Noname manuscript No.
(will be inserted by the editor)

Symbolic Reasoning about Quantum Circuits in Coq

Wenjun Shi · Qinxiang Cao · Yuxin Deng · Hanru Jiang · Yuan Feng

May 25, 2020

Abstract A quantum circuit is a computational unit that transforms an input quantum state to an output one. A natural way to reason about its behavior is to compute explicitly the unitary matrix implemented by it. However, when the number of qubits increases, the matrix dimension grows exponentially and the computation becomes intractable.

In this paper, we propose a symbolic approach to reasoning about quantum circuits. It is based on a small set of laws involving some basic manipulations on vectors and matrices. This symbolic reasoning scales better than the explicit one and is well suited to be automated in Coq, as demonstrated with some typical examples.

Keywords Symbolic reasoning · Quantum circuits · Dirac notation · Coq.

1 Introduction

Quantum circuit is a natural model of quantum computation [2]. It is a computational unit that transforms an input quantum state to an output one. Once a quantum circuit is designed to implement an algorithm, it is indispensable to analyze the circuit and ensure that it indeed conforms to the requirements of the algorithm. When a large number of qubits are involved, manually reasoning about a circuit's behavior is tedious and error-prone. One way of reasoning about quantum circuits (semi-) automatically and reliably is to mechanize the reasoning procedure in an interactive theorem prover, such as the Coq proof assistant [3]. For example, Rand et al. [5] verified a few quantum algorithms in Coq, using some semi-automated strategies to generate machine checkable proofs.

Existing approaches have apparent drawbacks in both efficiency and human readability. Quantum states and operations are represented and computed using matrices explicitly, and their comparison is done in an element-wise way, thus is highly non-scalable with respect to qubit numbers. Furthermore, as the system dimension grows,

East China Normal University, China · Shanghai Jiao Tong University, China · East China Normal University, China · Peng Cheng Laboratory, China · University of Technology Sydney, Australia

it is almost impossible for human beings to read the matrices printed by the theorem prover.

In this paper, we propose a symbolic approach for reasoning about the behavior of quantum circuits in Coq, which improves both efficiency of the reasoning procedure and readability of matrix representations. The main contributions of this paper include:

- A matrix representation in Coq using the Dirac notation [6], which is commonly used in quantum mechanics. Matrices are represented as combinations of $|0\rangle$, $|1\rangle$, scalars, and a set of basic operators such as tensor product and matrix multiplication. Here $|0\rangle$ and $|1\rangle$ are the Dirac notation for 2-dimensional column vectors $[1\ 0]^T$ and $[0\ 1]^T$, respectively. In this way, we have a concise representation for sparse matrices which are common in quantum computation.
- A tactic library for (semi-)automated symbolic reasoning about matrices. The tactics are based on a small set of inference laws (lemmas in Coq). The key idea is to reduce matrix multiplications in the form of $\langle i|j\rangle$ into scalars, and simplify the matrix representation by absorbing ones and eliminating zeros. In this way, our approach reasons about matrices by (semi-)automated rewriting instead of actually computing the matrices, and outperforms the computational approach of Rand et al. [5], as shown in proving the functional correctness of some typical quantum algorithms in Section 6.

We illustrate the intuition of our tactics by the following simple example which computes the result of applying the Pauli-X gate to the $|0\rangle$ state. In an explicit matrix-vector multiplication form, it reads as follows:

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \times 1 + 1 \times 0 \\ 1 \times 1 + 0 \times 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

and four multiplications are required for the whole computation. By contrast, if we use the Dirac notation for X and apply distribution and associativity laws, then

$$\begin{aligned} X|0\rangle &= (|0\rangle\langle 1| + |1\rangle\langle 0|)|0\rangle \\ &= |0\rangle\langle 1|0\rangle + |1\rangle\langle 0|0\rangle \\ &= 0|0\rangle + 1|1\rangle \\ &= |1\rangle. \end{aligned}$$

Note that the two terms $\langle 1|0\rangle$ and $\langle 0|0\rangle$ are reduced (symbolically) to 0 and 1, respectively. Consequently, no multiplication is required at all.

The rest of the paper is structured as follows. In Section 2 we recall some basic notation from linear algebra and quantum mechanics. In Section 3 we introduce a symbolic approach to reasoning about quantum circuits. In Section 4 we discuss some problems of representing matrices using Coq's type system and our solutions. In Section 5 we propose two notions of equivalence for quantum circuits. In Section 6 we conduct a few case studies. In Section 7 we discuss some related works. Finally, we conclude in Section 8.

