Foundations of Quantum Programming (Lecture 4)

Mingsheng Ying

Centre for Quantum Software and Information
University of Technology Sydney

Outline

Outline

Outline

Limitations of QWhile

- No classical variables

Limitations of QWhile

- No classical variables
- Poor data types/structures

Enhanced Expressivity of Programming Language

Limitations of QWhile

- No classical variables
- Poor data types/structures

Enhanced Expressivity of Programming Language

- Classical variables

Limitations of QWhile

- No classical variables
- Poor data types/structures

Enhanced Expressivity of Programming Language

- Classical variables
- Quantum arrays

Limitations of QWhile

- No classical variables
- Poor data types/structures

Enhanced Expressivity of Programming Language

- Classical variables
- Quantum arrays
- Parameterized quantum gates

Quantum Types

- A basic quantum type $\mathcal{H}$ denotes an intended Hilbert space.

Quantum Types

- A basic quantum type $\mathcal{H}$ denotes an intended Hilbert space.
- A higher quantum type is defined of the form:

$$T_1 \times ... \times T_n \to \mathcal{H}$$

where $T_1, ..., T_n$ are basic classical types, and $\mathcal{H}$ is a basic quantum type.

$$\mathcal{H}^{\otimes(T_1 \times ... \times T_n)} = \bigotimes_{t_1 \in T_1, ..., t_n \in T_n} \mathcal{H}$$

Quantum Types

- A basic quantum type $\mathcal{H}$ denotes an intended Hilbert space.
- A higher quantum type is defined of the form:

$$T_1 \times ... \times T_n \to \mathcal{H}$$

where $T_1, ..., T_n$ are basic classical types, and $\mathcal{H}$ is a basic quantum type.

$$\mathcal{H}^{\otimes(T_1 \times ... \times T_n)} = \bigotimes_{t_1 \in T_1, ..., t_n \in T_n} \mathcal{H}$$

- Example: A qubit array q of type **integer** $\to \mathcal{H}_2$
  Section $q[k : l]$ stands for the restriction of q to the interval $[k : l] = \{\text{integer } i \mid k \leq i \leq l\}$.

Quantum Variables

- Simple quantum variables, of a basic quantum type, say $\mathcal{H}$;

Quantum Variables

- Simple quantum variables, of a basic quantum type, say $\mathcal{H}$;
- Array quantum variables, of a higher quantum type, say $T_1 \times ... \times T_n \to \mathcal{H}$.

Quantum Variables

- Simple quantum variables, of a basic quantum type, say $\mathcal{H}$;
- Array quantum variables, of a higher quantum type, say $T_1 \times ... \times T_n \to \mathcal{H}$.
- Let q be an array quantum variable of the type $T_1 \times ... \times T_n \to \mathcal{H}$, and for each $1 \leq i \leq n$, let s_i be a classical expression of type T_i. Then

  $$q[s_1, ..., s_n]$$

  is a subscripted quantum variable of type $\mathcal{H}$.

Quantum Variables

- Simple quantum variables, of a basic quantum type, say $\mathcal{H}$;
- Array quantum variables, of a higher quantum type, say $T_1 \times ... \times T_n \to \mathcal{H}$.
- Let q be an array quantum variable of the type $T_1 \times ... \times T_n \to \mathcal{H}$, and for each $1 \leq i \leq n$, let s_i be a classical expression of type T_i. Then

  $$q[s_1, ..., s_n]$$

  is a subscripted quantum variable of type $\mathcal{H}$.
- Example: If q : **integer** $\times$ **integer** $\to \mathcal{H}_2$ is a two-dimensional array of qubits, and x, y are integer variables, then

  $$q[2x - y, 3x + 4y - 75]$$

  is a subscripted qubit variable.

