Symbolic Model Checking Quantum Circuits With Density Operators in Maude

Canh Minh Do*, Kazuhiro Ogata

Japan Advanced Institute of Science and Technology, 1-8 Asahidai, Nomi, Japan

Abstract

We proposed a symbolic approach to model checking quantum circuits using a set of laws from quantum mechanics and basic matrix operations with Dirac notation and used Maude, a high-level specification/programming language based on rewriting logic, to implement our symbolic approach. However, it only supports the formalization of pure states but not mixed states of quantum systems. This paper extends our current symbolic approach to deal with mixed states of quantum systems by using density operators for their representations. Once a quantum circuit is formalized by our proposed way with an initial state and a desired property expressed in Linear Temporal Logic (LTL), we use a built-in Maude LTL model checker to verify whether the quantum circuit enjoys the desired property from the initial state automatically. As case studies, we successfully verify several quantum communication protocols: Superdense Coding, Quantum Teleportation, Quantum Secret Sharing, and Entanglement Swapping.

Keywords

Mixed States, Density Operators, Quantum circuits, Dirac notation, Symbolic Model Checking, Maude

1. Introduction

Quantum computing represents a new frontier in computation, offering the potential to solve hard problems, which were previously considered intractable for the current computing technologies. For example, Shore's fast algorithm [1] can break the security of modern cryptographic systems relying on discrete logarithms and factoring in the future once practical quantum computers are available. To design and implement such quantum algorithms, quantum circuits are often used as a model of quantum computation. Unlike classical circuits, quantum circuits manipulate qubits using quantum operations (e.g., quantum gates). Because quantum computation is counter-intuitive and distinct from classical computing due to different principles, such as superposition, entanglement, and measurement, it is challenging to design and implement quantum algorithms (or quantum circuits) accurately. Therefore, it is crucial to verify that quantum circuits enjoy some desired properties in preparing for the quantum era.

Our research group proposed a symbolic approach [2] to model checking quantum circuits using a set of laws from quantum mechanics and basic matrix operations with Dirac notation [3] and used Maude [4], a high-level specification/programming language based on rewriting

The 2nd International Workshop on Formal Analysis and Verification of Post-Quantum Cryptographic Protocols, November 11, 2023, Brisbane, Australia

*Corresponding author.

canhdo@jaist.ac.jp (C. M. Do); ogata@jaist.ac.jp (K. Ogata)
0000-0002-1601-4584 (C. M. Do); 0000-0002-4441-3259 (K. Ogata)

logic [5], to implement our symbolic approach. However, it only supports the formalization of pure states but not mixed states of quantum systems. In many practical situations, a quantum system is not a single, well-defined state but a statistical mixture of multiple pure states. Therefore, it requires the formalization of mixed states to describe the probabilities associated with each pure state. This paper extends our current symbolic approach [2] to deal with mixed states of quantum systems by using density operators for their representations. Once a quantum circuit is formalized by our proposed way with an initial state and a desired property expressed in Linear Temporal Logic (LTL), we use the built-in Maude LTL model checker to automatically verify whether the quantum circuit enjoys the desired property from the initial state. As case studies, we successfully verify several quantum communication protocols: Superdense Coding [6], Quantum Teleportation [7], Quantum Secret Sharing [8], and Entanglement Swapping [9]. This demonstrates the usefulness of our symbolic model checking quantum circuits with density operators in Maude. The support tool and case studies are publicly available at https://github.com/canhminhdo/QTC-Maude under the `mixed-states` folder.

The rest of the paper is organized as follows: Sect. 2 provides basic quantum mechanics, symbolic reasoning for quantum computation based on Dirac notation, and Kripke structure; Sect. 3 describes the formalization of qubits, quantum gates, and quantum circuits; Sect. 4 demonstrates how to use our approach to model checking for Quantum Teleportation; Sect. 5 shows the experimental results with our support tool; Sect. 6 presents some existing work; and Sect. 7 concludes the paper with some pieces of future work.

2. Preliminaries

Firstly, we describe some basic notations from quantum mechanics based on linear algebra (refer to [10] for more details). Secondly, we describe symbolic reasoning [2, 11] to reason about quantum computation based on Dirac notation. Lastly, we describe Kripke structures to formalize quantum systems.

2.1. Basic Quantum Mechanics

In classical computing, the fundamental unit of information is a bit whose value is either 0 or 1. In quantum computing, the counterpart is a *quantum bit* or *qubit*, which has two basis states, conventionally written in Dirac notation [3] as $|\mathbf{0}\rangle$ and $|\mathbf{1}\rangle$, corresponding to one-bit classical values, whose values are two column vectors $\begin{pmatrix}1\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\end{pmatrix}$, respectively. In quantum theory, a general state of a quantum system is a superposition or linear combination of basis states. A single qubit has state $|\psi\rangle = \alpha\,|\mathbf{0}\rangle + \beta\,|\mathbf{1}\rangle$, where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. States can be represented by column complex vectors as follows: $|\psi\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix} = \alpha\,|\mathbf{0}\rangle + \beta\,|\mathbf{1}\rangle$, where $\{|\mathbf{0}\rangle, |\mathbf{1}\rangle\}$ forms an orthonormal basis of the 2D complex vector space. Formally, a quantum state is a unit vector in a Hilbert space $\mathcal{H}$, which is equipped with an inner product satisfying some axioms.

