Automated Quantum Protocol Verification with Concurrent Dynamic Quantum Logic in Maude

Canh Minh Do

Japan Advanced Institute of Science and Technology (JAIST)

canhdo@jaist.ac.jp

The 16th International Workshop on Rewriting Logic and Its Applications (WRLA 2026)
Torino, Italy - April 12, 2026

Contents

Contents

Introduction

- Building large-scale quantum computers capable of running advanced algorithms, such as Shor's algorithm[1] for effectively solving the factoring and discrete logarithm problems, is becoming increasingly feasible.
- Despite its promise, quantum computing is different from classical computing because it relies on principles of quantum mechanics that are often counterintuitive, making it challenging to design and implement quantum systems accurately.
- Moreover, traditional testing techniques are much less effective for quantum systems due to the nondeterminism and probabilistic outcomes of quantum measurement.
- 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 modeling 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 $\mathcal{H}$ serves as the state space of a quantum system that is a complete complex vector space equipped with an inner product.
- A qubit is a quantum system whose state space is the two-dimensional Hilbert space $\mathcal{H}_2 = \mathbb{C}^2$.
- A qubit $|\psi\rangle$ can be expressed as a superposition of $|0\rangle$ and $|1\rangle$ as follows:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = (\alpha, \beta)^T$$

  where $|0\rangle = (1,0)^T$, $|1\rangle = (0,1)^T$, and $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$.
- The set $\{|0\rangle, |1\rangle\}$ forms an orthonormal basis of $\mathcal{H}_2$ and is called the computational basis.
- For multiple qubits, the tensor product $\mathcal{H}_A \otimes \mathcal{H}_B$ of Hilbert spaces $\mathcal{H}_A$ and $\mathcal{H}_B$ is defined as a vector space consisting of linear combinations of the product states

$$|\psi_1\psi_2\rangle = |\psi_1\rangle\,|\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$$

  where $|\psi_1\rangle \in \mathcal{H}_A$ and $|\psi_2\rangle \in \mathcal{H}_B$.
- Any state in $\mathcal{H}_A \otimes \mathcal{H}_B$ cannot be written as a product state called an entangled state.

Unitary Transformation

- A unitary transformation on a Hilbert space $\mathcal{H}$ is a linear operator $U : \mathcal{H} \to \mathcal{H}$ satisfying $UU^\dagger = U^\dagger U = I_{\mathcal{H}}$, which deterministically changes a state $|\psi\rangle$ to $U|\psi\rangle$.
- 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:

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

- For example, we have

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = |1\rangle, \qquad H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$

$$X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = |0\rangle, \qquad H|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Unitary Transformation (cont.)

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

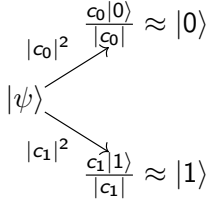
$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

- For example, we have

$$\begin{aligned} CX\,|00\rangle &= |00\rangle, & SWAP\,|00\rangle &= |00\rangle, \\ CX\,|01\rangle &= |01\rangle, & SWAP\,|01\rangle &= |10\rangle, \\ CX\,|10\rangle &= |11\rangle, & SWAP\,|10\rangle &= |01\rangle, \\ CX\,|11\rangle &= |10\rangle, & SWAP\,|11\rangle &= |11\rangle. \end{aligned}$$

Measurement

- A quantum measurement on a quantum system in the Hilbert space $\mathcal{H}$ is a collection of measurement operators $\{M_m\}$ satisfying the completeness relation $\sum_m M_m^\dagger M_m = I$.
- Measuring a state $|\psi\rangle$ yields outcome m with probability $p(m) = \| M_m |\psi\rangle \|^2$, and the state is changed to $|\psi_m\rangle = \frac{M_m|\psi\rangle}{\sqrt{p(m)}}$ nondeterministically.
- After executing the measurement $\{M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|\}$, a qubit $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$ is collapsed into either $\frac{M_0|\psi\rangle}{|c_0|}$ with probability $|c_0|^2$ or into $\frac{M_1|\psi\rangle}{|c_1|}$ with probability $|c_1|^2$.



Projection

- For each closed subspace V of $\mathcal{H}$, there exists a unique projection operator $\mathcal{P}_V$.
- Every state $|\psi\rangle \in \mathcal{H}$ can be written as

$$|\psi\rangle = |\psi_V\rangle + |\psi_{V^\perp}\rangle$$

  with $|\psi_V\rangle \in V$ and $|\psi_{V^\perp}\rangle \in V^\perp$, the orthogonal complement of V.
- The projection $\mathcal{P}_V : \mathcal{H} \to V$ is defined by $\mathcal{P}_V |\psi\rangle = |\psi_V\rangle$.
- If a state $|\psi\rangle$ is in the closed subspace V of the projection operator $\mathcal{P}_V$, then

$$\mathcal{P}_V |\psi\rangle = |\psi\rangle .$$

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 $A_1 \to a_1 | \ldots | A_n \to a_n$ 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 Π 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 π 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 Π^- 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 Π_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. $[\![[a]A]\!]^M = \{s \in \mathcal{H} : (s,t) \in R_a^M \text{ implies } t \in [\![A]\!]^M \text{ for any } t \in \mathcal{H}\}$;
5. $R_{\mathsf{skip}}^M = \{(s,t) : s = t\}$;
6. $R_{\mathsf{abort}}^M = \emptyset$;
7. $R_\pi^M = \{(s,t) : (v(\pi))(s) = t\}$;
8. $R_{a;b}^M = \{(s,t) : (s,u) \in R_a^M \text{ and } (u,t) \in R_b^M \text{ for some } u \in \mathcal{H}\}$;
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

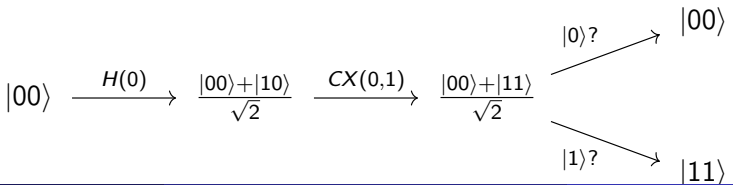
Kripke frames can be constructed from the family $\{R_a^M : a \in \Pi^-\}$ of relations on $\mathcal{H}$.