Syntax of Quantum Programming Language **qWhile**$^+$

$$
\begin{aligned}
P ::= {} & \textbf{skip} \mid x := e \mid q := |0\rangle \\
& \mid U(t_1, ..., t_m)[q_1, ..., q_n] \\
& \mid x := M[q_1, ..., q_n] \\
& \mid P_1; P_2 \mid \textbf{if } b \textbf{ then } P_1 \textbf{ else } P_0 \mid \textbf{while } b \textbf{ do } P
\end{aligned}
$$

Example - Circuit of QFT on n qubits

$QFT[q_1, ..., q_n] ::=$

$H[q_1]; C(R_2)[q_2, q_1]; C(R_3)[q_3, q_1]; ...; C(R_{n-1})[q_{n-1}, q_1]; C(R_n)[q_n, q_1];$

$H[q_2]; C(R_2)[q_3, q_2]; C(R_3)[q_4, q_2]; ...; C(R_{n-1})[q_n, q_2];$

.....

$H[q_{n-1}]; C(R_2)[q_n, q_{n-1}];$

$H[q_n];$

$Reverse[q_1, ..., q_n]$

where:

- $C(R_l)[q_i, q_j]$ — controlled-R_l:

$$R_l = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^l} \end{pmatrix}$$

Example - Circuit of QFT on n qubits

$QFT[q_1, ..., q_n] ::=$

$H[q_1]; C(R_2)[q_2, q_1]; C(R_3)[q_3, q_1]; ...; C(R_{n-1})[q_{n-1}, q_1]; C(R_n)[q_n, q_1];$

$H[q_2]; C(R_2)[q_3, q_2]; C(R_3)[q_4, q_2]; ...; C(R_{n-1})[q_n, q_2];$

.....

$H[q_{n-1}]; C(R_2)[q_n, q_{n-1}];$

$H[q_n];$

$Reverse[q_1, ..., q_n]$

where:

- $C(R_l)[q_i, q_j]$ — controlled-R_l:

$$R_l = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^l} \end{pmatrix}$$

- $Reverse[q_1, ..., q_n]$ is the quantum gate that reverses the order of qubits $q_1, ..., q_n$.

Example - QFT in **qWhile**$^+$

Introduce qubit array q.

$$QFT[q[1:n]] ::= QFT^*[q[1:n]]; Reverse[q[1:n]],$$
$$QFT^*[q[m:n]] ::= \textbf{if } m = n \textbf{ then } H[q[n]]$$
$$\textbf{else } H[Q[m]]; CR[q[m:n]]; QFT^*[q[m+1:n]],$$
$$CR[q[m:n]] ::= \textbf{if } m = n \textbf{ then skip}$$
$$\textbf{else } CR[q[m:n-1]]; C(R_{n-m+1})[q_n, q_m].$$

Outline

Limitations of QHL

- Curse of dimensionality: a quantum predicate for n qubits is a $2^n \times 2^n$ matrix —— ***a bottleneck for scalable verification***.

Limitations of QHL

- Curse of dimensionality: a quantum predicate for n qubits is a $2^n \times 2^n$ matrix —— ***a bottleneck for scalable verification***.
- Assertion language?

Limitations of QHL

- Curse of dimensionality: a quantum predicate for n qubits is a $2^n \times 2^n$ matrix —— ***a bottleneck for scalable verification***.
- Assertion language?
  - An assertion for a classical program is a predicate —— Boolean-valued function, over the state space.

Limitations of QHL

- Curse of dimensionality: a quantum predicate for n qubits is a $2^n \times 2^n$ matrix —— ***a bottleneck for scalable verification***.
- Assertion language?
  - An assertion for a classical program is a predicate —— Boolean-valued function, over the state space.
  - It can be represented by a first-order logical formula —— constructed from atomic formulas using connectives and quantifiers.

Limitations of QHL

- Curse of dimensionality: a quantum predicate for n qubits is a $2^n \times 2^n$ matrix —— ***a bottleneck for scalable verification***.
- Assertion language?
  - An assertion for a classical program is a predicate —— Boolean-valued function, over the state space.
  - It can be represented by a first-order logical formula —— constructed from atomic formulas using connectives and quantifiers.
  - Often much more economic than as a Boolean-valued function over the entire state space.