The Coq scripts of our tactic library and the examples used in our case studies are available at the following link

`https://github.com/Vickyswj/DiracRepr`.

2 Preliminaries

For the convenience of the reader, we briefly recall some basic notions from linear algebra and quantum theory which are needed in this paper. For more details, we refer to [2].

2.1 Basic linear algebra

A *Hilbert space* $\mathcal{H}$ is a complete vector space equipped with an inner product

$$\langle \cdot | \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{C}$$

such that

1. $\langle \psi | \psi \rangle \geq 0$ for any $|\psi\rangle \in \mathcal{H}$, with equality if and only if $|\psi\rangle = 0$;
2. $\langle \phi | \psi \rangle = \langle \psi | \phi \rangle^*$;
3. $\langle \phi | \sum_i c_i |\psi_i\rangle = \sum_i c_i \langle \phi | \psi_i \rangle$,

where $\mathbb{C}$ is the set of complex numbers, and for each $c \in \mathbb{C}$, c^* stands for the complex conjugate of c. A vector $|\psi\rangle \in \mathcal{H}$ is *normalised* if its length $\sqrt{\langle \psi | \psi \rangle}$ is equal to 1. Two vectors $|\psi\rangle$ and $|\phi\rangle$ are *orthogonal* if $\langle \psi | \phi \rangle = 0$. An *orthonormal basis* of a Hilbert space $\mathcal{H}$ is a basis $\{|i\rangle\}$ where each $|i\rangle$ is normalised and any pair of them are orthogonal.

Let $\mathcal{L}(\mathcal{H})$ be the set of linear operators on $\mathcal{H}$. For any $A \in \mathcal{L}(\mathcal{H})$, A is *Hermitian* if $A^\dagger = A$ where $A^\dagger$ is the adjoint operator of A such that $\langle \psi | A^\dagger | \phi \rangle = \langle \phi | A | \psi \rangle^*$ for any $|\psi\rangle, |\phi\rangle \in \mathcal{H}$. A linear operator $A \in \mathcal{L}(\mathcal{H})$ is *unitary* if $A^\dagger A = A A^\dagger = I_\mathcal{H}$ where $I_\mathcal{H}$ is the identity operator on $\mathcal{H}$. The *trace* of A is defined as $\mathrm{tr}(A) = \sum_i \langle i | A | i \rangle$ for some given orthonormal basis $\{|i\rangle\}$ of $\mathcal{H}$. It is worth noting that the trace function is actually independent of the orthonormal basis selected. It is also easy to check that the trace function is linear and $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ for any $A, B \in \mathcal{L}(\mathcal{H})$.

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 $|\psi_1 \psi_2\rangle = |\psi_1\rangle |\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$ with $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$. Here the tensor product of two vectors is defined by a new vector such that

$$\left(\sum_i \lambda_i |\psi_i\rangle \right) \otimes \left(\sum_j \mu_j |\phi_j\rangle \right) = \sum_{i,j} \lambda_i \mu_j |\psi_i\rangle \otimes |\phi_j\rangle.$$

Then $\mathcal{H}_1 \otimes \mathcal{H}_2$ is also a Hilbert space where the inner product is defined in the following way: for any $|\psi_1\rangle, |\phi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle, |\phi_2\rangle \in \mathcal{H}_2$,

$$\langle \psi_1 \otimes \psi_2 | \phi_1 \otimes \phi_2 \rangle = \langle \psi_1 | \phi_1 \rangle_{\mathcal{H}_1} \langle \psi_2 | \phi_2 \rangle_{\mathcal{H}_2}$$

where $\langle \cdot | \cdot \rangle_{\mathcal{H}_i}$ is the inner product of $\mathcal{H}_i$. For any $A_1 \in \mathcal{L}(\mathcal{H}_1)$ and $A_2 \in \mathcal{L}(\mathcal{H}_2)$, $A_1 \otimes A_2$ is defined as a linear operator in $\mathcal{L}(\mathcal{H}_1 \otimes \mathcal{H}_2)$ such that for each $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$,

$$(A_1 \otimes A_2) |\psi_1 \psi_2\rangle = A_1 |\psi_1\rangle \otimes A_2 |\psi_2\rangle.$$

2.2 Basic quantum mechanics

According to von Neumann's formalism of quantum mechanics [24], an isolated physical system is associated with a Hilbert space which is called the *state space* of the system. A *pure state* of a quantum system is a normalised vector in its state space, and a *mixed state* is represented by a density operator on the state space. Here a density operator ρ on Hilbert space $\mathcal{H}$ is a positive linear operator such that $\mathrm{tr}(\rho) = 1$. Another equivalent representation of a density operator is a probabilistic ensemble of pure states. In particular, given an ensemble $\{(p_i, |\psi_i\rangle)\}$ where $p_i \geq 0$, $\sum_i p_i = 1$, and $|\psi_i\rangle$ are pure states, then $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ is a density operator. Conversely, each density operator can be generated by an ensemble of pure states in this way. Finally, a pure state can be regarded as a special mixed state.