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

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

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

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



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.

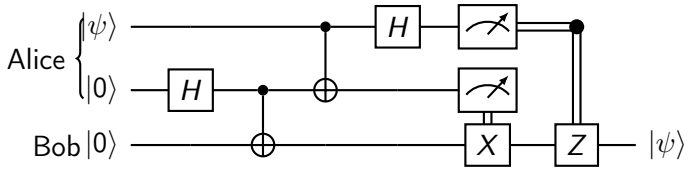


Figure: Quantum Teleportation

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)[6] to express concurrent behavior and communication. Similar to ACP, we define $\|$ using the left merge operator $\lfloor\!\lfloor$ 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:

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

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

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

Merge Operators

- The parallel composition operator $\|$ should satisfy

$$a \parallel b = (a \between b) \cup (b \between a) \cup (a \mid b).$$

- 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 γ 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. $a \parallel b = ((a \between b) \cup (b \between a)) \cup (a \mid b)$,
2. $\text{skip} \between a = a$,
3. $\text{abort} \between a = \text{abort}$,
4. $\pi \between a = \pi \,;\, a$,
5. $(\text{skip} \,;\, a) \between b = a \parallel b$,
6. $(\text{abort} \,;\, a) \between b = \text{abort}$,
7. $(\pi \,;\, a) \between b = \pi \,;\, (a \parallel b)$,
8. ...

Example 2

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

$$(\pi_1 \,;\, a) \parallel (\pi_2 \,;\, b) = ((\pi_1 \,;\, a) \between (\pi_2 \,;\, b)) \cup ((\pi_2 \,;\, b) \between (\pi_1 \,;\, a)) \cup ((\pi_1 \,;\, a) \mid (\pi_2 \,;\, b))$$

(By Definition 4.1 (1))

$$= (\pi_1 \,;\, (a \parallel (\pi_2 \,;\, b))) \cup (\pi_2 \,;\, (b \parallel (\pi_1 \,;\, a))) \cup ((\pi_1 \mid \pi_2) \,;\, (a \parallel b))$$

(By Definition 4.1 (7) and (21))

- In Example 2, the program can choose to execute nondeterministically among π_1, π_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 $(M, s) \models B$.

- 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 ∂_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 ∂_H is the unary function on Π 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 π_1 and π_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 Π_0 and L_0 as follows ($\mathbb{N}$ denotes natural numbers and $\mathbb{C}$ denotes complex numbers):

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

- 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

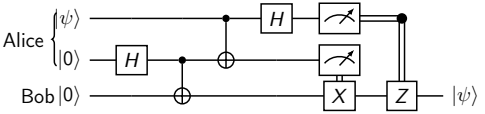
$$\overline{V}(p(i, |\psi\rangle)) = \mathbb{C}^{2^i} \otimes \mathrm{span}\{|\psi\rangle\} \otimes \mathbb{C}^{2^{n-i-1}},$$
$$\overline{V}(p(i, i+1, |\Psi\rangle)) = \mathbb{C}^{2^i} \otimes \mathrm{span}\{|\Psi\rangle\} \otimes \mathbb{C}^{2^{n-i-2}},$$

- 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



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

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 implemented in Maude for BDQL[7] to make a new support tool for CDQL to handle both BDQL and CDQL models.

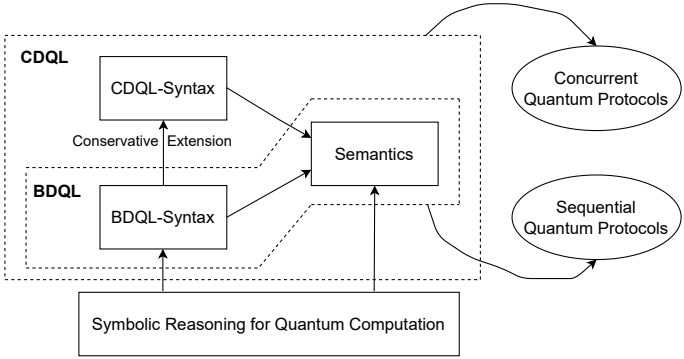


Figure: The architecture of the implementation of CDQL

[7] 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[8] to handle concurrent behavior and communication among participants in quantum protocols.
- The implementation is available at `https://github.com/canhminhdo/cdql`

Protocol	Qubits	BDQL Models		CDQL Models	
		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

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

Power of the Lazy Rewriting Strategy for CDQL Models

- The experiments were conducted on a MacPro computer that carries a 2.5 GHz microprocessor with 28 cores and 768 GB of RAM.

Protocol	Qubits	Basic		Moderate (lazy)		Advanced (lazy + memo)	
		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 Teleport.	6	1,763,198,792	5d 17h 3m	292,980	117ms	25,668	14ms
Two-qubit Quantum Teleport.	6	948,116,386	1d 13h 47m	117,790	78ms	65,828	45ms
Quantum Gate Teleport.	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

Conclusion 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 implemented in Maude, 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 probabilities 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!