The basis $\{|\mathbf{0}\rangle, |\mathbf{1}\rangle\}$ is called as the *standard* basis. Besides, we have some other bases of interest, such as *diagonal* (or *dual*, or *Hadamard*) basis consisting of the following vectors:

$$|+\rangle = \tfrac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle) \text{ and } |-\rangle = \tfrac{1}{\sqrt{2}}(|\mathbf{0}\rangle - |\mathbf{1}\rangle)$$

The evolution of a closed quantum system can be performed by a unitary transformation. If the state of a qubit is represented by a column vector, then a unitary transformation $\boldsymbol{U}$ can be represented by a complex-value matrix such that $\boldsymbol{U}\boldsymbol{U}^\dagger = \boldsymbol{U}^\dagger\boldsymbol{U} = \boldsymbol{I}$ or $\boldsymbol{U}^\dagger = \boldsymbol{U}^{-1}$, where $\boldsymbol{U}^\dagger$ is the conjugate transpose of $\boldsymbol{U}$. The trace of $\boldsymbol{U}$ is defined as $tr(\boldsymbol{U}) = \sum_i \langle\phi_i|\boldsymbol{U}|\phi_i\rangle$ for some given orthonormal basis $\{|\phi_i\rangle\}$ of $\mathcal{H}$. $\boldsymbol{U}$ acts on the Hilbert space $\mathcal{H}$ transforming a state $|\psi\rangle$ to a state $|\psi'\rangle$ by a matrix multiplication such that $|\psi'\rangle = \boldsymbol{U}|\psi\rangle$. There are some common quantum gates: the Hadamard gate $\boldsymbol{H}$, the identity gate $\boldsymbol{I}$, the Pauli gates $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$, and the controlled-NOT gate $\boldsymbol{CX}$. Note that the $\boldsymbol{CX}$ gate performs on two qubits, while the remaining gates perform on a single qubit. Their matrix representations are as follows:

$$\boldsymbol{I}_2 = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}, \qquad \boldsymbol{X} = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}, \qquad \boldsymbol{Y} = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix},$$

$$\boldsymbol{Z} = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}, \qquad \boldsymbol{H} = \tfrac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \qquad \boldsymbol{CX} = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}.$$

where i is the imaginary unit. For example, the Hadamard gate on a single qubit performs the mapping $|\mathbf{0}\rangle \mapsto \tfrac{1}{\sqrt{2}}(|\mathbf{0}\rangle + |\mathbf{1}\rangle)$ and $|\mathbf{1}\rangle \mapsto \tfrac{1}{\sqrt{2}}(|\mathbf{0}\rangle - |\mathbf{1}\rangle)$. The controlled-NOT gate on pairs of qubits performs the mapping $|\mathbf{00}\rangle \mapsto |\mathbf{00}\rangle, |\mathbf{01}\rangle \mapsto |\mathbf{01}\rangle, |\mathbf{10}\rangle \mapsto |\mathbf{11}\rangle, |\mathbf{11}\rangle \mapsto |\mathbf{10}\rangle$, which can be understood as inverting the second qubit (referred to as the *target*) if and only if the first qubit (referred to as the *control*) is one.

A quantum measurement is described as a collection $\{\boldsymbol{M}_m\}$ of measurement operators, where the indices m refer to the measurement outcomes. It is required that the measurement operators satisfy $\sum_m \boldsymbol{M}_m^\dagger \boldsymbol{M}_m = \boldsymbol{I}_{\mathcal{H}}$. If the state of a quantum system is $|\psi\rangle$ before the measurement, then the probability for the result m is as follows:

$$p(m) = \langle\psi|\, \boldsymbol{M}_m^\dagger \boldsymbol{M}_m \,|\psi\rangle,$$

and the state of the quantum system after the measurement is $\frac{\boldsymbol{M}_m|\psi\rangle}{\sqrt{p(m)}}$ provided that $p(m) > 0$. For example, if a qubit is in state $\alpha|\mathbf{0}\rangle + \beta|\mathbf{1}\rangle$ and measuring with $\{\boldsymbol{M}_0, \boldsymbol{M}_1\}$ operators, we have the result 0 with probability $|\alpha|^2$ at the post-measurement state $|\mathbf{0}\rangle$ and the result 1 with probability $|\beta|^2$ at the post-measurement state $|\mathbf{1}\rangle$, where $\boldsymbol{M}_0 = |\mathbf{0}\rangle \times \langle\mathbf{0}|$ and $\boldsymbol{M}_1 = |\mathbf{1}\rangle \times \langle\mathbf{1}|$.

For multiple qubits, we use the tensor product of Hilbert spaces. Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two Hilbert spaces. Their tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$ is defined as a vector space consisting of linear combinations of the vectors $|\psi_1\psi_2\rangle = |\psi_1\rangle|\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, where $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$. Systems of two or more qubits may be in *entangled* states, meaning that states of qubits are correlated and inseparable. Entanglement shows that an entangled state of two qubits cannot be expressed as a tensor product of single-qubit states. We can use $\boldsymbol{H}$ and $\boldsymbol{CX}$ gates to create entangled states as follows: $\boldsymbol{CX}((\boldsymbol{H} \otimes \boldsymbol{I})\,|\mathbf{00}\rangle) = \frac{1}{\sqrt{2}}(|\mathbf{00}\rangle + |\mathbf{11}\rangle)$.

A pure quantum state can be represented by a ket vector $|\psi\rangle$, while a mixed state represents the probabilistic mixtures of pure states and can be described by a density operator. Given $\{(p_i, |\psi_i\rangle)\}$ be an ensemble of pure states $|\psi_i\rangle$, where $p_i \geq 0$ and $\sum_i p_i = 1$, $\rho = \sum_i p_i\,|\psi_i\rangle\langle\psi_i|$ is the density operator representing the mixed state. $\boldsymbol{U}$ transforms a mixed state ρ to a mixed state ρ' by a matrix multiplication such that $\rho' = \boldsymbol{U}\rho\boldsymbol{U}^\dagger$. For the quantum measurement $\{\boldsymbol{M}_m\}$ above, if the mixed state of a quantum system is ρ before the measurement, then the probability for the result m is as follows:

$$p(m) = tr(\boldsymbol{M}_m^\dagger \boldsymbol{M}_m \rho),$$

and the mixed state after the measurement is $\frac{\boldsymbol{M}_m \rho \boldsymbol{M}_m^\dagger}{p(m)}$ provided that $p(m) > 0$.