The state space of a composite system (for example, a quantum system consisting of many qubits) is the tensor product of the state spaces of its components. Note that in general, the state of a composite system cannot be decomposed into the tensor product of the reduced states on its component systems. A well-known example is the 2-qubit state

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

This kind of state is called an *entangled state*. Entanglement is an outstanding feature of quantum mechanics which has no counterpart in the classical world, and is the key to many quantum information processing tasks.

The evolution of a closed quantum system is described by a unitary operator on its state space. If the states of the system at times t_1 and t_2 are $|\psi_1\rangle$ and $|\psi_2\rangle$, respectively, then $|\psi_2\rangle = U|\psi_1\rangle$ for some unitary operator U which depends only on t_1 and t_2. A convenient way to understand unitary operators is in terms of their matrix representations. In fact, the unitary operator and matrix viewpoints turn out to be completely equivalent. An m by n complex unitary matrix U with entries U_{ij} can be considered as a unitary operator sending vectors in the vector space $\mathbb{C}^n$ to the vector space $\mathbb{C}^m$, under matrix multiplication of the matrix U by a vector in $\mathbb{C}^n$.

We often denote a single qubit as a vector $|\psi\rangle = a|0\rangle + b|1\rangle$ parameterized by two complex numbers satisfying $|a|^2 + |b|^2 = 1$. A unitary operator for a qubit is then described by a 2×2 unitary matrices. Quantum circuits are a popular model for quantum computation, where quantum gates usually stand for basic unitary operators whose mathematical meanings are given by appropriate unitary matrices. Some commonly used quantum gates to appear in the current work include the 1-qubit Hadamard gate H, the Pauli gates I_2, X, Y, Z, and the controlled-NOT gate CX performed on two qubits. Their matrix representation is given below:

$$I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

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

For example, in Figure 1 we display a circuit that can generate the 3-qubit GHZ state [9]. In the circuit, a Hadamard gate is applied on the first qubit, then two controlled-NOT gates are used, with the first qubit controlling the second, which in turn controls the third.

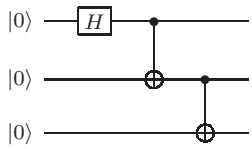


Fig. 1 A circuit for creating the 3-qubit GHZ state

A quantum *measurement* is described by a collection $\{M_m\}$ of measurement operators, where the indices m refer to the measurement outcomes. It is required that the measurement operators satisfy the completeness equation $\sum_m M_m^\dagger M_m = I_{\mathcal{H}}$. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement, then the probability that result m occurs is given by

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle,$$

and the state of the system after the measurement is $\frac{M_m|\psi\rangle}{\sqrt{p(m)}}$. If the states of the system at times t_1 and t_2 are mixed, say ρ_1 and ρ_2, respectively, then $\rho_2 = U\rho_1 U^\dagger$ after the unitary operation U is applied on the system. For the same measurement $\{M_m\}$ as above, if the system is in the mixed state ρ, then the probability that measurement result m occurs is given by

$$p(m) = \mathrm{tr}(M_m^\dagger M_m \rho),$$

and the state of the post-measurement system is $\frac{M_m \rho M_m^\dagger}{p(m)}$ provided that $p(m) > 0$.

3 Symbolic reasoning

Our symbolic reasoning is based on terms constructed from scalars and basic vectors using some constructors:

- Scalars are complex numbers. We write $\mathbb{C}$ for the set of complex numbers. Our formal treatment of complex numbers is based on the definitions and rewriting tactics from the Coquelicot [8] library of Coq.
- Basic vectors are the base states of a qubit, i.e., $|0\rangle$ and $|1\rangle$ in the Dirac notation. Mathematically, $|0\rangle$ stands for the vector $[1\ 0]^T$ and $|1\rangle$ for $[0\ 1]^T$.
- Constructors include the scalar product $\cdot$, matrix product $\times$, matrix addition $+$, tensor product $\otimes$, and the conjugate transpose $A^\dagger$ of a matrix A.