Limitations of QHL

- Curse of dimensionality: a quantum predicate for n qubits is a $2^n \times 2^n$ matrix —— ***a bottleneck for scalable verification***.
- Assertion language?
  - An assertion for a classical program is a predicate —— Boolean-valued function, over the state space.
  - It can be represented by a first-order logical formula —— constructed from atomic formulas using connectives and quantifiers.
  - Often much more economic than as a Boolean-valued function over the entire state space.
  - How to define an assertion language for quantum programs?

Correctness Specifications

- Syntax of quantum predicates:

$$A ::= K(\bar{t})[\bar{q}] \mid \neg A \mid A_1 \otimes A_2 \mid F(\bar{t})[\bar{q}](\{A_i\}).$$

Correctness Specifications

- Syntax of quantum predicates:

$$A ::= K(\bar{t})[\bar{q}] \mid \neg A \mid A_1 \otimes A_2 \mid F(\bar{t})[\bar{q}](\{A_i\}).$$

- Hoare triples:

$$\{\varphi, A\}\ P\ \{\psi, B\}$$

Correctness Specifications

- Syntax of quantum predicates:

$$A ::= K(\bar{t})[\bar{q}] \mid \neg A \mid A_1 \otimes A_2 \mid F(\bar{t})[\bar{q}](\{A_i\}).$$

- Hoare triples:

$$\{\varphi, A\}\ P\ \{\psi, B\}$$

  - φ, ψ are first-order logical formulas;

Correctness Specifications

- Syntax of quantum predicates:

$$A ::= K(\bar{t})[\bar{q}] \mid \neg A \mid A_1 \otimes A_2 \mid F(\bar{t})[\bar{q}](\{A_i\}).$$

- Hoare triples:

$$\{\varphi, A\}\ P\ \{\psi, B\}$$

  - φ, ψ are first-order logical formulas;
  - A, B are quantum predicates parameterized by classical variables

Partial and Total Correctness

- $\{\varphi, A\}\ P\ \{\psi, B\}$ is true in the sense of total correctness, written

$$\models_{tot} \{\varphi, A\}\ P\ \{\psi, B\},$$

if for any input (σ, ρ) with $\sigma \models \varphi$, it holds that

$$\mathrm{tr}(\sigma(A)\rho) \leq \sum \left\{ |\mathrm{tr}\left(\sigma'(B)\rho'\right) | (\sigma', \rho') \in [\![P]\!](\sigma, \rho) \text{ and } \sigma' \models \psi| \right\}.$$

Partial and Total Correctness

- $\{\varphi, A\}\ P\ \{\psi, B\}$ is true in the sense of total correctness, written

$$\models_{tot} \{\varphi, A\}\ P\ \{\psi, B\},$$

if for any input (σ, ρ) with $\sigma \models \varphi$, it holds that

$$\mathrm{tr}(\sigma(A)\rho) \leq \sum \left\{ |\mathrm{tr}\left(\sigma'(B)\rho'\right) | (\sigma', \rho') \in [\![P]\!](\sigma, \rho) \text{ and } \sigma' \models \psi| \right\}.$$

- Partial correctness $\models_{par} \{\varphi, A\}\ P\ \{\psi, B\}$:

$$\mathrm{tr}(\sigma(A)\rho) \leq \sum \left\{ |\mathrm{tr}\left(\sigma'(B)\rho'\right) | (\sigma', \rho') \in [\![P]\!](\sigma, \rho) \text{ and } \sigma' \models \psi| \right\} + NT(P)(\sigma, \rho)$$

where $NT(P)(\sigma, \rho)$ — nontermination probability.

Proof system

- The proof system can be conveniently combined with classical first-order logic.

Proof system

- The proof system can be conveniently combined with classical first-order logic.