2.2. Symbolic Reasoning

We proposed symbolic reasoning [2, 11] based on Dirac notation with scalars using a set of laws from quantum mechanics and basic matrix operations for reasoning about quantum computation and developed a support tool in Maude to automate the reasoning. This section briefly describes terms used in our symbolic reasoning and laws used to reduce terms. The reader is recommended to refer to [2] for more details.

2.2.1. Terms

Terms are built from scalars and basis vectors with some constructors.

- Scalars are complex numbers. We extend rational numbers supported in Maude to deal with complex numbers. Some constructors for scalars, such as multiplication, fraction, addition, conjugation, absolute, power, and square root are formalized, but we do not mention them here to make the paper concise.
- Basis vectors are the standard basis written in Dirac notation as $|\mathbf{0}\rangle$ and $|\mathbf{1}\rangle$.
- Constructors for matrices consist of scalar multiplication of matrices $\cdot$, matrix product $\times$, matrix addition $+$, tensor product $\otimes$, and the conjugate transpose $\boldsymbol{A}^\dagger$ of a matrix $\boldsymbol{A}$.

In Dirac notation, $\langle\mathbf{0}|$ is the dual of $|\mathbf{0}\rangle$ such that $\langle\mathbf{0}|^\dagger = |\mathbf{0}\rangle$ and $|\mathbf{0}\rangle^\dagger = \langle\mathbf{0}|$; similarly for $\langle\mathbf{1}|$. The terms $|j\rangle \times \langle k|$ and the inner product of $|j\rangle$ and $|k\rangle$ may be written shortly as $|j\rangle\,\langle k|$ and $\langle j|k\rangle$ for any $j, k \in \{0, 1\}$. By using these notations with the laws below, we can intuitively explain how quantum operations work.

2.2.2. Laws

Table 1 presents a set of laws derived from the properties of quantum mechanics and basic matrix operations. Because $|\mathbf{0}\rangle$ and $|\mathbf{1}\rangle$ can be viewed as 2×1 matrices, then the laws actually describe matrix calculations with Dirac notation, zero and identity matrices, and scalars. These

Table 1
A set of laws used for symbolic reasoning

No.	Law
L1	$\langle\mathbf{0}\|\mathbf{0}\rangle = \langle\mathbf{1}\|\mathbf{1}\rangle = 1, \langle\mathbf{1}\|\mathbf{1}\rangle = \langle\mathbf{0}\|\mathbf{1}\rangle = 0$
L2	Associativity of $\times, +, \otimes$ and Commutativity of $+$
L3	$0 \cdot \boldsymbol{A}_{m\times n} = \mathbf{O}_{m\times n},\ c \cdot \mathbf{O} = \mathbf{O},\ 1 \cdot \boldsymbol{A} = \boldsymbol{A}$
L4	$c \cdot (\boldsymbol{A} + \boldsymbol{B}) = c \cdot \boldsymbol{A} + c \cdot \boldsymbol{B}$
L5	$c_1 \cdot \boldsymbol{A} + c_2 \cdot \boldsymbol{A} = (c_1 + c_2) \cdot \boldsymbol{A}$
L6	$c_1 \cdot (c_2 \cdot \boldsymbol{A}) = (c_1 \cdot c_2) \cdot \boldsymbol{A}$
L7	$(c_1 \cdot \boldsymbol{A}) \times (c_2 \cdot \boldsymbol{B}) = (c_1 \cdot c_2) \cdot (\boldsymbol{A} \times \boldsymbol{B})$
L8	$\boldsymbol{A} \times (c \cdot \boldsymbol{B}) = (c \cdot \boldsymbol{A}) \times \boldsymbol{B} = c \cdot (\boldsymbol{A} \times \boldsymbol{B})$
L9	$\boldsymbol{A} \otimes (c \cdot \boldsymbol{B}) = (c \cdot \boldsymbol{A}) \otimes \boldsymbol{B} = c \cdot (\boldsymbol{A} \otimes \boldsymbol{B})$
L10	$\mathbf{O}_{m\times n} \times \boldsymbol{A}_{n\times p} = \boldsymbol{A}_{m\times n} \times \mathbf{O}_{n\times p} = \mathbf{O}_{m\times p}$
L11	$\boldsymbol{I}_m \times \boldsymbol{A}_{m\times n} = \boldsymbol{A}_{m\times n} \times \boldsymbol{I}_n = \boldsymbol{A}_{m\times n}$
L12	$\boldsymbol{A} + \mathbf{O} = \mathbf{O} + \boldsymbol{A} = \mathbf{O}$
L13	$\mathbf{O}_{m\times n} \otimes \boldsymbol{A}_{p\times q} = \boldsymbol{A}_{p\times q} \otimes \mathbf{O}_{m\times n} = \mathbf{O}_{mp\times nq}$
L14	$\boldsymbol{A} \times (\boldsymbol{B} + \boldsymbol{C}) = \boldsymbol{A} \times \boldsymbol{B} + \boldsymbol{A} \times \boldsymbol{C}$
L15	$(\boldsymbol{A} + \boldsymbol{B}) \times \boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{C} + \boldsymbol{B} \times \boldsymbol{C}$
L16	$(\boldsymbol{A} \otimes \boldsymbol{B}) \times (\boldsymbol{C} \otimes \boldsymbol{D}) = (\boldsymbol{A} \times \boldsymbol{C}) \otimes (\boldsymbol{B} \times \boldsymbol{D})$
L17	$\boldsymbol{A} \otimes (\boldsymbol{B} + \boldsymbol{C}) = \boldsymbol{A} \otimes \boldsymbol{B} + \boldsymbol{A} \otimes \boldsymbol{C}$
L18	$(\boldsymbol{A} + \boldsymbol{B}) \otimes \boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{C} + \boldsymbol{B} \otimes \boldsymbol{C}$
L19	$(c \cdot \boldsymbol{A})^\dagger = c^* \cdot \boldsymbol{A}^\dagger,\ (\boldsymbol{A} \times \boldsymbol{B})^\dagger = \boldsymbol{B}^\dagger \times \boldsymbol{A}^\dagger$
L20	$(\boldsymbol{A} + \boldsymbol{B})^\dagger = \boldsymbol{A}^\dagger + \boldsymbol{B}^\dagger,\ (\boldsymbol{A} \otimes \boldsymbol{B})^\dagger = \boldsymbol{A}^\dagger \otimes \boldsymbol{B}^\dagger$
L21	$\boldsymbol{I_m}^\dagger = \boldsymbol{I}_m, \boldsymbol{O}^\dagger_{m\times n} = \boldsymbol{O}_{n\times m}, (\boldsymbol{A}^\dagger)^\dagger = \boldsymbol{A}$
L22	$\|\mathbf{0}\rangle^\dagger = \langle\mathbf{0}\|,\ \langle\mathbf{0}\|^\dagger = \|\mathbf{0}\rangle,\ \|\mathbf{1}\rangle^\dagger = \langle\mathbf{1}\|,\ \langle\mathbf{1}\|^\dagger = \|\mathbf{1}\rangle$