Scalars:	$\mathbb{C}$	
Basic vectors:	$\|0\rangle$, $\|1\rangle$	
Operators:	$\cdot$, $\times$, $+$, $\otimes$, $\dagger$	
Laws:	**L**1	$\langle 0\|0\rangle = \langle 1\|1\rangle = 1,\ \langle 0\|1\rangle = \langle 1\|0\rangle = 0$
	L2	Associativity of $\cdot,\ \times,\ +,\ \otimes$
	L3	$0 \cdot A_{m\times n} = \mathbf{0}_{m\times n},\ c \cdot \mathbf{0} = \mathbf{0},\ 1 \cdot A = A$
	L4	$c \cdot (A + B) = c \cdot A + c \cdot B$
	L5	$c \cdot (A \times B) = (c \cdot A) \times B = A \times (c \cdot B)$
	L6	$c \cdot (A \otimes B) = (c \cdot A) \otimes B = A \otimes (c \cdot B)$
	L7	$\mathbf{0}_{m\times n} \times A_{n\times p} = \mathbf{0}_{m\times p} = A_{m\times n} \times \mathbf{0}_{n\times p}$
	L8	$I_m \times A_{m\times n} = A_{m\times n} = A_{m\times n} \times I_n,\ \ I_m \otimes I_n = I_{mn}$
	L9	$\mathbf{0} + A = A = A + \mathbf{0}$
	L10	$\mathbf{0}_{m\times n} \otimes A_{p\times q} = \mathbf{0}_{mp\times nq} = A_{p\times q} \otimes \mathbf{0}_{m\times n}$
	L11	$(A + B) \times C = A \times C + B \times C,\ C \times (A + B) = C \times A + C \times B$
	L12	$(A + B) \otimes C = A \otimes C + B \otimes C,\ C \otimes (A + B) = C \otimes A + C \otimes B$
	L13	$(A \otimes B) \times (C \otimes D) = (A \times C) \otimes (B \times D)$
	L14	$(c \cdot A)^\dagger = c^* \cdot A^\dagger,\ (A \times B)^\dagger = B^\dagger \times A^\dagger$
	L15	$(A + B)^\dagger = A^\dagger + B^\dagger,\ (A \otimes B)^\dagger = A^\dagger \otimes B^\dagger$
	L16	$(A^\dagger)^\dagger = A$

Table 1 Terms and laws

In Dirac notations, $\langle 0|$ represents the dual of $|0\rangle$, i.e. $|0\rangle^\dagger$; similarly for $\langle 1|$. The term $\langle j| \times |k\rangle$ is abbreviated into $\langle j|k\rangle$, for any $j, k \in \{0, 1\}$. This notation introduces an intuitive explanation of quantum operation. For example, the effect of the X operator is to map $|0\rangle$ into $|1\rangle$ and $|1\rangle$ into $|0\rangle$. Thus we define X in Coq as $|0\rangle\langle 1| + |1\rangle\langle 0|$, instead of $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Then it is obvious that $X|0\rangle = |1\rangle\langle 0|0\rangle + |0\rangle\langle 1|0\rangle = |1\rangle$ and similarly for $X|1\rangle$.

Some commonly used vectors and gates can be derived from the basic terms. For example, we define the vectors $|+\rangle,\ |-\rangle$, the Hadamard matrix H, the Pauli-X gate, and the controlled-NOT gate CX as follows:

$$
\begin{aligned}
|+\rangle &= \tfrac{1}{\sqrt{2}} \cdot |0\rangle + \tfrac{1}{\sqrt{2}} \cdot |1\rangle \\
|-\rangle &= \tfrac{1}{\sqrt{2}} \cdot |0\rangle + (-\tfrac{1}{\sqrt{2}}) \cdot |1\rangle \\
B_0 &= |0\rangle \times \langle 0| \\
B_1 &= |0\rangle \times \langle 1| \\
B_2 &= |1\rangle \times \langle 0| \\
B_3 &= |1\rangle \times \langle 1| \\
H &= \tfrac{1}{\sqrt{2}} \cdot B_0 + \tfrac{1}{\sqrt{2}} \cdot B_1 + \tfrac{1}{\sqrt{2}} \cdot B_2 + (-\tfrac{1}{\sqrt{2}}) \cdot B_3 \\
X &= B_1 + B_2 \\
CX &= B_0 \otimes I_2 + B_3 \otimes X
\end{aligned}
$$

Notice that using the matrices B_j ($j \in \{0, 1, 2, 3\}$), linear combination and tensor product, we can represent any matrix implemented by a quantum circuit without measurements.