Proof System - Partial Correctness

$$(\textit{Axiom-Ski}) \quad \{\varphi, A\}\ \textbf{skip}\ \{\varphi, A\}$$

$$(\textit{Axiom-Ass}) \quad \{\varphi[e/x], A[e/x]\}\ x := e\ \{\varphi, A\}$$

$$(\textit{Axiom-Init}) \quad \{\varphi, F_B[q](A)\}\ q := |0\rangle\ \{\varphi, A\}$$

$$(\textit{Axiom-Uni}) \quad \{\varphi, F_U(\bar{t})[\bar{q}](A)\}\ U(\bar{t})[\bar{q}]\ \{\varphi, A\}$$

$$(\textit{Axiom-Meas}) \quad \frac{y \notin \textit{free}(\varphi) \cup \textit{cv}(A) \cup \{x\}}{\{\varphi[y/x], F_M(y)[\bar{q}](A[y/x])\}\ x := M[\bar{q}]\ \{\varphi \wedge x = y, A\}}$$

$$(\textit{Rule-Seq}) \quad \frac{\{\varphi, A\}\ P_1\ \{\psi, B\} \qquad \{\psi, B\}\ P_2\ \{\theta, , C\}}{\{\varphi, A\}\ P_1; P_2\ \{\theta, C\}}$$

$$(\textit{Rule-Cond}) \quad \frac{\{\varphi \wedge b, A\}\ P_1\ \{\psi, B\} \qquad \{\varphi \wedge \neg b, A\}\ P_0\ \{\psi, B\}}{\{\varphi, A\}\ \textbf{if}\ b\ \textbf{then}\ P_1\ \textbf{else}\ P_0\ \{\psi, B\}}$$

Proof System - Partial Correctness (Continued)

$$(\textit{Rule-Loop-par}) \quad \frac{\{\varphi \wedge b, A\}\ P\ \{\varphi, A\}}{\{\varphi, A\}\ \textbf{while}\ b\ \textbf{do}\ P\ \{\varphi \wedge \neg b, A\}}$$

$$(\textit{Rule-Conseq}) \quad \frac{(\varphi', A') \models (\varphi, A) \quad \{\varphi, A\}\ P\ \{\psi, B\} \quad (\psi, B) \models (\psi', B')}{\{\varphi', A'\}\ P\ \{\psi', B'\}}$$

$$(\textit{Rule-Accum1}) \quad \frac{\begin{array}{c}\{\varphi, A_i\}\ P\ \{\psi_i, B\} \text{ for every } i = 1, ..., n \\ (\forall i_1, i_2)\ \left(i_1 \neq i_2 \rightarrow \neg(\psi_{i_1} \wedge \psi_{i_2})\right)\end{array}}{\left\{\varphi, \frac{1}{n}\sum_{i=1}^{n} A_i\right\}\ P\ \left\{\bigvee_{i=1}^{n} \psi_i, \frac{1}{n} B\right\}}$$

$$(\textit{Rule-Accum2}) \quad \frac{\begin{array}{c}\{\varphi, A_i\}\ P\ \{\psi, B_i\} \text{ for every } i \\ 0 \leq p_i \text{ for every } i \qquad \sum_i p_i \leq 1\end{array}}{\{\varphi, \textstyle\sum_i p_i A_i\}\ P\ \{\psi, \sum_i p_i B_i\}}$$

Proof System - Total Correctness

$$(\textit{Rule-Loop-tot}) \qquad \frac{\begin{array}{c}\{\varphi \wedge b, A\}\ P\ \{\varphi, A\} \\ \{\varphi \wedge b \wedge t = z, I\}\ P\ \{t < z, I\} \\ \varphi \rightarrow t \geq 0\end{array}}{\{\varphi, A\}\ \textbf{while}\ b\ \textbf{do}\ P\ \{\varphi \wedge \neg b, A\}}$$

Proof System - Total Correctness

$$(\textit{Rule-Loop-tot}) \qquad \frac{\begin{array}{c}\{\varphi \wedge b, A\}\ P\ \{\varphi, A\} \\ \{\varphi \wedge b \wedge t = z, I\}\ P\ \{t < z, I\} \\ \varphi \to t \geq 0\end{array}}{\{\varphi, A\}\ \textbf{while}\ b\ \textbf{do}\ P\ \{\varphi \wedge \neg b, A\}}$$

Soundness Theorem

For any quantum program P, classical first-order logical formulas φ, ψ, and quantum predicate formulas A, B:

$$\vdash_{QHL^{+}_{par}} \{\varphi, A\}\ P\ \{\psi, B\} \text{ implies } \models_{par} \{\varphi, A\}\ P\ \{\psi, B\};$$

$$\vdash_{QHL^{+}_{tot}} \{\varphi, A\}\ P\ \{\psi, B\} \text{ implies } \models_{tot} \{\varphi, A\}\ P\ \{\psi, B\}.$$

[1] M. S. Ying, A practical quantum Hoare logical with classical variables, I, *Information and Computation* 2026 (also see *arXiv* 2412.09869).

