# Automated Quantum Protocol Verification Based on Concurrent Dynamic Quantum Logic

Canh Minh Do

Japan Advanced Institute of Science and Technology (JAIST)

1-8 Asahidai, Nomi, Japan

Presented Online at La Trobe-Kyushu Joint Seminar on Mathematics for Industry
July 2, 2024

## Contents

## Contents

## Introduction

- Quantum computing is a rapidly emerging technology that uses the laws of quantum mechanics to solve complex problems beyond the capabilities of classical computers, such as Shore's fast algorithms[1] for discrete logarithms and factoring.
- Due to radically different principles of quantum mechanics, such as superposition, entanglement, and measurement, it is challenging to accurately design and implement quantum algorithms, quantum programs, and quantum protocols.
- Therefore, it is crucial to ensure the correctness of quantum systems through verification.

[1] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994.

## Formal Verification of Quantum Programs

- Previous Studies of Quantum Program Verification
  - Quantum Hoare Logic (QHL)[2]: a quantum counterpart of Hoare Logic
  - Dynamic Quantum Logic (DQL)[3]: a quantum counterpart of Dynamic Logic
- Problems of Previous Studies
  - QHL can semi-automatically perform proofs of correctness with a support tool[4] implemented in Coq. Meanwhile, DQL still requires manual proof verification.
  - In this study, we propose an automatic verification method based on Concurrent Dynamic Quantum Logic (CDQL), an extended version of Basic Dynamic Quantum Logic (BDQL)[5], for describing concurrent behavior and communication among participants in quantum protocols.

[2] Mingsheng Ying. "Floyd–Hoare Logic for Quantum Programs". In: *ACM Trans. Program. Lang. Syst.* (2012).

[3] Alexandru Baltag and Sonja Smets. "Reasoning about Quantum Information: An Overview of Quantum Dynamic Logic". In: *Applied Sciences* (2022).

[4] Junyi Liu et al. "Formal Verification of Quantum Algorithms Using Quantum Hoare Logic". In: *Computer Aided Verification*. 2019.

[5] Tsubasa Takagi, Canh Minh Do, and Kazuhiro Ogata. "Automated Quantum Program Verification in a Dynamic Quantum Logic". In: *DaLí: Dynamic Logic – New trends and applications*. 2023.

## Contents

## Hilbert Spaces

- A Hilbert space $\mathcal{H}$ usually serves as the state space of a quantum system that is a complex vector space equipped with an inner product such that each Cauchy sequence of vectors has a limit.
- An $n$-qubit system is the complex $2^n$-space $\mathbb{C}^{2^n}$, where $\mathbb{C}$ stands for the complex plane.
- Pure states in the $n$-qubit systems $\mathbb{C}^{2^n}$ are unit vectors in $2^n$-space $\mathbb{C}^{2^n}$.
- The orthogonal basis called computational basis in the one-qubit system $\mathbb{C}^2$ is the set $\{|0\rangle, |1\rangle\}$ that consists of the column vectors $|0\rangle = (1,0)^T$ and $|1\rangle = (0,1)^T$, where $^T$ denotes the transpose operator.
- In the two-qubit system $\mathbb{C}^4$, there are pure states that cannot be represented in the form $|\psi_1\rangle \otimes |\psi_2\rangle$ and called entangled states, where $\otimes$ denotes the tensor product (more precisely, the Kronecker product).
- For example, the EPR state (Einstein-Podolsky-Rosen state) $|EPR\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ is an entangled state, where $|00\rangle = |0\rangle \otimes |0\rangle$ and $|11\rangle = |1\rangle \otimes |1\rangle$.

## Unitary Operators

- A unitary operator is a linear operator $U : \mathcal{H} \to \mathcal{H}$ that satisfies $U^* U = UU^* = I$.
- Quantum computation is represented by unitary operators (also called quantum gates).
- For example, the Hadamard gate $H$ and Pauli gates $X$, $Y$, and $Z$ are quantum gates on the one-qubit system $\mathbb{C}^2$ and are defined as follows:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

- Two typical quantum gates on the two-qubit systems $\mathbb{C}^4$ are the controlled-X gate (also called the controlled-NOT gate) $CX$ and the swap gate $SWAP$ are defined by

$$CX = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X,$$
$$SWAP = CX(I \otimes |0\rangle\langle 0| + X \otimes |1\rangle\langle 1|)CX,$$

where $I$ denotes the identity matrix of size $2 \times 2$.

