Foundations of Quantum Programming
(Lecture 2)

Mingsheng Ying

Centre for Quantum Software and Information
University of Technology Sydney

Outline

Outline

Quantum programming languages/tools

- IBM: Qiskit

Quantum programming languages/tools

- IBM: Qiskit
- Microsoft: Q#

Quantum programming languages/tools

- IBM: Qiskit
- Microsoft: Q#
- Google: Cirq

Quantum programming languages/tools

- IBM: Qiskit
- Microsoft: Q#
- Google: Cirq
- Quantinuum: Gupy

Quantum programming languages/tools

- IBM: Qiskit
- Microsoft: Q#
- Google: Cirq
- Quantinuum: Gupy
- Europe: Qrisp

Quantum programming languages/tools

- IBM: Qiskit
- Microsoft: Q#
- Google: Cirq
- Quantinuum: Gupy
- Europe: Qrisp
- .....

This lecture

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

This lecture

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

Programming languages research

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

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

- Semantics: operational, denotational, axiomatic

Programming languages research

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

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

- Semantics: operational, denotational, axiomatic
- Turing-complete?

Programming languages research

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

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

- Semantics: operational, denotational, axiomatic
- Turing-complete?
- Compilers

Programming languages research

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

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

- Semantics: operational, denotational, axiomatic
- Turing-complete?
- Compilers
- Program test, debugging, analysis

Programming languages research

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

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

- Semantics: operational, denotational, axiomatic
- Turing-complete?
- Compilers
- Program test, debugging, analysis
- How to verify correctness of your programs?

Programming languages research

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

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

- Semantics: operational, denotational, axiomatic
- Turing-complete?
- Compilers
- Program test, debugging, analysis
- How to verify correctness of your programs?
- Data abstraction

Programming languages research

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

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

- Semantics: operational, denotational, axiomatic
- Turing-complete?
- Compilers
- Program test, debugging, analysis
- How to verify correctness of your programs?
- Data abstraction
- .....

Outline

Combinational Quantum Circuits

- A combinational quantum circuit is a sequence of quantum gates:

$$C \equiv G_1 ... G_m$$

Combinational Quantum Circuits

- A combinational quantum circuit is a sequence of quantum gates:

$$C \equiv G_1 ... G_m$$

Example

Quantum circuit $Z[q_1]H[q_2]C[q_1, q_2]Y[q_1]H[q_2]$:

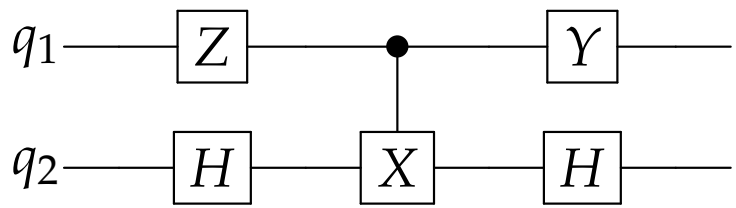

Figure: A combinational quantum circuit.

Dynamic Quantum Circuits

- Quantum measurements are usually used for readout of outcomes at the end of quantum circuits.

Dynamic Quantum Circuits

- Quantum measurements are usually used for readout of outcomes at the end of quantum circuits.
- Measurements can also occur at the middle of a quantum circuit where the outcomes are used to conditionally control subsequent steps of the computation.

Example - Quantum teleportation

Transmit quantum information (e.g. the exact state of a photon) via only classical communication but with the help of entanglement between the sender and receiver.