laws are described by equations in Maude and are used to automatically reduce terms until no more matrix operation is applicable. Some laws dedicated to simplifying the expressions about complex numbers are also formalized in Maude by means of equations.

2.3. Kripke Structures

A Kripke structure K is $\langle S, I, T, A, L\rangle$ [12], where S is a set of states, $I \subseteq S$ is the set of initial states, $T \subseteq S \times S$ is a left-total binary relation over S, A is a set of atomic propositions and L is a labeling function whose type is $S \to 2^A$. Each element $(s, s') \in T$ is called a state transition from s to s' and T may be called the state transitions (with respect to K). For a state $s \in S$, $L(s)$ is the set of atomic propositions that hold in s. A path π is an infinite sequence $s_0, \ldots, s_i, s_{i+1}, \ldots$ such that $s_i \in S$ and $(s_i, s_{i+1}) \in T$ for each i. We use the following notations for paths: $\pi^i \triangleq s_i, s_{i+1}, \ldots$, $\pi_i \triangleq s_0, \ldots, s_i, s_i, s_i, \ldots$, $\pi(i) \triangleq s_i$, where $\triangleq$ is used as "be defined as." π^i is obtained by deleting the first i states $s_0, s_1, \ldots, s_{i-1}$ from π. π_i is obtained by taking the first $i+1$ states $s_0, s_1, \ldots, s_{i-1}, s_i$ and adding s_i unboundedly many times at the end. $\pi(i)$ is the ith state s_i. Let $\mathcal{P}$ be the set of all paths. π is called a computation if $\pi(0) \in I$. Let $\mathcal{C}$ be the set of all computations.

The syntax of a formula φ in LTL for K is as follows:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\ \varphi \mid \varphi\,\mathcal{U}\,\varphi$$

where $p \in A$. Let $\mathcal{F}$ be the set of all formulas in LTL for K. An arbitrary path $\pi \in \mathcal{P}$ of K and an arbitrary LTL formula $\varphi \in \mathcal{F}$ of K, $K, \pi \models \varphi$ is inductively defined as follows:

- $K, \pi \models \top$
- $K, \pi \models p$ iff $p \in L(\pi(0))$
- $K, \pi \models \neg\varphi_1$ iff $K, \pi \not\models \varphi_1$
- $K, \pi \models \varphi_1 \wedge \varphi_2$ iff $K, \pi \models \varphi_1$ and $K, \pi \models \varphi_2$
- $K, \pi \models \bigcirc\varphi_1$ iff $K, \pi^1 \models \varphi_1$
- $K, \pi \models \varphi_1\,\mathcal{U}\,\varphi_2$ iff there exists a natural number i such that $K, \pi^i \models \varphi_2$ and for all natural numbers $j < i$, $K, \pi^j \models \varphi_1$

where φ_1 and φ_2 are LTL formulas. Then, $K \models \varphi$ iff $K, \pi \models \varphi$ for each computation $\pi \in \mathcal{C}$ of K. $\bigcirc$ and $\mathcal{U}$ are called the next temporal connective and the until temporal connective, respectively. We define $\Diamond\,\varphi \triangleq \top\,\mathcal{U}\,\varphi$, where $\Diamond$ is called the eventual temporal connective. In this paper, we use LTL formulas to express desired properties under verification for our case studies.

In this paper, a state is expressed as a braced associative-commutative (AC) collection of name-value pairs, where a name may have parameters. The order of elements is not relevant in AC collections, such as sets. AC collections are called soups, and name-value pairs are called observable components. That is, a state is expressed as a braced soup of observable components. The juxtaposition operator is used as the constructor of soups. Let oc_1, oc_2, oc_3 be observable components, and then $oc_1\ oc_2\ oc_3$ is the soup of those three observable components. Since the order is irrelevant because of AC, $oc_1\ oc_2\ oc_3$ is the same as some others, such as $oc_3\ oc_2\ oc_1$. A state is expressed as $\{oc_1\ oc_2\ oc_3\}$. In this paper, rewrite rules are used to specify state transitions. Concretely, we use Maude [4], a programming/specification language based on rewriting logic. Maude makes it possible to specify complex systems flexibly and is also equipped with a built-in LTL model checker to conduct model checking experiments.