## Measurement

- Measurement is a completely different process from applying quantum gates. Here we roughly explain specific projective measurements.
- For the general definition of projective measurement, see the famous textbook of quantum computation[6].
- Observe that $P_0 = |0\rangle\langle 0|$ and $P_1 = |1\rangle\langle 1|$ are projectors, respectively.
- After executing the measurement $\{P_0, P_1\}$, a current state $|\psi\rangle = c_0 |0\rangle + c_1 |1\rangle$ is collapsed into either $\frac{P_0|\psi\rangle}{|c_0|}$ with probability $|c_0|^2$ or into $\frac{P_1|\psi\rangle}{|c_1|}$ with probability $|c_1|^2$.

$$\begin{tikzcd} & \frac{c_0|0\rangle}{|c_0|} \approx |0\rangle \\ |\psi\rangle \arrow[ur, "|c_0|^2"] \arrow[dr, "|c_1|^2"'] & \\ & \frac{c_1|1\rangle}{|c_1|} \approx |1\rangle \end{tikzcd}$$

[6] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

## Contents

## Regular Program

| Program | Name | Meaning |
|---|---|---|
| **skip** | Skip | Do nothing. |
| **abort** | Abort | Forcing to halt. |
| $a\ ;\ b$ | Composition | Execute a and then execute b. |
| $a \cup b$ | Non-deterministic Choices | Execute either $a$ or $b$ non-deterministically. |
| $a^*$ | Iteration | Repeat $a$ some finite number of times. |
| $p?$ | Test | Confirm that $p$ is whether true or false. |

- Regular Program = Regular Expression + Test
- Conditional/Loop programs can be defined in terms of regular programs
  - if $A$ then $a$ else $b$ fi $= (A?\ ;\ a) \cup (\neg A?\ ;\ b)$
  - if $A_1 \rightarrow a_1 | \ldots | A_n \rightarrow a_n$ fi $= (A_1?\ ;\ a_1) \cup \ldots \cup (A_n?\ ;\ a_n)$
  - while $A$ do $a$ od $= (A?\ ;\ a)^*\ ;\ \neg A?$
  - repeat $a$ until $A = a\ ;\ (\neg A?\ ;\ a)^*\ ;\ A?$

## Dynamic Logic

- Dynamic Logic = Formulas + Regular Programs + Dynamic Operator $[a]$
- The set $L$ of all formulas and the set $\Pi$ of all regular programs are defined by the following simultaneous induction:

$$L \ni A ::= p \mid \neg A \mid A \wedge A \mid [a]A,$$
$$\Pi \ni a ::= \text{skip} \mid \text{abort} \mid \pi \mid a \,;\, a \mid a^* \mid a \cup a \mid A?,$$

where $p$ denotes an atomic formula and $\pi$ denotes an atomic program.

| **Formula** | **Name** | **Meaning** |
|---|---|---|
| $\neg A$ | Negation | Not $A$ |
| $A \wedge B$ | Conjunction | $A$ and $B$ |
| $[a]A$ | Dynamic Operator | It is always $A$ after $a$ is executed |

☞ Dynamic Logic is compatible with formal verification because it can express exhaustive searches.

## Semantics of DQL

- For the sake of simplicity, we use regular programs $\Pi^-$ without the iteration operator $^*$.

**Definition 1**

**Quantum dynamic frame** is a pair $(\mathcal{H}, v)$ of a Hilbert space $\mathcal{H}$ and a function $v$ from the set $\Pi_0$ of all atomic programs to the set $\mathcal{U}(\mathcal{H})$ of all unitary operators on $\mathcal{H}$. Here, $v$ is called an interpretation function of atomic programs.

**Definition 2**

**Quantum dynamic model** is a triple $(\mathcal{H}, v, V)$ that consists of a quantum dynamic frame $(\mathcal{H}, v)$ and a function $V$ from the set $L_0$ of all atomic formulas to the set $\mathcal{C}(\mathcal{H})$ of all closed subspaces of $\mathcal{H}$. Here, $V$ is called an interpretation function of atomic formulas.

- Quantum logic interprets formulas as closed subspaces.