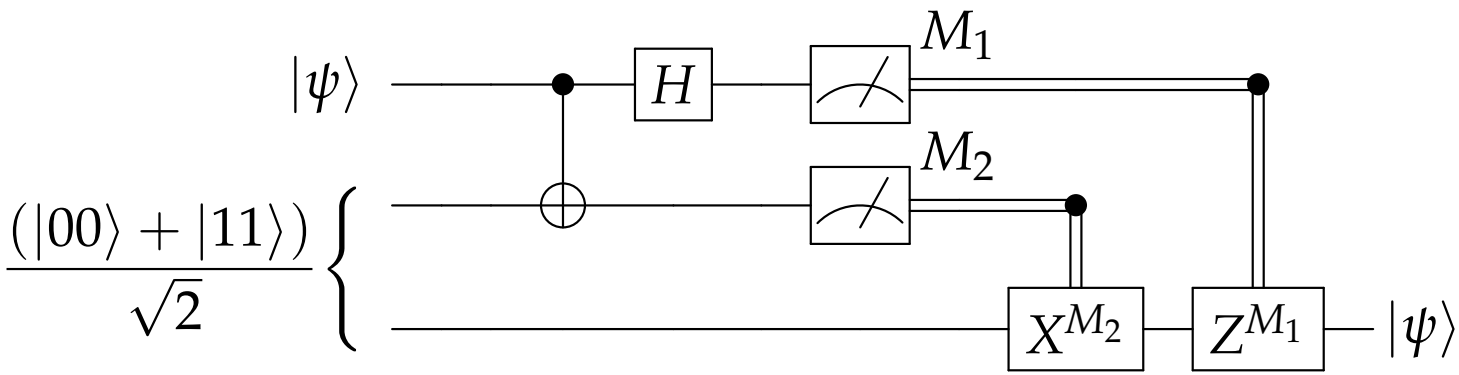

Figure: Quantum teleportation circuit

Quantum Random Walks

- Quantum walk on an n-circle with an absorbing boundary at position 1.

Quantum Random Walks

- Quantum walk on an n-circle with an absorbing boundary at position 1.
- $\mathcal{H}_c$ denotes the coin space, the 2-dimensional Hilbert space with orthonormal basis states $|L\rangle$ and $|R\rangle$, indicating directions.

Quantum Random Walks

- Quantum walk on an n-circle with an absorbing boundary at position 1.
- $\mathcal{H}_c$ denotes the coin space, the 2-dimensional Hilbert space with orthonormal basis states $|L\rangle$ and $|R\rangle$, indicating directions.
- $\mathcal{H}_p$ is the n-dimensional Hilbert space with orthonormal basis states $|0\rangle, |1\rangle, ..., |n-1\rangle$, where $|i\rangle$ denotes position i.

Quantum Random Walks

- Quantum walk on an n-circle with an absorbing boundary at position 1.
- $\mathcal{H}_c$ denotes the coin space, the 2-dimensional Hilbert space with orthonormal basis states $|L\rangle$ and $|R\rangle$, indicating directions.
- $\mathcal{H}_p$ is the n-dimensional Hilbert space with orthonormal basis states $|0\rangle, |1\rangle, ..., |n-1\rangle$, where $|i\rangle$ denotes position i.
- The quantum walk is the composite system of the coin and a walker moving on these positions.

Quantum Random Walks

- Quantum walk on an n-circle with an absorbing boundary at position 1.
- $\mathcal{H}_c$ denotes the coin space, the 2-dimensional Hilbert space with orthonormal basis states $|L\rangle$ and $|R\rangle$, indicating directions.
- $\mathcal{H}_p$ is the n-dimensional Hilbert space with orthonormal basis states $|0\rangle, |1\rangle, ..., |n-1\rangle$, where $|i\rangle$ denotes position i.
- The quantum walk is the composite system of the coin and a walker moving on these positions.
- The state space of the walk is the tensor product $\mathcal{H} = \mathcal{H}_c \otimes \mathcal{H}_p$.

Quantum Random Walks

- The initial state is $|L\rangle|0\rangle$.