Suppose the state of a quantum system is represented by a vector. The central idea of our symbolic reasoning is to employ the laws in Table 1 to rewrite terms, trying to put together the basic vectors and simplify them using the laws in **L**1. Technically, we design a series of strategies for that purpose.

Firstly, we define a strategy called `orthogonal_reduce` to verify that the laws in **L**1 are sound. In this case, we explicitly represent $|0\rangle$ and $|1\rangle$ as matrices. Both of them are 2×1 matrices, so we can use matrix multiplication to prove that the elements in the matrices on both sides of the equation are consistent. For example, the law $\langle 0|0\rangle = 1$ is actually shown via $[1\ 0]\begin{bmatrix}1\\0\end{bmatrix} = 1$. We add the laws in **L**1 to the set named `S_db`. The laws in **L**2-16 are also proved through explicit matrix representation. We establish their soundness and then collect them in a library that contains many useful properties about matrices.

Secondly, we design a strategy called `base_reduce` to prove some equations about the four basic matrices $B_0, ..., B_3$ acting on base states $|0\rangle$ and $|1\rangle$. For example, let us consider the commonly used equation $B_0 \times |0\rangle = |0\rangle$. We first represent B_0 by $|0\rangle \times \langle 0|$, then use the associativity of matrix multiplication to form the subterm $\langle 0| \times |0\rangle$. Now we can use the proved laws in **L**1 to rewrite $\langle 0| \times |0\rangle$ into 1. The last step is to deal with scalar multiplications. We add these equations to the set named `B_db`.

Thirdly, we introduce a strategy called `gate_reduce` to prove some equations about the matrices I, X, Y, Z, H acting on base states. For example, consider the equation $X \times |0\rangle = |1\rangle$. We first expand X into $B_1 + B_2$. In order to prove the equation $(B_1 + B_2) \times |0\rangle = |1\rangle$, we use the distributivity of matrix multiplication over addition to rewrite the left hand side of the equation into the sum of $B_1 \times |0\rangle$ and $B_2 \times |0\rangle$. Then we employ the proved laws in `B_db` to rewrite them. Eventually, we deal with scalar multiplications and cancel zero matrices. We add these equations to the set named `G_db`.

Lastly, we propose a strategy called `operate_reduce` that puts together all the results above to reason about circuits applied to input states represented in vector form. For example, let us revisit the 3-qubit GHZ state. The state can be generated by applying the circuit in Figure 1 to the initial state $|0\rangle \otimes |0\rangle \otimes |0\rangle$. We would like to verify that the output state is indeed what we expect by establishing the following equation.

$$\begin{array}{l}(I_2 \otimes CX) \times (CX \otimes I_2) \times (H \otimes I_2 \otimes I_2) \times (|0\rangle \otimes |0\rangle \otimes |0\rangle) \\ = \frac{1}{\sqrt{2}} \cdot (|0\rangle \otimes |0\rangle \otimes |0\rangle) + \frac{1}{\sqrt{2}} \cdot (|1\rangle \otimes |1\rangle \otimes |1\rangle)\end{array} \tag{1}$$

We first expand CX into $B_0 \otimes I_2 + B_3 \otimes X$. So the left hand side of the equation turns into

$$\begin{array}{l}(I_2 \otimes (B_0 \otimes I_2 + B_3 \otimes X)) \times ((B_0 \otimes I_2 + B_3 \otimes X) \otimes I_2) \\ \times (H \otimes I_2 \otimes I_2) \times (|0\rangle \otimes |0\rangle \otimes |0\rangle).\end{array} \tag{2}$$

Next, we try all the distribution laws for matrix addition to rewrite (2) into a sum of matrices without addition operator. So it takes the following form:

$$\begin{array}{l}((I_2 \otimes B_0 \otimes I_2) \times (B_0 \otimes I_2 \otimes I_2) \times (H \otimes I_2 \otimes I_2) \times (|0\rangle \otimes |0\rangle \otimes |0\rangle) \\ +((I_2 \otimes B_0 \otimes I_2) \times (B_3 \otimes X \otimes I_2) \times (H \otimes I_2 \otimes I_2) \times (|0\rangle \otimes |0\rangle \otimes |0\rangle) \\ +((I_2 \otimes B_3 \otimes X) \times (B_0 \otimes I_2 \otimes I_2) \times (H \otimes I_2 \otimes I_2) \times (|0\rangle \otimes |0\rangle \otimes |0\rangle) \\ +((I_2 \otimes B_3 \otimes X) \times (B_3 \otimes X \otimes I_2) \times (H \otimes I_2 \otimes I_2) \times (|0\rangle \otimes |0\rangle \otimes |0\rangle).\end{array} \tag{3}$$