QHL$^+$ Theorem Prover

- Two teams are currently working on the AI-assisted formalisation of QHL$^+$ in Lean 4.

QHL^+ Theorem Prover

- Two teams are currently working on the AI-assisted formalisation of QHL^+ in Lean 4.

- Classical program verification:

  Program $\Rightarrow$ VC (verification condition) generation
  $\Rightarrow$ First-order logical formula $\Rightarrow$ SAT/SMT solver

QHL$^+$ Theorem Prover

- Two teams are currently working on the AI-assisted formalisation of QHL$^+$ in Lean 4.

- Classical program verification:

  Program $\Rightarrow$ VC (verification condition) generation
  $\Rightarrow$ First-order logical formula $\Rightarrow$ SAT/SMT solver

- Quantum program verification:

  Program $\Rightarrow$ VC (verification condition) generation
  $\Rightarrow$ Matrix inequality $\Rightarrow$ SDP solver +???

Outline

Open Question

- (Relative) Completeness?

Open Question

- (Relative) Completeness?

Scalable Verification

- Separation logic

  [2] X. -B. Le, et al., A quantum interpretation of separating conjunction for local reasoning of quantum programs based on separation logic, *POPL* 2022.
  [3] L. Zhou, et al., A quantum interpretation of bunched logic for quantum separation logic, *LICS* 2021.

Open Question

- (Relative) Completeness?

Scalable Verification

- Separation logic

  [2] X. -B. Le, et al., A quantum interpretation of separating conjunction for local reasoning of quantum programs based on separation logic, *POPL* 2022.
  [3] L. Zhou, et al., A quantum interpretation of bunched logic for quantum separation logic, *LICS* 2021.

- AI for Formalisation and Verification???

Applications

- Classical-quantum hybrid reasoning for QEC

  [4] W. Fang et al., Symbolic execution for quantum error correction programs, *PLDI* 2024.
  [5] Q. F. Huang, et al., Efficient formal verification of quantum error correcting programs, *PLDI* 2025.
  [6] K. A. Chen, et al., Verifying fault-tolerance of quantum error correction codes, *CAV* 2025.

Applications

- Classical-quantum hybrid reasoning for QEC

  [4] W. Fang et al., Symbolic execution for quantum error correction programs, *PLDI* 2024.
  [5] Q. F. Huang, et al., Efficient formal verification of quantum error correcting programs, *PLDI* 2025.
  [6] K. A. Chen, et al., Verifying fault-tolerance of quantum error correction codes, *CAV* 2025.

- Incorrectness logic for debugging

  [7] P. Yan, et al., On incorrectness logic for quantum programs, *OOPSLA* 2022.

Applications

- Relational reasoning for quantum compilers + security

[8] D. Unruh, Quantum relational Hoare logic, *POPL* 2019.
[9] G. Barthe, et al., Relational proofs for quantum programs, *POPL* 2020.
[10] M. Barbosa, et al., EasyPQC: verifying post-quantum cryptography, *CCS* 2021.

Applications

- Relational reasoning for quantum compilers + security

  [8] D. Unruh, Quantum relational Hoare logic, *POPL* 2019.
  [9] G. Barthe, et al., Relational proofs for quantum programs, *POPL* 2020.
  [10] M. Barbosa, et al., EasyPQC: verifying post-quantum cryptography, *CCS* 2021.

- Concurrency Reasoning for QOS???

  [11] J. Z. Ye, ..., J. Palsberg, HALO: A fine-grained resource sharing quantum operating system, *arXiv* 2602.07191.
  [12] 2. C. Delle Donne, et al., An operating system for executing applications on quantum network nodes, *Nature* 2025.
  [13] E. Giortamis, et al., QOS: quantum operation system, *OSDI* 2025.

Thanks!