Quantum Random Walks

- The initial state is $|L\rangle|0\rangle$.
- Each step of the walk consists of:

Quantum Random Walks

- The initial state is $|L\rangle|0\rangle$.
- Each step of the walk consists of:
  1. Measure the position of the system to see whether it is 1. If the outcome is "yes", then the walk terminates; otherwise, it continues:

$$M_{yes} = |1\rangle\langle 1|,\ M_{no} = I_p - M_{yes} = \sum_{i \neq 1} |i\rangle\langle i|$$

Quantum Random Walks

- The initial state is $|L\rangle|0\rangle$.
- Each step of the walk consists of:
  1. Measure the position of the system to see whether it is 1. If the outcome is "yes", then the walk terminates; otherwise, it continues:

$$M_{yes} = |1\rangle\langle 1|,\ M_{no} = I_p - M_{yes} = \sum_{i \neq 1} |i\rangle\langle i|$$

  2. The Hadamard "coin-tossing" operator

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

  is applied on the coin (or direction) space $\mathcal{H}_c$;

Quantum Random Walks

- The initial state is $|L\rangle|0\rangle$.
- Each step of the walk consists of:
  1. Measure the position of the system to see whether it is 1. If the outcome is "yes", then the walk terminates; otherwise, it continues:

$$M_{yes} = |1\rangle\langle 1|,\ M_{no} = I_p - M_{yes} = \sum_{i\neq 1} |i\rangle\langle i|$$

  2. The Hadamard "coin-tossing" operator

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

     is applied on the coin (or direction) space $\mathcal{H}_c$;
  3. The shift operator S is performed on the space $\mathcal{H}$:

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

     The system walks one step left or right according to the direction state.

Quantum Random Walk

- An essential difference between quantum walk and classical random walk:

Quantum Random Walk

- An essential difference between quantum walk and classical random walk:
  - Coin (or direction) variable c can be in a superposition of $|L\rangle$ and $|R\rangle$ like $|+\rangle = \frac{1}{\sqrt{2}}(|L\rangle + |R\rangle)$;

Quantum Random Walk

- An essential difference between quantum walk and classical random walk:
  - Coin (or direction) variable c can be in a superposition of $|L\rangle$ and $|R\rangle$ like $|+\rangle = \frac{1}{\sqrt{2}}(|L\rangle + |R\rangle)$;
  - Thus, the walker is moving left and right "simultaneously":

$$\frac{1}{\sqrt{2}}(|L\rangle + |R\rangle)|i\rangle \rightarrow \frac{1}{\sqrt{2}}(|L\rangle|i \ominus 1\rangle + |R\rangle|i \oplus 1\rangle).$$

Outline

Syntax

A simple language for imperative quantum programming — **QWhile**:

$$
\begin{aligned}
S ::= \ & \mathbf{skip} \mid q := |0\rangle \\
& \mid S_1; S_2 \\
& \mid U[\overline{q}] \\
& \mid \mathbf{if}\ (\square m \cdot M[\overline{q}] = m \to S_m)\ \mathbf{fi} \\
& \mid \mathbf{while}\ M[\overline{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}
\end{aligned}
$$

Syntax

A simple language for imperative quantum programming — **QWhile**:

$$\begin{aligned} S ::= \mathbf{skip} \mid & q := |0\rangle \\ & | S_1; S_2 \\ & | \ U[\bar{q}] \\ & | \ \mathbf{if} \ (\square m \cdot M[\bar{q}] = m \to S_m) \ \mathbf{fi} \\ & | \ \mathbf{while} \ M[\bar{q}] = 1 \ \mathbf{do} \ S \ \mathbf{od} \end{aligned}$$

Example: Quantum Random Walk

$$\begin{aligned} QW \equiv c := |L\rangle; p := |0\rangle; \\ \mathbf{while} \ M[p] = no \ \mathbf{do} \ H[c]; \ S[c,p] \ \mathbf{od} \end{aligned}$$

Operational Semantics

A *configuration*: $\langle S, \rho \rangle$

- S is a quantum program or $\downarrow$ (termination)