Then we use the distributivity law for scalar product to pull scalars to the front of each summand. In this simple example, the operation changes nothing. But in more complex algorithms, we find this step necessary. To continue the reasoning, we use associativity laws to do matrix-vector multiplications from right to left. We also exploit the law in **L**13 to change a matrix product of two tensored terms into a tensor product of two matrix multiplications. To illustrate the idea, we show a few intermediate steps in simplifying the first summand in (3); the other summands are similar.

$$\begin{aligned}
&((I_2 \otimes B_0 \otimes I_2) \times (B_0 \otimes I_2 \otimes I_2) \times (H \otimes I_2 \otimes I_2) \times (|0\rangle \otimes |0\rangle \otimes |0\rangle)\\
&= ((I_2 \otimes (B_0 \otimes I_2)) \times (B_0 \otimes (I_2 \otimes I_2)) \times (H \otimes (I_2 \otimes I_2)) \times (|0\rangle \otimes (|0\rangle \otimes |0\rangle)))\\
&= (I_2 \times (B_0 \times (H \times |0\rangle))) \otimes (B0 \times (I_2 \times (I_2 \times |0\rangle))) \otimes (I_2 \times (I_2 \times (I_2 \times |0\rangle)))\\
&= \tfrac{1}{\sqrt{2}} \cdot (|0\rangle \otimes |0\rangle \otimes |0\rangle).
\end{aligned}$$

In the above reasoning, we have employed the laws established in `G_db` and `B_db` to rewrite terms. We also use scalar multiplications and the cancellation of zero matrices. The above steps appear a bit complex, but they can be fully automated in Coq, fortunately. The script for implementing the strategy `operate_reduce` is as follows:

```
Ltac operate_reduce :=
  autounfold with G2_db;
  distribute_plus;
  isolate_scale;
  assoc_right;
  repeat mult_kron;
  repeat autorewrite with G_db;
  reduce_scale.
```

In summary, using the strategy `operate_reduce`, we can simplify the first summand in (3) into $\frac{1}{\sqrt{2}} \cdot (|0\rangle \otimes |0\rangle \otimes |0\rangle)$. By the same strategy, the second and third summands turn out to be $\mathbf{0}$, and the fourth one becomes $\frac{1}{\sqrt{2}} \cdot (|1\rangle \otimes |1\rangle \otimes |1\rangle)$. As a result, we have formally proved the equation in (1).

Although it is very intuitive to represent pure quantum states by vectors, there is inconvenience. In quantum mechanics, the global phase of a qubit is often ignored. For example, we would not distinguish $|\psi\rangle$ from $e^{i\theta}|\psi\rangle$ for any θ. However, when written in vector form, $|\psi\rangle$ and $e^{i\theta}|\psi\rangle$ may be different because of the coefficient $e^{i\theta}$ present in the latter but not in the former. Therefore, we use the symbol $\approx$ to denote such an equivalence, i.e. $e^{i\theta}|\psi\rangle \approx |\psi\rangle$. As a matter of fact, we can be more general and define the equivalence for matrices, as given below.

```
Definition phase_equiv m n : nat (A B : Matrix m n) : Prop :=
  exists c : C, Cmod c = R1 /\  c .* A = B.
Infix "≈" := phase_equiv.
```

In the above definition, the condition `Cmod c = R1` means that the norm of the complex number `c` is one and `c .* A = B` says that the matrix `A` is equal to `B` after a scalar product with the coefficient `c`. See Section 6.1 for more concrete examples that use the relation $\approx$.

Note that if quantum states are represented by density matrices, we have

$$(e^{i\theta}|\psi\rangle)(e^{i\theta}|\psi\rangle)^\dagger \;=\; (e^{i\theta}|\psi\rangle)(e^{-i\theta}\langle\psi|) \;=\; |\psi\rangle\langle\psi|.$$

Therefore, the discrepancy entailed by the global phase disappears and the two vectors correspond to the same density matrix. Representing states by density matrices is a small but useful trick in formal verification of quantum circuits, which does not seem to have been exploited in the literature.

