \(S\).

- The correctness formula \(\{P\} S \{Q\}\) is true in the sense of *partial correctness*, written
  
  \[\models_{\text{par}} \{P\} S \{Q\},\]

  if:
  
  \[tr(P\rho) \leq tr(Q[S](\rho)) + [tr(\rho) - tr([S](\rho))]\]

  for all \(\rho \in D(\mathcal{H}_{all})\).
Proof System for Partial Correctness

\[(Ax - Sk) \quad \{P\} \textbf{Skip}\{P\}\]

\[(Ax - In) \text{ If } type(q) = \textbf{Boolean}, \text{ then } \]

\[
\{|0\rangle_q\langle 0|P|0\rangle_q\langle 0| + |1\rangle_q\langle 0|P|0\rangle_q\langle 1|\}q := |0\rangle\{P\}
\]

If type(q) = \textbf{integer}, then

\[
\left\{ \sum_{n=-\infty}^{\infty} |n\rangle_q\langle 0|P|0\rangle_q\langle n| \right\}q := |0\rangle\{P\}
\]

\[(Ax - UT) \quad \{U^\dagger PU\}q := Uq\{P\}\]
Proof System for Partial Correctness (Continued)

(R – SC) \[
\frac{\{P\} S_1 \{Q\} \quad \{Q\} S_2 \{R\}}{\{P\} S_1 ; S_2 \{R\}}
\]

(R – IF) \[
\frac{\{P_m\} S_m \{Q\} \text{ for all } m}{\{\sum_m M^\dagger_m P_m M_m\} \text{ if } (\Box m \cdot M[\overline{q}] = m \rightarrow S_m) \text{ fi} \{Q\}}
\]

(R – LP) \[
\frac{\{Q\} S \{M_0^\dagger P M_0 + M_1^\dagger Q M_1\}}{\{M_0^\dagger P M_0 + M_1^\dagger Q M_1\} \text{ while } M[\overline{q}] = 1 \text{ do } S \text{ od} \{P\}}
\]

(R – Or) \[
\frac{P \sqsubseteq P' \quad \{P'\} S\{Q'\} \quad Q' \sqsubseteq Q}{\{P\} S\{Q\}}
\]
Soundness Theorem

For any quantum `while`-program $S$ and quantum predicates $P, Q \in \mathcal{P}(\mathcal{H}_{all})$:

$$\vdash_{qPD} \{P\}S\{Q\} \text{ implies } \models_{par} \{P\}S\{Q\}.$$
Soundness Theorem
For any quantum while-program $S$ and quantum predicates $P, Q \in \mathcal{P}(\mathcal{H}_{all})$:

$$\vdash_{qPD} \{P\}S\{Q\} \text{ implies } \models_{par} \{P\}S\{Q\}.$$ 

Exercise 5
Prove soundness theorem.
**Soundness Theorem**
For any quantum *while*-program $S$ and quantum predicates $P, Q \in \mathcal{P}(\mathcal{H}_{all})$:

$$\vdash_{qPD} \{P\}S\{Q\} \text{ implies } \models_{par} \{P\}S\{Q\}.$$

**Exercise 5**
Prove soundness theorem.

**(Relative) Completeness Theorem**
For any quantum *while*-program $S$ and quantum predicates $P, Q \in \mathcal{P}(\mathcal{H}_{all})$:

$$\models_{par} \{P\}S\{Q\} \text{ implies } \vdash_{qPD} \{P\}S\{Q\}.$$
Bound (Ranking) Functions

- Let $P \in \mathcal{P}(\mathcal{H}_{all})$ be a quantum predicate, real number $\epsilon > 0$. 
Bound (Ranking) Functions

- Let $P \in \mathcal{P}(\mathcal{H}_{all})$ be a quantum predicate, real number $\epsilon > 0$.
- A function
  
  $t : \mathcal{D}(\mathcal{H}_{all}) \rightarrow \omega$

  is a $(P, \epsilon)$-bound function of quantum loop

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

  if for all $\rho \in \mathcal{D}(\mathcal{H}_{all})$: 


Bound (Ranking) Functions

- Let $P \in \mathcal{P}(\mathcal{H}_{all})$ be a quantum predicate, real number $\epsilon > 0$.
- A function
  
  $$t : \mathcal{D}(\mathcal{H}_{all}) \to \omega$$

  is a $(P, \epsilon)$-bound function of quantum loop

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

  if for all $\rho \in \mathcal{D}(\mathcal{H}_{all})$:
  1. $t(\llbracket S \rrbracket (M_1 \rho M_1^\dagger)) \leq t(\rho)$;
Bound (Ranking) Functions

- Let $P \in \mathcal{P}(\mathcal{H}_{all})$ be a quantum predicate, real number $\epsilon > 0$.
- A function $t : \mathcal{D}(\mathcal{H}_{all}) \to \omega$

is a $(P, \epsilon)$-bound function of quantum loop

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

if for all $\rho \in \mathcal{D}(\mathcal{H}_{all})$:
1. $t \left( \llbracket S \rrbracket \left( M_1 \rho M_1^\dagger \right) \right) \leq t(\rho)$;
2. $\text{tr}(P \rho) \geq \epsilon$ implies

$t \left( \llbracket S \rrbracket \left( M_1 \rho M_1^\dagger \right) \right) < t(\rho)$
Proof System for Total Correctness

- $\{Q\}S\{M_0^\dagger PM_0 + M_1^\dagger QM_1\}$
- for any $\epsilon > 0$, $t_\epsilon$ is a $(M_1^\dagger QM_1, \epsilon)$-bound function of loop while $M[\vec{q}] = 1$ do $S$ od

(R − LT) \[\{M_0^\dagger PM_0 + M_1^\dagger QM_1\} \text{while } M[\vec{q}] = 1 \text{ do } S \text{ od}\{P\}\]
Soundness Theorem
For any quantum program $S$ and quantum predicates $P, Q \in \mathcal{P}(\mathcal{H}_{all})$:

\[
\vdash_{qTD} \{ P \} S \{ Q \} \text{ implies } \models_{tot} \{ P \} S \{ Q \}.
\]
Soundness Theorem
For any quantum program $S$ and quantum predicates $P, Q \in \mathcal{P}(\mathcal{H}_{all})$:

$$\vdash_{qTD} \{P\} S \{Q\} \text{ implies } \models_{tot} \{P\} S \{Q\}.$$  

(Relative) Completeness Theorem
For any quantum program $S$ and quantum predicates $P, Q \in \mathcal{P}(\mathcal{H}_{all})$:

$$\models_{tot} \{P\} S \{Q\} \text{ implies } \vdash_{qTD} \{P\} S \{Q\}.$$
Outline

Introduction

Syntax of Quantum Programs

Operational Semantics

Denotational Semantics

(Floyd-)Hoare Logic for Quantum Programs

Research Problems
Research problems

1. Compute the expected running time of the quantum walk.
Research problems

1. Compute the expected running time of the quantum walk.
2. Develop a logic for recursive quantum programs.

Further reading

Research problems

1. Compute the expected running time of the quantum walk.
2. Develop a logic for recursive quantum programs.
3. Parallel or distributed quantum programs?

Further reading

Research problems

1. Compute the expected running time of the quantum walk.
2. Develop a logic for recursive quantum programs.
3. Parallel or distributed quantum programs?
4. Improve the invariant generation and termination analysis algorithms for quantum programs.

Further reading

Thank You!