Operational Semantics

A *configuration*: $\langle S, \rho \rangle$

- S is a quantum program or $\downarrow$ (termination)
- ρ is a partial density operator on

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

Operational Semantics

$$(Sk) \qquad \langle \mathbf{skip}, \rho \rangle \to \langle \downarrow, \rho \rangle$$

$$(Ini) \qquad \langle q := |0\rangle, \rho \rangle \to \langle \downarrow, \rho_0^q \rangle \text{ where } \rho_0^q = \sum_n |0\rangle_q \langle n|\rho|n\rangle_q \langle 0|$$

$$(Uni) \qquad \langle U[\bar{q}], \rho \rangle \to \langle \downarrow, U_{\bar{q}} \rho U_{\bar{q}}^\dagger \rangle$$

$$(Seq) \qquad \frac{\langle S_1, \rho \rangle \to \langle S_1', \rho' \rangle}{\langle S_1; S_2, \rho \rangle \to \langle S_1'; S_2, \rho' \rangle} \qquad (\text{Convention} : \downarrow; S_2 = S_2)$$

Operational Semantics

$$(IF)\quad \frac{(\text{for each possible outcome } m)}{\langle \textbf{if } (\square m \cdot M[\overline{q}] = m \to S_m) \textbf{ fi}, \rho\rangle \to \langle S_m, M_m \rho M_m^\dagger\rangle}$$

$$(L0)\quad \langle \textbf{while } M[\overline{q}] = 0 \textbf{ do } S \textbf{ od}, \rho\rangle \to \langle \downarrow, M_0 \rho M_0^\dagger\rangle$$

$$(L1)\quad \langle \textbf{while } M[\overline{q}] = 1 \textbf{ do } S, \rho\rangle \to \langle S; \textbf{while } M[\overline{q}] = 1 \textbf{ do } S, M_1 \rho M_1^\dagger\rangle$$

Denotational Semantics

Semantic function of quantum program S:

$$[\![S]\!] : \mathcal{D}(\mathcal{H}_{\text{all}}) \to \mathcal{D}(\mathcal{H}_{\text{all}})$$

$$[\![S]\!](\rho) = \sum\{|\rho' : \langle S, \rho\rangle \to^* \langle\downarrow, \rho'\rangle|\} \text{ for all } \rho \in \mathcal{D}(\mathcal{H}_{\text{all}})$$

Observation:

$$tr([\![S]\!](\rho)) \leq tr(\rho) \tag{1}$$

for any quantum program S and all $\rho \in \mathcal{D}(\mathcal{H}_{\text{all}})$.

- Exercise: Prove inequality (1). [Hint: Induction on the structure of S.]

Observation:

$$tr([\![S]\!](\rho)) \leq tr(\rho) \tag{1}$$

for any quantum program S and all $\rho \in \mathcal{D}(\mathcal{H}_{\text{all}})$.

- Exercise: Prove inequality (1). [Hint: Induction on the structure of S.]
- $tr(\rho) - tr([\![S]\!](\rho))$ is the probability that program S diverges from input state ρ.

Outline

Questions

- How to specify correctness of programs?

Questions

- How to specify correctness of programs?
- How to prove correctness of programs?

(Floyd)-Hoare logic

Questions

- How to specify correctness of programs?
- How to prove correctness of programs?

(Floyd)-Hoare logic

- A. Turing, Checking a large routine, *Talk given at the inaugural conference of the EDSAC computer*, Cambridge University, 1949.

Questions

- How to specify correctness of programs?
- How to prove correctness of programs?

(Floyd)-Hoare logic

- A. Turing, Checking a large routine, *Talk given at the inaugural conference of the EDSAC computer*, Cambridge University, 1949.

- R. W. Floyd, Assigning meaning to programs, *Proc. Symposia in Applied Math.* 1967

Questions

- How to specify correctness of programs?
- How to prove correctness of programs?