If the state of a quantum system is represented by a density matrix, the reasoning strategies discussed above can still be used. For instance, suppose a system is in the initial state given by density matrix ρ. After the execution of a quantum circuit implementing some unitary transformation U, the system changes into the new state $\rho' = U\rho U^\dagger$. Let $\rho = \sum_j p_j|j\rangle\langle j|$ be its spectral decomposition, where p_j are eigenvalues of ρ and the vectors $|j\rangle$ the corresponding eigenvectors. It follows that

$$\rho' \;=\; U(\sum_j p_j|j\rangle\langle j|)U^\dagger \;=\; \sum_j p_j U|j\rangle(U|j\rangle)^\dagger\,. \tag{4}$$

Therefore, we can first simplify $U|j\rangle$ into a vector, take its dual and then obtain ρ' easily. Our approach of symbolic reasoning also applies in this setting.

We define two functions `density` and `super` in advance. The former converts states in the vector form into corresponding states in the density matrix form. The latter formalizes the transformation process between states in the density matrix form.

```
Definition density n (φ : Matrix n 1) : Matrix n n := φ × φ†.
Definition super m n (M : Matrix m n) : Matrix n n -> Matrix m m :=
  fun ρ => M × ρ × M†.
```

We introduce the simplification strategy called `super_reduce` for states in the density matrix form.

```
Ltac super_reduce:=
  unfold super,density;                (* Expand super and density *)
  match goal with                      (* Match the pattern of target
  | |-context [ @Mmult ?n ?m ?n           with U × φ × φ† × U† *)
        (@Mmult ?n ?m ?m ?A ?B)
         (@adjoint ?n ?m ?A)] =>
    match B with
    | @Mmult ?m ?one ?m ?C
       (@adjoint ?m ?one ?C) =>
         transitivity (@Mmult n one n         (* Cast uniform types *)
           (@Mmult n m one A C) (@Mmult one m m
             (@adjoint m one C) (@adjoint n m A)))
     end
  end;
    [repeat rewrite <- Mmult_assoc; reflexivity| ..];
     rewrite <- Mmult_adjoint;                (* Extract adjoint *)
  let Hs := fresh "Hs" in
    match goal with
    | |-context [ @Mmult ?n ?o ?n
      (@Mmult ?n ?m ?o ?A ?B) (@adjoint ?m ?o ?C ) =
        @Mmult  ?n ?p ?n ?D (@adjoint ?n ?p ?D)] =>
      match C with
      | @Mmult ?n ?m ?o ?A ?B=>
        assert (@Mmult n m o A B = D) as Hs
      end                                   (* Use operate_reduce to
    end;                                       prove vector states
    [try reflexivity; try operate_reduce |     and rewrite it in
      repeat rewrite Hs; reflexivity].         density matrix form *)
```

In the above strategy, we first expand the `density` and `super` functions in the target. Next, we match the pattern of the target to see whether it is in the right form $U \times |\psi\rangle \times \langle\psi| \times U^\dagger$ (the middle of the equation in (4)) and cast uniform types, for the reasons to be discussed in Section 4 . Then we exploit the law in **L**14 to extract adjoint of multiplication terms so the target becomes $U \times |\psi\rangle \times (U \times |\psi\rangle)^\dagger$ (the right hand side of the equation in (4). Finally, we use the strategy `operate_reduce` to conduct the proof for states in vector form and rewrite it back in density matrix form.

4 Problems from Coq's type system and our solution

In principle, the Dirac notation is fully symbolic, i.e. no matter how we formalize it, the relevant laws and their proofs should remain unchanged. However, it turns out that different design choices in formalization do make a difference. Coq is a typed system. Thus, we should decide whether $2^{n+1} \times 2^{n+1}$ matrices and $2^{1+n} \times 2^{1+n}$ matrices, as two different Coq types, should be $\beta\eta$-reducible to each other.

From proof engineering point of view, the answer should be yes. These two kinds of matrices are mathematically the same object, and they should be used interchangeably. However, $(1 + n)$ and $(n + 1)$ are not $\beta\eta$-reducible to each other in Coq for a general variable n. Thus, we have to carefully define the Coq type of matrices so that those two kinds above are reducible to each other. We define matrices (no matter how large it is) to be a function from two natural numbers (column numbers and row numbers) to complex numbers. But still, there are two problems that need to be solved.

The elements outside the range of a matrix. We could choose to only reason about *well-formed* matrices whose "outside elements" are all zero. Rand et al. [10] heavily used this approach in their work. However, having well-formed matrices imposes a heavy burden for formal proofs because the condition of well-formedness needs to be checked each time we manipulate matrices. In our development, we define a relaxed notion of matrix equivalence, so that two matrices are deemed to be equivalent if they are equal component-wisely within the dimensions, and outside the dimensions the corresponding elements can be different. With a slight abuse of notation, we still use the symbol $=$ to denote the newly defined matrix equivalence[1], and prove its elementary properties about scale product, matrix product, matrix addition, tensor product and conjugate transpose. Reasoning about matrices modulo that equivalence turns out to be convenient in Coq. Specifically, the automation of the rewriting strategies mentioned above does not require side condition proofs about well-formedness.