## Semantics of DQL

For each quantum dynamic model $M = (\mathcal{H}, v, V)$, the function $[\![\ ]\!]^M : L \to \mathcal{C}(\mathcal{H})$ and family $\{R_a^M : a \in \Pi^-\}$ of relations on $\mathcal{H}$ are defined by simultaneous induction as follows:

1. $[\![p]\!]^M = V(p)$;
2. $[\![\neg A]\!]^M$ is the orthogonal complement of $[\![A]\!]^M$;
3. $[\![A \wedge B]\!]^M = [\![A]\!]^M \cap [\![B]\!]^M$;
4. $[\![[a]A]\!]^M = \{s \in \mathcal{H} : (s,t) \in R_a^M$ implies $t \in [\![A]\!]^M$ for any $t \in \mathcal{H}\}$;
5. $R_{\text{skip}}^M = \{(s,t) : s = t\}$;
6. $R_{\text{abort}}^M = \emptyset$;
7. $R_\pi^M = \{(s,t) : (v(\pi))(s) = t\}$;
8. $R_{a;b}^M = \{(s,t) : (s,u) \in R_a^M$ and $(u,t) \in R_b^M$ for some $u \in \mathcal{H}\}$;
9. $R_{a \cup b}^M = R_a^M \cup R_b^M$;
10. $R_{A?}^M = \{(s,t) : P_{[\![A]\!]^M}(s) = t\}$, where $P_{[\![A]\!]^M}$ stands for the projection onto $[\![A]\!]^M$.

## Semantics of DQL

Kripke frames can be constructed from the family $\{R_a^M : a \in \Pi^-\}$ of relations on $\mathcal{H}$.

**Example 1 (A Kripke frame for a simple quantum program)**

$$S = \left\{ |00\rangle, \frac{|00\rangle + |10\rangle}{\sqrt{2}}, \frac{|00\rangle + |11\rangle}{\sqrt{2}}, |11\rangle \right\}, \quad R = R_{H(0)} \cup R_{CX(0,1)} \cup R_{|0\rangle ?} \cup R_{|1\rangle ?}$$

$$R_{H(0)} = \left\{ \left( |00\rangle, \frac{|00\rangle + |10\rangle}{\sqrt{2}} \right) \right\}, \quad R_{CX(0,1)} = \left\{ \left( \frac{|00\rangle + |10\rangle}{\sqrt{2}}, \frac{|00\rangle + |11\rangle}{\sqrt{2}} \right) \right\},$$

$$R_{|0\rangle ?} = \left\{ \left( \frac{|00\rangle + |11\rangle}{\sqrt{2}}, |00\rangle \right) \right\}, \quad R_{|1\rangle ?} = \left\{ \left( \frac{|00\rangle + |11\rangle}{\sqrt{2}}, |11\rangle \right) \right\}$$

$$\begin{tikzcd}
 & & & {|00\rangle} \\
{|00\rangle} \arrow[r, "H(0)"] & {\frac{|00\rangle+|10\rangle}{\sqrt{2}}} \arrow[r, "CX(0,1)"] & {\frac{|00\rangle+|11\rangle}{\sqrt{2}}} \arrow[ur, "|0\rangle ?"] \arrow[dr, "|1\rangle ?"'] & \\
 & & & {|11\rangle}
\end{tikzcd}$$

## Semantics of DQL

- Henceforth, we write $(M, s) \models A$ for $s \in [\![A]\!]^M$.
- $(M, s) \models A$ if and only if $P_{[[A]]^M}(s) = s$.

☞ There is a bijection between a closed subspace and a projection onto it.

**Theorem 1**

For any $M$ and $s \in \mathcal{H}$, the following holds:

1. $(M, s) \models A \wedge B$, if and only if $(M, s) \models A$ and $(M, s) \models B$.
2. $(M, s) \models [\text{skip}]A$ if and only if $(M, s) \models A$.
3. $(M, s) \models [\text{abort}]A$.
4. $(M, s) \models [\pi]A$ if and only if $(M, (v(\pi))(s)) \models A$.
5. $(M, s) \models [a\ ;\ b]A$ if and only if $(M, s) \models [a][b]A$.
6. $(M, s) \models [a \cup b]A$ if and only if $(M, s) \models [a]A \wedge [b]A$.
7. $(M, s) \models [A?]B$ if and only if $(M, P_{[\![A]\!]^M}(s)) \models B$.