3. Formal Specification

This section summarizes how we formalize qubits, quantum gates, measurements, and then quantum circuits in [2] and describes how we extend it to deal with mixed states.

3.1. Formalization of Qubits, Gates, and Measurements

Pure states of qubits are represented in our formalization as the linear combination of tensor product of the standard basis in Dirac notation with scalars. This representation is also adopted for quantum gates. In our work, we exclusively focus on binary projective measurements conducted on the standard basis. This implies that our measurement operators consist of two elements, denoted as $\boldsymbol{M}_0$ and $\boldsymbol{M}_1$. To formalize a mixed state, we prepare an ensemble of pure states and calculate its corresponding density operator as described in Section 2.

3.2. A Formalization of Quantum Circuits

Our approach to formalizing quantum circuits begins by representing them as a sequence comprising various actions, including quantum gates, measurements, qubit initializations, and other operations, as depicted in Figure 1. Following this, we establish Kripke structures tailored to quantum circuits to conduct model checking that quantum circuits enjoy desired properties.

3.2.1. Elements of Quantum Circuits

A whole quantum state is formalized as a density operator representing a mixed state. In our previous work [2], a whole quantum state is formalized as a pure state, which is a linear combination of the tensor product of basic vectors with Dirac notation and complex numbers.

Classical bits are formalized the same as in our previous work [2]. They are represented as a mapping from circuit indices to Boolean values. Each entry in this mapping takes the form of $(i \mapsto b)$, indicating that the value of the classical bit stored at position c_i is b, with b taking on values of either 0 or 1.

A sequence comprising *quantum gates, measurements*, and *conditional gates* in a quantum circuit is formalized as a list of actions. Each action can take one of the following forms:

- I(i) applies the $\boldsymbol{I}$ gate on qubit at index i,
- X(i) applies the $\boldsymbol{X}$ gate on qubit at index i,
- Y(i) applies the $\boldsymbol{Y}$ gate on qubit at index i,
- Z(i) applies the $\boldsymbol{Z}$ gate on qubit at index i,
- H(i) applies the $\boldsymbol{H}$ gate on qubit at index i,
- CX(i, j) applies the $\boldsymbol{CX}$ gate on qubits at indices i and j,
- CY(i, j) applies the $\boldsymbol{CY}$ gate on qubits at indices i and j,
- CZ(i, j) applies the $\boldsymbol{CZ}$ gate on qubits at indices i and j,
- SWAP(i, j) applies the $\boldsymbol{SWAP}$ gate on qubits at indices i and j,
- CCX(i, j, k) applies the $\boldsymbol{CCX}$ gate on qubits at indices i, j and k,
- CCZ(i, j, k) applies the $\boldsymbol{CCX}$ gate on qubits at indices i, j and k,
- CSWAP(i, j, k) applies the $\boldsymbol{CSWAP}$ gate on qubits at indices i, j and k,
- M(i) measures q_i with the standard basis,
- c[i] == b ? AL checks if the classical bit at c_i equals b, then a list AL of actions is executed.

All actions except the two last actions representing measurements and conditional actions are called basic actions. In this paper, we support more quantum gates than in our previous work [2], including $\boldsymbol{S}$, $\boldsymbol{T}$, $\boldsymbol{CY}$, $\boldsymbol{CZ}$, $\boldsymbol{SWAP}$, $\boldsymbol{CCY}$, $\boldsymbol{CCZ}$, and $\boldsymbol{CSWAP}$ gates.

3.2.2. Kripke Structures of Quantum Circuits

Let K be the Kripke structure formalizing a quantum circuit. In our formalization, we define five distinct observable components as follows:

- `(mState:` ms`)` denotes the mixed quantum state ms.
- `(#qubits:` n`)` denotes the number of qubits n.
- `(bits:` bm`)` denotes the classical bits obtained from measurements and stored in a bit map bm.
- `(prob:` p`)` denotes the probability p at the current quantum state.
- `(actions:` al`)` denotes the action list al, guiding us on how the circuit works.
- `(isEnd:` b`)` denotes termination with Boolean flag b.

Each state in S is expressed as $\{obs\}$, where obs is a soup of those five distinct observable components. The `mState` and `#qubits` observable components are newly added to deal with mixed states compared to our previous work [2].

T now consists of six rewrite rules in our formalization. Let `OCs` be a Maude variable of observable component soups, `MS` and `MS'` be Maude variables of whole quantum states, `BM` be a Maude variable of bit maps, `Prob` and `Prob'` be Maude variables of scalars, `AL` and `AL'` be Maude variables of action lists, `B` be a Maude variable of Boolean values, and `N`, `N'`, `N1`, and `N2` are Maude variables of natural numbers.

The first rewrite rule is as follows:

```
crl [U] : {(mState: MS) (actions: (A AL)) (#qubits: N) OCs}
=> {(mState: MS') (actions: AL) (#qubits: N) OCs}
if isBasicAction(A) /\ MS' := unitary(MS, A, N) .
```

The rule `U` simulates unitary transformation on the whole quantum state in `mState` observable component if its basic action appears in `actions` observable component.