(Floyd)-Hoare logic

- A. Turing, Checking a large routine, *Talk given at the inaugural conference of the EDSAC computer*, Cambridge University, 1949.

- R. W. Floyd, Assigning meaning to programs, *Proc. Symposia in Applied Math.* 1967
- C. A. R. Hoare, An Axiomatic Basis for Computer Programming, *Communications of the ACM* 1969.

Questions

- How to specify correctness of programs?
- How to prove correctness of programs?

(Floyd)-Hoare logic

- A. Turing, Checking a large routine, *Talk given at the inaugural conference of the EDSAC computer*, Cambridge University, 1949.

- R. W. Floyd, Assigning meaning to programs, *Proc. Symposia in Applied Math.* 1967
- C. A. R. Hoare, An Axiomatic Basis for Computer Programming, *Communications of the ACM* 1969.
- S. A. Cook, Soundness and Completeness of an Axiom System for Program Verification, *SIAM J. Comput.* 1978.

Example: square root

Find the greatest y whose square is less than or equal to x

$$
\begin{aligned}
SR \equiv\ & y := 0; \\
& \mathit{while}\ y * y \leq x\ \mathit{do} \\
& \qquad y := y + 1; \\
& y := y - 1
\end{aligned}
$$

Correctness formulas - Hoare triples

- How to specify correctness of programs?

Correctness formulas - Hoare triples

- How to specify correctness of programs?
- Hoare triple:

$$\{\varphi\}P\{\psi\}$$

Example: square root

$$\{x \geq 0\}\ SR\ \{y * y \leq x \wedge (\forall z)(z * z \leq x \rightarrow z \leq y)\}$$

Correctness formulas - Hoare triples

- How to specify correctness of programs?
- Hoare triple:

$$\{\varphi\}P\{\psi\}$$

  - P is a programs;

Example: square root

$$\{x \geq 0\}\ SR\ \{y * y \leq x \wedge (\forall z)(z * z \leq x \rightarrow z \leq y)\}$$

Correctness formulas - Hoare triples

- How to specify correctness of programs?
- Hoare triple:

$$\{\varphi\}P\{\psi\}$$

  - P is a programs;
  - φ, ψ are first-order logic formulas, called pre/postconditions.

Example: square root

$$\{x \geq 0\}\ SR\ \{y * y \leq x \wedge (\forall z)(z * z \leq x \rightarrow z \leq y)\}$$

Verification of programs

- How to prove correctness of a program?

Verification of programs

- How to prove correctness of a program?

Hoare Logic

- Axiomatic System

Axiom-Skip $\quad \{p\}\textbf{skip}\{p\}$

Axiom-Assignment $\quad \{p[u := t]\}u := t\{p\}$

Rule-Composition $\quad \dfrac{\{p\}C_1\{q\} \quad \{q\}C_2\{r\}}{\{p\}C_1; C_2\{r\}}$

Rule-Conditional $\quad \dfrac{\{p \wedge B\}C_1\{q\} \quad \{p \wedge \neg B\}C_2\{q\}}{\{p\}\textbf{if } B \textbf{ then } C_1 \textbf{ else } C_2 \textbf{ fi}\{q\}}$

Rule-Loop $\quad \dfrac{\{p \wedge B\}C\{p\}}{\{p\}\textbf{while } B \textbf{ do } C \textbf{ od}\{p \wedge \neg B\}}$

Rule-Consequence $\quad \dfrac{p \to p' \quad \{p'\}C\{q'\} \quad q' \to q}{\{p\}C\{q\}}$

Quantum Hoare Triples

- A *quantum predicate* is a Hermitian operator (obsevable) P such that $0 \sqsubseteq P \sqsubseteq I$.

  [1] E. D'Hondt and P. Panangaden, Quantum weakest preconditions, *Mathematical Structures in Computer Science* 2006.

Quantum Hoare Triples

- A *quantum predicate* is a Hermitian operator (obsevable) P such that $0 \sqsubseteq P \sqsubseteq I$.