## Contents

## Motivation

Quantum Teleportation is a quantum communication protocol for teleporting an arbitrary pure state by sending two bits of classical information.

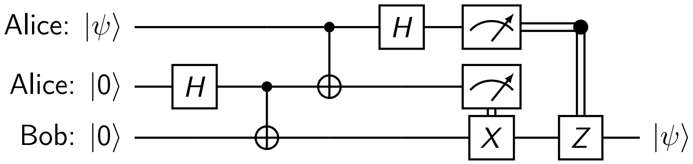


Figure: Quantum Teleportation

## Motivation

The protocol is described as follows:

1. **Preparing Quantum Channel**: Alice has the first qubit in a state $|\psi\rangle$ and the second qubit in the state $|0\rangle$, and Bob has the third qubit in the state $|0\rangle$. After executing $\mathtt{H}(1)\ ;\ \mathtt{CX}(1,2)$, Alice's second qubit and Bob's qubit are entangled.
2. **Alice's Action** 1: Alice executes $\mathtt{CX}(0,1)$ to her qubits.
3. **Alice's Action** 2: Alice executes $\mathtt{H}(0)$ to her qubit.
4. **Alice's Action** 3: Alice measures her second qubit in the computational basis $\{|0\rangle, |1\rangle\}$, and then sends the outcome (either 0 or 1) to Bob via the channel $c_{12}$.
5. **Alice's Action** 4: Alice measures her first qubit in the computational basis $\{|0\rangle, |1\rangle\}$, and then sends the outcome (either 0 or 1) to Bob via the channel $c_{02}$.
6. **Bob's Action** 1: If Bob receives the information 0 via $c_{12}$, then he does nothing. Otherwise, Bob receives the information 1 via $c_{12}$ and then he executes $\mathtt{X}(2)$.
7. **Bob's Action** 2: If Bob receives the information 0 via $c_{02}$, then he does nothing. Otherwise, Bob receives the information 1 via $c_{02}$ and then he executes $\mathtt{Z}(2)$.

## Motivation

- In this protocol, the two processes (Alice's actions and Bob's actions) are executed concurrently. For example, one schedule for this protocol is executing (1), (2), (3), (4), (6), (5), (7) in this order, and another schedule is executing (1), (2), (3), (4), (6), (7), (5) in this order.
- For this reason, to verify the correctness of Quantum Teleportation executed concurrently, we need to carefully define the parallel composition $\|$ with communication. Formally, the program of Quantum Teleportation is described as follows:

$$\begin{aligned}
\text{teleport} &= \mathtt{H}(1) \; ; \mathtt{CX}(1,2) \; ; (\text{Alice} \parallel \text{Bob}), \\
\text{Alice} &= \mathtt{CX}(0,1) \; ; \mathtt{H}(0) \\
&\quad ; \text{if } p(1, |0\rangle) \text{ then } c_{12} <: 0 \text{ else } c_{12} <: 1 \text{ fi} \\
&\quad ; \text{if } p(0, |0\rangle) \text{ then } c_{02} <: 0 \text{ else } c_{02} <: 1 \text{ fi}, \\
\text{Bob} &= ((c_{12} :> 1 \; ; \mathtt{X}(2)) \cup c_{12} :> 0) \\
&\quad ; ((c_{02} :> 1 \; ; \mathtt{Z}(2)) \cup c_{02} :> 0).
\end{aligned}$$

## Syntax of CDQL

- We use the idea of parallel composition from the Algebra of Communicating Processes (ACP)[7] to express concurrent behavior and communication. Similar to ACP, we define $\parallel$ using the left merge operator $⌊\!⌊$ and the communication operator $|$.
- The set $\widehat{L}$ of all formulas in CDQL and the set $\widehat{\Pi}$ of all star-free concurrent regular programs are generated by simultaneous induction as follows:

$$\widehat{L} \ni A ::= p \mid \neg A \mid A \wedge A \mid [a]A,$$
$$\widehat{\Pi} \ni a ::= \mathrm{skip} \mid \mathrm{abort} \mid \pi \mid a \,;\, a \mid a \cup a \mid A? \mid a \parallel a \mid a \;⌊\!⌊\; a \mid (a \mid a),$$

where $p \in L_0$ and $\pi \in \Pi_0$.

[7] Wan Fokkink. *Introduction to process algebra*. Springer, 1999.