The next two rewrite rules are as follows:

```
crl [M0] :
{(mState: MS) (actions: (M(N') AL)) (prob: Prob) (bits: BM) (#qubits: N) OCs}
=> {(mState: MS') (actions: AL) (prob: (Prob .* Prob')) (bits: insert(N', 0, BM))
    (#qubits: N) OCs}
if {mState: MS', prob: Prob'} := measure(MS, N, P0, N') .

crl [M1] :
{(mState: MS) (actions: (M(N') AL)) (prob: Prob) (bits: BM) (#qubits: N) OCs}
=> {(mState: MS') (actions: AL) (prob: (Prob .* Prob')) (bits: insert(N', 1, BM))
    (#qubits: N) OCs}
if {mState: MS', prob: Prob'} := measure(MS, N, P1, N') .
```

where `P0` and `P1` are Maude constants of matrices representing the measurement operators $\boldsymbol{M}_0$ and $\boldsymbol{M}_1$, respectively. Therefore, the rules `M0` and `M1` govern the process of measuring a qubit at index `N'` with the measurement operators $\boldsymbol{M}_0$ and $\boldsymbol{M}_1$, respectively. In these rules, the classical outcomes are stored accordingly into the bit map in `bits` observable component;

the probabilities and the post-measurement mixed states are updated in `prob` and `mState` observable components, respectively. It is important to note that these two rules introduce non-deterministic probabilistic transitions when measuring a single qubit.

The next rewrite rule is identical to the one presented in our previous work [2]. It outlines the processes of conditionally executing the next actions based on classical bits obtained from measurements if applicable.

```
rl [cif] :
{(qstate: Q) (bits: ((N |-> N1),BM)) (actions: ((c[N] == N2 ? AL') AL)) OCs}
=> {(qstate: Q) (bits: ((N |-> N1), BM))
    (actions: ((if (N1 == N2) then AL' else nil fi) AL)) OCs} .
```

This rule says that if `c[N] == N2 ? AL'` exists in the action list and the classical bit `N1` at index `N` equals the conditional value `N2`, then the action list `AL'` is added at the beginning of the action list `AL` in `actions` observable component. This means that `AL'` will be executed next. Otherwise, it is simply ignored and `AL` remains unchanged.

The last two rules are the same as in our previous work [2] as follows:

```
rl [end] : {(actions: nil) (isEnd: false) OCs}
=> {(actions: nil) (isEnd: true) OCs} .

rl [stutter]: {(isEnd: true) OCs} => {(isEnd: true) OCs} .
```

The rule `end` indicates termination when the action list is empty, denoted as `nil`. On the other hand, the rule `stutter` is to make T total when `isEnd` observable component is true.

4. A Case Study: Quantum Teleportation

For the sake of simplicity, this section only demonstrates how to use our symbolic approach with density operators to conduct model checking for Quantum Teleportation [7]. Meanwhile, other communication protocols are similar and the full specifications of all quantum communication protocols concerned in this paper are publicly available at https://github.com/canhminhdo/QTC-Maude under the `mixed-states` folder.

4.1. Introduction

Quantum Teleportation, as described in [7], leverages the unique properties of entanglement in quantum mechanics to transmit an unknown quantum state $|\psi\rangle$ from Alice to Bob, utilizing only three qubits and two classical bits. This protocol holds significant importance because of the no-cloning theorem [13], which prohibits the exact copy of an arbitrary unknown quantum state. As a result, the protocol becomes a crucial method for transmitting an arbitrary unknown quantum state from one source to another.

The circuit illustrated in Figure 1 presents how the protocol works. Alice manipulates on qubits q_0 and q_1, and Bob manipulates on qubit q_2 as follows:

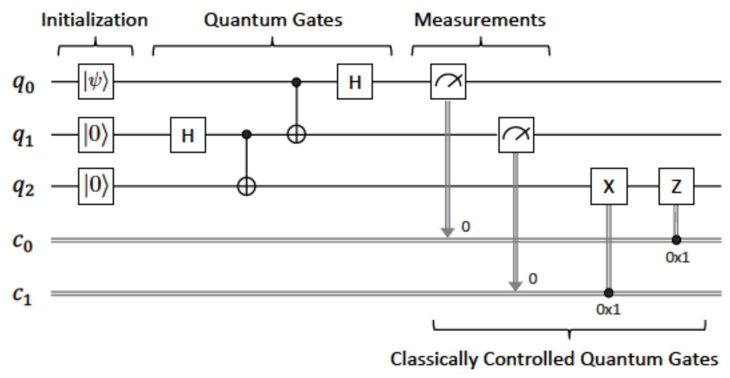

Figure 1: Quantum Teleportation

- *State preparation*: Initially, an unknown state $|\psi\rangle$ is prepared at qubit q_0, where $|\psi\rangle = \alpha\,|\mathbf{0}\rangle + \beta\,|\mathbf{1}\rangle$. The complex numbers α and β are such that $|\alpha|^2 + |\beta|^2 = 1$. Initially, qubits q_1 and q_2 are in the state $|\mathbf{0}\rangle$.
- *Gate operations*: A sequence of quantum gates is applied to manipulate the three qubits. This includes the single-qubit Hadamard $\boldsymbol{H}$ and two-qubit controlled-NOT $\boldsymbol{CX}$ gates. The following operations are performed: Alice and Bob share an entangled state by applying the $\boldsymbol{H}$ gate to qubit q_1 and the $\boldsymbol{CX}$ gate to qubits q_1 and q_2. Alice further manipulates qubits q_0 and q_1 by applying the $\boldsymbol{CX}$ gate to them, followed by the $\boldsymbol{H}$ gate to qubit q_0.
- *Measurement*: Qubits q_0 and q_1 are measured, yielding two classical outcomes (0 or 1) stored in c_0 and c_1, respectively.
- *Conditional gates*: Based on the classical bits in c_0 and c_1, conditional single-qubit $\boldsymbol{X}$ and $\boldsymbol{Z}$ gates are applied to qubit q_2. Specifically, the $\boldsymbol{X}$ gate is used if c_1 equals one, followed by the $\boldsymbol{Z}$ gate if c_0 equals one.