  [1] E. D'Hondt and P. Panangaden, Quantum weakest preconditions, *Mathematical Structures in Computer Science* 2006.

- A *correctness formula* is a statement of the form:

$$\{P\}S\{Q\}$$

  where:

Quantum Hoare Triples

- A *quantum predicate* is a Hermitian operator (obsevable) P such that $0 \sqsubseteq P \sqsubseteq I$.

  [1] E. D'Hondt and P. Panangaden, Quantum weakest preconditions, *Mathematical Structures in Computer Science* 2006.

- A *correctness formula* is a statement of the form:

$$\{P\}S\{Q\}$$

  where:

  - S is a quantum program

Quantum Hoare Triples

- A *quantum predicate* is a Hermitian operator (obsevable) P such that $0 \sqsubseteq P \sqsubseteq I$.

  [1] E. D'Hondt and P. Panangaden, Quantum weakest preconditions, *Mathematical Structures in Computer Science* 2006.

- A *correctness formula* is a statement of the form:

$$\{P\}S\{Q\}$$

  where:

  - S is a quantum program
  - P and Q are quantum predicates.

Quantum Hoare Triples

- A *quantum predicate* is a Hermitian operator (obsevable) P such that $0 \sqsubseteq P \sqsubseteq I$.

  [1] E. D'Hondt and P. Panangaden, Quantum weakest preconditions, *Mathematical Structures in Computer Science* 2006.

- A *correctness formula* is a statement of the form:

$$\{P\}S\{Q\}$$

  where:

  - S is a quantum program
  - P and Q are quantum predicates.
  - Operator P is called the *precondition* and Q the *postcondition*.

Partial and Total Correctness

1. $\{P\}S\{Q\}$ is true in the sense of *total correctness*:

$$\models_{\text{tot}} \{P\}S\{Q\}$$

if

$$tr(P\rho) \leq tr(Q[\![S]\!](\rho)) \text{ for all } \rho.$$

Partial and Total Correctness

1. $\{P\}S\{Q\}$ is true in the sense of *total correctness*:

$$\models_{\text{tot}} \{P\}S\{Q\}$$

if

$$tr(P\rho) \leq tr(Q[\![S]\!](\rho)) \text{ for all } \rho.$$

2. $\{P\}S\{Q\}$ is true in the sense of *partial correctness*:

$$\models_{\text{par}} \{P\}S\{Q\},$$

if

$$tr(P\rho) \leq tr(Q[\![S]\!](\rho)) + [tr(\rho) - tr([\![S]\!](\rho))]$$

for all ρ.

Proof System for Partial Correctness

(Axiom-Sk) $\{P\}\mathbf{Skip}\{P\}$

(Axiom-Ini) $\left\{\sum_n |n\rangle_q\langle 0|P|0\rangle_q\langle n|\right\} q := |0\rangle\{P\}$

(Axiom-Uni) $\{U^\dagger P U\}\ U[\overline{q}]\ \{P\}$

(Rule-Seq) $\dfrac{\{P\}S_1\{Q\} \quad \{Q\}S_2\{R\}}{\{P\}S_1; S_2\{R\}}$

Proof System for Partial Correctness

(Rule-IF) $$\frac{\{P_m\}S_m\{Q\} \text{ for all } m}{\{\sum_m M_m^\dagger P_m M_m\}\, if\ \ (\square m \cdot M[\bar{q}] = m \to S_m)\ fi\{Q\}}$$

(Rule-LP) $$\frac{\{Q\}S\{M_0^\dagger P M_0 + M_1^\dagger Q M_1\}}{\{M_0^\dagger P M_0 + M_1^\dagger Q M_1\}\textbf{while}\, M[\bar{q}] = 1\ \textbf{do}\ S\ \textbf{od}\{P\}}$$

(Rule-Ord) $$\frac{P \sqsubseteq P' \quad \{P'\}S\{Q'\} \quad Q' \sqsubseteq Q}{\{P\}S\{Q\}}$$