## Merge Operators

- The parallel composition operator $\|$ should satisfy

$$a \parallel b = (a \mathbin{⫫} b) \cup (b \mathbin{⫫} a) \cup (a \mid b).$$

- Let $c <: d$ and $c :> d$ be atomic programs, representing sending and receiving a datum $d$ via a channel $c$.
- We suppose that the result of the simultaneous execution of $c <: d$ and $c :> d$ is always skip, and that of the other atomic programs is always abort as follows:

$$\pi_1 \mid \pi_2 = \pi_2 \mid \pi_1 = \gamma(\pi_1, \pi_2) = \begin{cases} \text{skip} & (\pi_1 = c <: d \text{ and } \pi_2 = c :> d \text{ for some } c, d), \\ \text{abort} & (\text{otherwise}). \end{cases}$$

where $\gamma$ is called the communication function.

## Properties of Merge Operators

There are several other properties that the merge operators should satisfy as follows:

**Definition 4.1**

Assume that $c :> d, c <: d \in \Pi_0$ for channels $c$ and data $d$. The following conditions are imposed on the merge operators.

1. $a \parallel b = ((a \mathbin{\lfloor\!\lfloor} b) \cup (b \mathbin{\lfloor\!\lfloor} a)) \cup (a \mid b)$,
2. $\text{skip} \mathbin{\lfloor\!\lfloor} a = a$,
3. $\text{abort} \mathbin{\lfloor\!\lfloor} a = \text{abort}$,
4. $\pi \mathbin{\lfloor\!\lfloor} a = \pi \; ; a$,
5. $(\text{skip} \; ; a) \mathbin{\lfloor\!\lfloor} b = a \parallel b$,
6. $(\text{abort} \; ; a) \mathbin{\lfloor\!\lfloor} b = \text{abort}$,
7. $(\pi \; ; a) \mathbin{\lfloor\!\lfloor} b = \pi \; ; (a \parallel b)$,
8. . . . (see our paper for more details)

## How Merge Operators Handle Concurrent Behavior and Communication

### Example 2

Let us consider the program $(\pi_1 \,;\, a) \parallel (\pi_2 \,;\, b)$ as follows:

$$\begin{aligned}(\pi_1 \,;\, a) \parallel (\pi_2 \,;\, b) &= ((\pi_1 \,;\, a) \mathbin{\lfloor\!\lfloor} (\pi_2 \,;\, b)) \cup ((\pi_2 \,;\, b) \mathbin{\lfloor\!\lfloor} (\pi_1 \,;\, a)) \cup ((\pi_1 \,;\, a) \mid (\pi_2 \,;\, b)) \\ &\qquad \text{(By Definition 4.1 (1))} \\ &= (\pi_1 \,;\, (a \parallel (\pi_2 \,;\, b))) \cup (\pi_2 \,;\, (b \parallel (\pi_1 \,;\, a))) \cup ((\pi_1 \mid \pi_2) \,;\, (a \parallel b)) \\ &\qquad \text{(By Definition 4.1 (7) and (21))}\end{aligned}$$

- In Example 2, the program can choose to execute nondeterministically among $\pi_1$, $\pi_2$, and $\pi_1 \mid \pi_2$.

## CDQL - A Conservative Extension of BDQL

**Theorem 2**

CDQL is a conservative extension of BDQL: for any $M$, $s \in S$, $A \in \widehat{L}$, there exists $B \in L$ such that $(M, s) \models A$ if and only if $(M, s) \models B$.

**Proof.**

See our paper for more details. $\square$

- We can transform CDQL models into BDQL models and use the semantics of BDQL to verify BDQL models without the necessity to define the semantics of CDQL.

## Encapsulation Operator

- We introduce a unary operator $\partial_H$ called an encapsulation operator, which is used to enforce communication in programs.

### Definition 4.2

More formally, for each $H \subseteq \Pi_0$, the encapsulation operator $\partial_H$ is the unary function on $\Pi$ defined as follows:

1. $\partial_H(\pi) = \begin{cases} \text{abort} & (\pi \in H) \\ \pi & (\pi \notin H) \end{cases}$,
2. $\partial_H(\text{skip}) = \text{skip}$,
3. $\partial_H(\text{abort}) = \text{abort}$,
4. $\partial_H(a\ ;\ b) = \partial_H(a)\ ;\ \partial_H(b)$,
5. $\partial_H(a \cup b) = \partial_H(a) \cup \partial_H(b)$,
6. $\partial_H(A?) = A?$.