Finally, Bob will receive the original quantum state $|\psi\rangle$, while Alice will no longer have it. Our goal is to verify whether Alice can successfully transmit an arbitrary unknown quantum state to Bob at the end of this protocol by using our symbolic model checking approach with density operators.

4.2. Formalization of Quantum Teleportation

We can describe the circuit for Quantum Teleportation based on the actions formalized in Sect. 3 as follows:

```
H(1) CX(1, 2) CX(0, 1) H(0) M(0) M(1) (c[1] == 1 ? X(2)) (c[0] == 1 ? Z(2))
```

Let ES be an ensemble of pure states as follows:

```
{ (a . |0> + b . |1>) (x) |0>  (x) |0>, 1 }
```

where `(x)` denotes the tensor product; and `a` and `b` are Maude constants of scalars denoting arbitrary scalars such that $|\mathrm{a}|^2 + |\mathrm{b}|^2 = 1$. We suppose that the mixed state consists of only one pure state with a certain probability in the ensemble.

The set I of initial states for Quantum Teleportation includes only one initial state as follows:

```
{(isEnd: false)
(#qubits: findN(ES))
(mState: convert(ES)
(prob: 1)
(bits: empty)
(actions: H(1) CX(1, 2) CX(0, 1) H(0)
          M(0) M(1)
          c[1] == 1 ? X(2)
          c[0] == 1 ? Z(2))}
```

where `findN(_)` and `convert(_)` are two functions to calculate the number of qubits and the density operator of a mixed state given an ensemble. Initially, `isEnd` observable component is false, `prob` observable component is one, `mState` represent the corresponding density operator of the symbolic state as the input state of the protocol, `actions` observable component contains the action list describing how the protocol works.

4.3. Model Checking Quantum Teleportation

Let K and `init` be the Kripke structure and the initial state for Quantum Teleportation, respectively. In order to perform model checking on the Kripke structure K and verify that it satisfies the desired properties, we define the set of atomic propositions A and the labeling function L. A contains one atomic proposition, denoted as `isSuccess`. L is defined as follows:

```
eq {(isEnd: true) (mState: MS) (prob: Prob) (#qubits: N) OCs} |= isSuccess
= Prob > 0 implies
  tr[1]((tr[0](MS, N)), N) == (I (x) I (x) (PSI x (PSI)^+)) .
eq {OCs} |= PROP = false [owise] .
```

where `PSI` is the input state of the protocol being transferred, the function `tr[_](_,_)` take inputs as the index at which the information of the qubit is erased, the mixed state, and the number of qubits. The function will erase information of a subsystem from the mixed state at an index. Using this function we can retain the information of the qubit at the index 2 in the density operator. This function works as the partial trace over a quantum system [10].

The two equations say that `isSuccess` holds at a state if the state contains `(isEnd: true)`, `(mState: MS)`, `(prob: Prob)`, and `(#qbits: N)` such that the condition `tr[1]((tr[0](MS, N)), N) == (I (x) I (x) (PSI x (PSI)^+)` holds whenever `Prob > 0` holds, meaning that the qubit received by Bob at the end is equal to the qubit sent by Alice at the beginning with a non-zero probability by means of density operators. Notice that, the use of density operators to represent quantum states can eliminate the global phase when comparing two quantum states. Let `teleProp` be an LTL formula defined as `<> isSuccess`, where `<>` is the eventual temporal connective.

Table 2
Experimental results with pure states and mixed states for representing quantum states

Protocol	Qubits	States	Pure States		Mixed States	
			Rewrite Steps	Time	Rewrite Steps	Time
Superdense Coding	2	9	685	≈ 0ms	2,088	2ms
Quantum Teleportation	3	27	4,340	3ms	29,095	30ms
Quantum Secret Sharing	4	65	16,449	9ms	211,831	519ms
Entanglement Swapping	4	33	6,930	4ms	56,193	40ms

We want to model check that $K = \langle S, I, T, A, L \rangle$ satisfies `teleProp` from the initial state `init` in Maude as follows:

```
red modelCheck(init, teleProp) .
```

No counterexample is found in just a few moments and so K satisfies `teleProp`. In other words, we successfully verify the correctness of Quantum Teleportation by using our symbolic model checking approach with density operators.

5. Experimental Results

We used an iMac that carries a 4 GHz microprocessor with eight cores and 32 GB memory RAM to conduct experiments in this section. As case studies, we conduct model checking experiments to verify the correctness of several quantum communication protocols with our approach in which both pure states and mixed states are used for representing quantum states as follows:

- Superdense Coding [6] for transmitting two classical bits using an entangled state,
- Quantum Teleportation [7] for teleporting an arbitrary pure state by sending two bits of classical information,
- Quantum Secret Sharing [8] for teleporting a pure state from a sender (Alice) to a receiver (Bob) with the help of a third party (Charlie),
- Entanglement Swapping [9] for creating a new entangled state,

Superdense Coding is the simplest one that uses only two qubits; Quantum Teleportation uses three qubits; Quantum Secret Sharing proposed relying on the mechanism of Quantum Teleportation uses four qubits; and Entanglement Swapping uses four qubits. Note that the properties being verified for both pure states and mixed states are identical in our experiments. Our support tool and case studies are publicly available at https://github.com/canhminhdo/QTC-Maude under the `mixed-states` folder.

The experimental results are shown in Table 2. The second and third columns denote the number of qubits in each protocol and the number of states in the reachable state space of each protocol under model checking, respectively. Notice that the number of states of each protocol under model checking is the same for both pure states and mixed states. The fourth and fifth columns denote the number of rewriting steps performed for each protocol and the verification time when pure states are used to represent quantum states; and similarly for the last two columns when mixed states are used. Note that, in these experiments, the mixed state in each protocol consists of only one pure state with a certain probability in an ensemble.