Proof System for Total Correctness

Let P be a quantum predicate and $\epsilon > 0$. A function

$$t : \mathcal{D}(\mathcal{H}_{\text{all}}) \text{ (density operators)} \to N$$

is called a (P, ϵ)-*ranking function* of quantum loop:

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

if for all ρ:

1. $t([\![S]\!](M_1 \rho M_1^\dagger)) \leq t(\rho)$;

Proof System for Total Correctness

Let P be a quantum predicate and $\epsilon > 0$. A function

$$t : \mathcal{D}(\mathcal{H}_{\text{all}}) \text{ (density operators)} \to N$$

is called a (P, ϵ)-*ranking function* of quantum loop:

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

if for all ρ:

1. $t(\llbracket S \rrbracket(M_1 \rho M_1^\dagger)) \leq t(\rho)$;
2. $tr(P\rho) \geq \epsilon$ implies $t(\llbracket S \rrbracket(M_1 \rho M_1^\dagger)) < t(\rho)$

Proof System for Total Correctness

$$\text{(Rule-LT)}\quad \frac{\begin{array}{c}(1)\ \{Q\}S\{M_0^\dagger PM_0 + M_1^\dagger QM_1\} \\ (2)\ \text{for any } \epsilon > 0,\ t_\epsilon \text{ is a } (M_1^\dagger QM_1, \epsilon)-\text{ranking} \\ \text{function of loop}\end{array}}{\{M_0^\dagger PM_0 + M_1^\dagger QM_1\}\textbf{while}\ M[\bar{q}] = 1\ \textbf{do}\ S\ \textbf{od}\{P\}}$$

Theorem (Soundness and Completeness)

For any quantum program S and quantum predicates P Q,

$$\models_{\text{par}} \{P\}S\{Q\} \text{ if and only if } \vdash_{PD} \{P\}S\{Q\}$$
$$\models_{\text{tot}} \{P\}S\{Q\} \text{ if and only if } \vdash_{TD} \{P\}S\{Q\}.$$

- Exercise: Prove the soundness. [Hint: Induction on the height of proof tree.]

[2] M. S. Ying, Floyd-Hoare logic for quantum programs, *TOPLAS* 2011

Outline

QHL Theorem Provers

- QHLProver based on Isabelle/HOL

  [3] J. Y. Liu, B. H. Zhan et al., Formal verification of quantum algorithms using quantum Hoare logic, *CAV* 2019

QHL Theorem Provers

- QHLProver based on Isabelle/HOL

  [3] J. Y. Liu, B. H. Zhan et al., Formal verification of quantum algorithms using quantum Hoare logic, *CAV* 2019

- CoqQ = QHLProver based on Rocq/Coq

  [4] L. Zhou, G. Barthe, P. -Y. Strub, J. Y. Liu and M. S. Ying, CoqQ: foundational verification of quantum programs, *POPL* 2023

Invariant Generation

[5] M. S. Ying, S. G. Ying and X. D. Wu, Invariants of quantum programs: characterisation and generation, *POPL* 2017.

QHL Theorem Provers

- QHLProver based on Isabelle/HOL

  [3] J. Y. Liu, B. H. Zhan et al., Formal verification of quantum algorithms using quantum Hoare logic, *CAV* 2019

- CoqQ = QHLProver based on Rocq/Coq

  [4] L. Zhou, G. Barthe, P. -Y. Strub, J. Y. Liu and M. S. Ying, CoqQ: foundational verification of quantum programs, *POPL* 2023

Invariant Generation

[5] M. S. Ying, S. G. Ying and X. D. Wu, Invariants of quantum programs: characterisation and generation, *POPL* 2017.

Termination Analysis

[6] Y. J. Li and M. S. Ying, Algorithmic analysis of termination problem for quantum programs, *POPL* 2018.

THANKS!