## Encapsulation Operator

### Example 3

Let us consider the program $\partial_H((\pi_1 ; a) \parallel (\pi_2 ; b))$ as follows:

$$
\begin{aligned}
&\partial_H((\pi_1 ; a) \parallel (\pi_2 ; b)) \\
&= \partial_H((\pi_1 ; (a \parallel (\pi_2 ; b))) \cup (\pi_2 ; (b \parallel (\pi_1 ; a))) \cup ((\pi_1 \mid \pi_2) ; (a \parallel b))) && \text{(By Example 2)} \\
&= \partial_H(\pi_1 ; (a \parallel (\pi_2 ; b))) \cup \partial_H(\pi_2 ; (b \parallel (\pi_1 ; a))) \cup \partial_H((\pi_1 \mid \pi_2) ; (a \parallel b)) && \text{(By Definition 4.2 (5))} \\
&= (\partial_H(\pi_1) ; \partial_H(a \parallel (\pi_2 ; b))) \cup (\partial_H(\pi_2) ; \partial_H(b \parallel (\pi_1 ; a))) \cup (\partial_H(\pi_1 \mid \pi_2) ; \partial_H(a \parallel b)) \\
& && \text{(By Definition 4.2 (4))}
\end{aligned}
$$

- In Example 3, if $\pi_1$ and $\pi_2$ are atomic communication programs, $\partial_H(\pi_1)$ and $\partial_H(\pi_2)$ will become **abort**; and $\partial_H(\pi_1 \mid \pi_2)$ will become either **skip** or **abort**.
- The encapsulation operator can effectively limit the number of interleavings with communication arising from concurrency under verification.

## Contents

## Standard Interpretation

- Now we discuss the verification of concrete quantum programs based on CDQL
- Fix $\Pi_0$ and $L_0$ as follows ($\mathbb{N}$ denotes natural numbers and $\mathbb{C}$ denotes complex numbers):

$$\begin{aligned} \Pi_0 = \{&\mathtt{H}(i), \mathtt{X}(i), \mathtt{Y}(i), \mathtt{Z}(i), \mathtt{CX}(i,j), \mathtt{SWAP}(i,j) : i,j \in \mathbb{N}, i \neq j\} \\ &\cup \{c <: d, c :> d : c \in C, d \in D\}, \\ L_0 = \{&p(i, |\psi\rangle), p(i, i+1, |\Psi\rangle) : i \in \mathbb{N}, |\psi\rangle \in \mathbb{C}^2, |\Psi\rangle \in \mathbb{C}^4\}, \end{aligned}$$

- Standard interpretation $\bar{v} : \Pi_0 \to \mathcal{U}(\mathbb{C}^{2^n})$ for atomic programs

$$\begin{aligned} \bar{v}(\mathtt{H}(i)) &= I^{\otimes i} \otimes H \otimes I^{\otimes n-i-1}, \quad \bar{v}(\mathtt{X}(i)) = I^{\otimes i} \otimes X \otimes I^{\otimes n-i-1}, \\ \bar{v}(\mathtt{Y}(i)) &= I^{\otimes i} \otimes Y \otimes I^{\otimes n-i-1}, \quad \bar{v}(\mathtt{Z}(i)) = I^{\otimes i} \otimes Z \otimes I^{\otimes n-i-1}, \\ \bar{v}(\mathtt{CX}(i,j)) &= I^{\otimes i} \otimes |0\rangle\langle 0| \otimes I^{\otimes n-i-1} + (I^{\otimes i} \otimes |1\rangle\langle 1| \otimes I^{\otimes n-i-1})(I^{\otimes j} \otimes X \otimes I^{\otimes n-j-1}), \\ \bar{v}(\mathtt{SWAP}(i,j)) &= \bar{v}(\mathtt{CX}(i,j)\ ;\ \mathtt{CX}(j,i)\ ;\ \mathtt{CX}(i,j)), \end{aligned}$$

where $I^{\otimes i} = \overbrace{I \otimes \cdots \otimes I}^{i}$.