Although all model checking experiments were completed in just a few moments, the number of rewriting steps and the verification time for mixed states is considerably larger than that for pure states. This is not surprising because we intentionally used an ensemble that contains only one pure state with a certain probability in these experiments, and the pure state is represented by a vector $|\psi\rangle$, while the mixed state is represented by a density operator $|\psi\rangle\langle\psi|$, which is a matrix. The calculation for a matrix is more expensive than that for a vector. However, with mixed states, we can present a statistical mixture of multiple pure states and eliminate the global phase compared with the pure states. Our symbolic model checking quantum circuits in Maude can handle both pure states and mixed states, showing its usefulness in quantum circuit verification.

6. Related Work

Gay et al. [14] introduced a method for employing classical model checkers, such as PRISM, a probabilistic model checker, to verify quantum protocols. In their approach, each quantum state is assigned a distinct number of identifiers and transitions from one unique number to another representing the operations of quantum gates and measurements. However, this method necessitates the prior enumeration of states and the computation of state transitions, which are subsequently encoded into a PRISM specification. Despite their development of a tool called PRISMGEN to automate this process, its practicality is limited in real-world scenarios, supporting only two or three qubits due to the exponential proliferation of state numbers. In contrast, our approach does not require the pre-enumeration of states, as quantum states are directly formalized using Dirac notation with scalar values. Furthermore, we utilize rewrite rules to represent the effects of quantum gates and measurements, rendering our approach capable of handling a larger number of qubits. For instance, we have successfully verified the correctness of Quantum Secret Sharing and Entanglement Swapping involving four qubits using our approach.

Our approach to symbolic quantum circuit model checking draws inspiration from a prior work by Yuan et al. [15], which closely resembles our methodology. However, it's worth noting that their approach primarily centers around theorem proving rather than model checking. They adopt Dirac notation and a set of rules to formalize quantum states, gates, measurements, and reason about quantum circuits within the Coq interactive theorem prover. Nevertheless, a notable distinction is that their approach often necessitates users to supply essential lemmas to facilitate the completion of their proofs, a task that is generally considered challenging. In contrast, our approach operates entirely autonomously, requiring no human intervention.

7. Conclusion

We have extended our symbolic approach to deal with mixed states using density operators and have developed a support tool in Maude. Several quantum communication protocols have been successfully analyzed using our approach/support tool, including Superdense Coding, Quantum Teleportation, Quantum Secret Sharing, and Entanglement Swapping. This demonstrates the usefulness of our symbolic model checking quantum circuits with density operators in Maude. As one piece of future work, we would conduct more case studies, where a statistical mixture of multiple pure states is realistically presented in them.

Acknowledgments

The research was supported by JAIST Research Grant for Fundamental Research, by JST SICORP Grant Number JPMJSC20C2, and by JSPS KAKENHI Grant Number JP23K19959.

References

[1] P. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: Proceedings 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124–134. doi:10.1109/SFCS.1994.365700.

[2] C. M. Do, K. Ogata, Symbolic model checking quantum circuits in maude, in: The 35th International Conference on Software Engineering and Knowledge Engineering, SEKE 2023, 2023, pp. 103–108. doi:10.18293/SEKE2023-014.

[3] P. A. M. Dirac, A new notation for quantum mechanics, Mathematical Proceedings of the Cambridge Philosophical Society 35 (1939) 416–418. doi:10.1017/S0305004100021162.

[4] M. Clavel, et al., All About Maude, volume 4350 of *Lecture Notes in Computer Science*, Springer, 2007. doi:10.1007/978-3-540-71999-1.

[5] J. Meseguer, Twenty years of rewriting logic, The Journal of Logic and Algebraic Programming 81 (2012) 721–781. doi:10.1016/j.jlap.2012.06.003.

[6] C. H. Bennett, S. J. Wiesner, Communication via one- and two-particle operators on einstein-podolsky-rosen states, Phys. Rev. Lett. 69 (1992) 2881–2884. doi:10.1103/PhysRevLett.69.2881.

[7] C. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, W. Wootters, Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels, Physical review letters 70 (1993) 1895–1899. doi:10.1103/PhysRevLett.70.1895.

[8] M. Hillery, V. Bužek, A. Berthiaume, Quantum secret sharing, Physical Review A 59 (1999) 1829–1834. doi:10.1103/physreva.59.1829.

[9] M. Żukowski, A. Zeilinger, M. A. Horne, A. K. Ekert, "Event-ready-detectors" Bell experiment via entanglement swapping, Phys. Rev. Lett. 71 (1993) 4287–4290. doi:10.1103/PhysRevLett.71.4287.

[10] M. A. Nielsen, I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition, Cambridge University Press, 2010. doi:10.1017/CBO9780511976667.

[11] T. Takagi, C. M. Do, K. Ogata, Automated quantum program verification in dynamic quantum logic (to appear), in: DaLí: Dynamic Logic – New trends and applications, 2023.

[12] E. M. Clarke, T. A. Henzinger, H. Veith, R. Bloem (Eds.), Handbook of Model Checking, Springer, 2018. doi:10.1007/978-3-319-10575-8.

[13] W. K. Wootters, W. H. Zurek, A single quantum cannot be cloned, Nature 299 (1982) 802–803. doi:10.1038/299802a0.

[14] S. Gay, R. Nagarajan, N. Papanikolaou, Probabilistic model–checking of quantum protocols, 2005. doi:10.48550/arXiv.quant-ph/0504007.

[15] W. Shi, Q. Cao, Y. Deng, H. Jiang, Y. Feng, Symbolic reasoning about quantum circuits in coq, J. Comput. Sci. Technol. 36 (2021) 1291–1306. doi:10.1007/s11390-021-1637-9.