## Standard Interpretation

- Standard interpretation $\overline{V} : L_0 \to \mathcal{C}(\mathbb{C}^{2^n})$ for atomic formulas

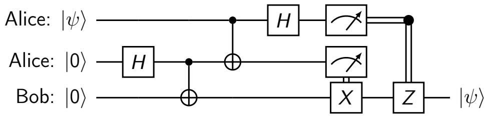
$$\overline{V}(p(i, |\psi\rangle)) = \mathbb{C}^{2^i} \otimes \mathrm{span}\{|\psi\rangle\} \otimes \mathbb{C}^{2^{n-i-1}},$$
$$\overline{V}(p(i, i+1, |\Psi\rangle)) = \mathbb{C}^{2^i} \otimes \mathrm{span}\{|\Psi\rangle\} \otimes \mathbb{C}^{2^{n-i-2}},$$

- Conditional quantum programs for quantum tests in CDQL:

$$\text{if } A \text{ then } a \text{ else } b \text{ fi} = (A? \,;\, a) \cup (\neg A? \,;\, b).$$

☞ considering binary projective measurements

# Quantum Teleportation



$$\begin{aligned}
\text{teleport} &= \texttt{H}(1) \; ; \texttt{CX}(1,2) \; ; \partial_H(\text{Alice} \parallel \text{Bob}),\\
\text{Alice} &= \texttt{CX}(0,1) \; ; \texttt{H}(0)\\
&\quad ; \text{if } p(1, |0\rangle) \text{ then } c_{12} <: 0 \text{ else } c_{12} <: 1 \text{ fi}\\
&\quad ; \text{if } p(0, |0\rangle) \text{ then } c_{02} <: 0 \text{ else } c_{02} <: 1 \text{ fi},\\
\text{Bob} &= ((c_{12} :> 1 \; ; \texttt{X}(2)) \cup c_{12} :> 0)\\
&\quad ; ((c_{02} :> 1 \; ; \texttt{Z}(2)) \cup c_{02} :> 0).
\end{aligned}$$

$$\text{where } H = \{c_i <: 0, c_i <: 1, c_i :> 0, c_i :> 1 : i \in \{12, 02\}\}.$$

- The desired property of Quantum Teleportation is that "an arbitrary pure state $|\psi\rangle$ is correctly teleported." In CDQL, this property is expressed as:

$$(\overline{M}_3, |\psi\rangle \otimes |0\rangle \otimes |0\rangle) \models [\text{teleport}]p(2, |\psi\rangle).$$

## Implementation of CDQL

- We extend our support tool for BDQL[8] to make a new support tool for CDQL to handle both BDQL and CDQL models as a one-stop place.
- We use Maude, a high-level specification/programming language based on rewriting logic, for our tool development.

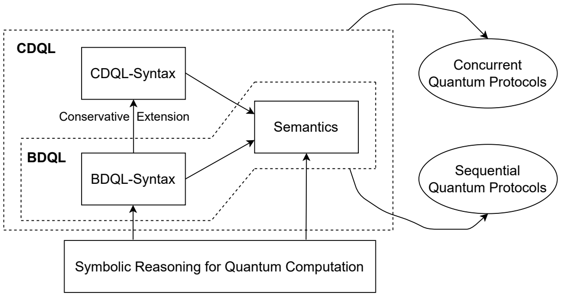

CDQL
CDQL-Syntax
Conservative Extension
BDQL
BDQL-Syntax
Semantics
Concurrent Quantum Protocols
Sequential Quantum Protocols
Symbolic Reasoning for Quantum Computation

Figure: The architecture of the implementation of CDQL

[8] Takagi, Do, and Ogata, "Automated Quantum Program Verification in a Dynamic Quantum Logic".

## Handling Large Interleavings from Concurrency in CDQL

- We propose a Lazy Rewriting Strategy to effectively handle large interleavings from concurrency in CDQL:
  - Control the transformation process for interleaving enumeration and the verification process for each interleaving step by step to detect unnecessary interleavings early.
  - Memorize the previous results of equational simplification for subprograms and reuse the results to prevent duplicated work if applicable.
- To make it more intuitive, let us recall the simple program in Example 3 as follows:

$$\begin{aligned}&\partial_H((\pi_1 \,;\, a) \parallel (\pi_2 \,;\, b))\\ &= (\partial_H(\pi_1) \,;\, \partial_H(a \parallel (\pi_2 \,;\, b))) \cup (\partial_H(\pi_2) \,;\, \partial_H(b \parallel (\pi_1 \,;\, a))) \cup (\partial_H(\pi_1 \mid \pi_2) \,;\, \partial_H(a \parallel b))\end{aligned}$$

(By Example 3)

☞ if $\partial_H(\pi_1)$, $\partial_H(\pi_2)$, or $\partial_H(\pi_1|\pi_2)$ is reduced to **abort**, then its subprogram should not be handled or rewritten anymore.

☞ In the case of no **abort** for $\partial_H(\pi_1)$, $\partial_H(\pi_2)$, or $\partial_H(\pi_1|\pi_2)$, the subprogram $\partial_H(a \parallel b)$ are more likely to appear multiple times for the subsequent expansions.

## A Support Tool and Experiment Results

- A support tool for CDQL is extended from our previous support tool for BDQL[9] to handle concurrent behavior and communication among participants in quantum protocols.
- The implementation is available at `https://github.com/canhminhdo/DQL`

| | | BDQL Models | | CDQL Models | |
|---|---|---|---|---|---|
| **Protocol** | **Qubits** | **Rewrite Steps** | **Time** | **Rewrite Steps** | **Time** |
| Quantum Teleportation | 3 | 2,558 | 1ms | 3,431 | 2ms |
| Entanglement Swapping | 4 | 4,134 | 2ms | 10,196 | 5ms |
| Quantum Secret Sharing | 4 | 6,665 | 3ms | 32,592 | 20ms |
| Quantum Relay Scheme | 5 | 13,060 | 6ms | 19,304 | 12ms |
| Bidirectional Quantum Teleportation | 6 | 14,683 | 6ms | 25,668 | 14ms |
| Two-qubit Quantum Teleportation | 6 | 52,161 | 35ms | 65,828 | 45ms |
| Quantum Gate Teleportation | 6 | 56,873 | 41ms | 224,898 | 144ms |

[9] Takagi, Do, and Ogata, "Automated Quantum Program Verification in a Dynamic Quantum Logic".

## The Power of the Lazy Rewriting Strategy for CDQL Models

- A basic version of our support tool does not use the strategy.
- A moderate version of our support tool partially uses the strategy without memorization.
- An advanced version of our support tool fully uses the strategy.

| | | Basic | | Moderate | | Advanced | |
|---|---|---|---|---|---|---|---|
| Protocol | Qubits | Rewrite Steps | Time | Rewrite Steps | Time | Rewrite Steps | Time |
| Quantum Teleportation | 3 | 69,507 | 16ms | 3,761 | 2ms | 3,431 | 2ms |
| Entanglement Swapping | 4 | 361,465 | 107ms | 48,513 | 20ms | 10,196 | 5ms |
| Quantum Secret Sharing | 4 | 634,699,523 | 17h 39m 56s | 285,482 | 110ms | 32,592 | 20ms |
| Quantum Relay Scheme | 5 | N/A | N/A | 29,134 | 12ms | 19,304 | 12ms |
| Bidirectional Quantum Teleportation | 6 | 1,763,198,792 | 5d 17h 3m | 292,980 | 117ms | 25,668 | 14ms |
| Two-qubit Quantum Teleportation | 6 | 948,116,386 | 1d 13h 47m | 117,790 | 78ms | 65,828 | 45ms |
| Quantum Gate Teleportation | 6 | N/A | N/A | 13,661,654 | 9,775ms | 224,898 | 144ms |

☞ N/A denotes that the experiments could not be completed within one week.

## Contents

## Conclusions and Future Work

- We have presented Concurrent Dynamic Quantum Logic (CDQL), a conservative extension of Basic Dynamic Quantum Logic (BDQL), to formalize and verify quantum protocols, where concurrent behavior and communication are considered.
- We have presented a new support tool for CDQL which is based on the support tool for BDQL, where both BDQL and CDQL models can be handled.
- We have presented the lazy rewriting strategy to handle large interleavings arising from concurrency in CDQL.
- We consider several lines of future work as follows:
  - Extend both BDQL and CDQL to support probabilistics in their formalizations.
  - Enhance our logic to support auxiliary variables and some specific data structures like arrays or queues for formalizing asynchronous messages in CDQL.
  - Handle iteration to describe infinite behaviors.

Thank You!