Coq type casting. In math, $|0\rangle \otimes |0\rangle$ is a 4×1 matrix and it is only verbose to say it is a $(2 \cdot 2) \times (1 \cdot 1)$ matrix. Even though $(2 \cdot 2) \times (1 \cdot 1)$ is convertible to 4×1, the

[1] Nevertheless, we keep our Coq script in the repository at Github more rigid. There we use $\equiv$ to stand for the relaxed notion of matrix equivalence and reserve $=$ for the stronger notion of equivalence in the sense that $A = B$ means the two matrices A and B are equal component-wisely both within and outside their dimensions.

two typing claims are different in Coq and this difference is significant in rewriting. For example, the associativity of multiplication says:

```
forall m n o p (A: matrix m n) (B: matrix n o) (C: matrix o p),
@Mmult m n p A (@Mmult n o p B C) = @Mmult m o p (@Mmult m n o A B) C.
```

However, rewriting does not work in the following case:

```
@Mmult 1 1 1 A (@Mmult (1*1) (1*1) (1*1) B C)
```

because rewriting uses an exact syntax match but not unification. This problem of type mismatch often occurs after we use the law **L**13 for rewriting. We choose to build a customized rewrite tactic to overcome this problem. Using the example above, we want to rewrite via the associativity of multiplication. We first do a pattern matching for expressions of the form

```
@Mmult ?m ?n1 ?p1 ?A (@Mmult ?n2 ?o ?p2 ?B ?C)
```

no matter whether `n1` and `p1` coincide with `n2` and `p2`. We then use Coq's built-in unification to unify `n1`, `p1` with `n2`, `p2`. This unification must succeed or else the original expression of matrix computation is not well-formed. After the expression is changed to

```
@Mmult ?m ?n1 ?p1 ?A (@Mmult ?n1 ?o ?p1 ?B ?C)
```

we can use Coq's original rewrite tactic via the associativity of multiplication.

We handle the above mentioned type problems silently and whoever uses our system to formalize his/her own proof will not even feel these problems.

5 Circuit equivalence

In order to judge whether two circuits have the same behavior, we need to formulate reasonable notions of circuit equivalence. We will propose two candidate relations: one is called matrix equivalence and the other operator equivalence.

5.1 Matrix equivalence

A natural way of interpreting a quantum circuit without measurements is to consider each quantum gate as a unitary matrix and the whole circuit as a composition of matrices that eventually reduces to a single matrix. From this viewpoint, two circuits are equivalent if they denote the same unitary matrix, that is, matrix equivalence $=$ suffices to stand for circuit equivalence.

Directly showing that two matrices are equivalent requires to inspect their elements and compare them component-wisely. Instead, we can take a functional view of matrix equivalence. Let A, B be two matrices, then $A = B$ if and only if $A|v\rangle = B|v\rangle$ for any vector $|v\rangle$.

```
Lemma MatrixEquiv_spec: forall n (A B: Matrix n n),
  A = B <-> (forall v: Vector n, A × v = B × v).
```

In Figure 2 we list some laws that are often useful in simplifying circuits before showing that they are equivalent. Let us verify the validity of the laws. Take the first one as an example. Its validity is stated in Lemma `unit_X`. In order to prove that lemma, we apply `MatrixEquiv_spec` and reduce it to Lemma `unit_X'`, which can be easily proved by the strategy `operate_reduce`.

$$
\begin{array}{rcl@{\qquad}rcl}
XX & = & I_2 & \frac{1}{\sqrt{2}} \cdot (X+Z) & = & H \\
YY & = & I_2 & H_2 \times CX \times H_2 & = & CZ \\
ZZ & = & I_2 & CX \times X_1 \times CX & = & X_1 \times X_2 \\
HH & = & I_2 & CX \times Y_1 \times CX & = & Y_1 \times X_2 \\
CX \times CX & = & I_4 & CX \times Z_1 \times CX & = & Z_1 \\
HXH & = & Z & CX \times X_2 \times CX & = & X_2 \\
HYH & = & -Y & CX \times Y_2 \times CX & = & Z_1 \times Y_2 \\
HZH & = & X & CX \times Z_2 \times CX & = & Z_1 \times Z_2
\end{array}
$$

Fig. 2 More laws

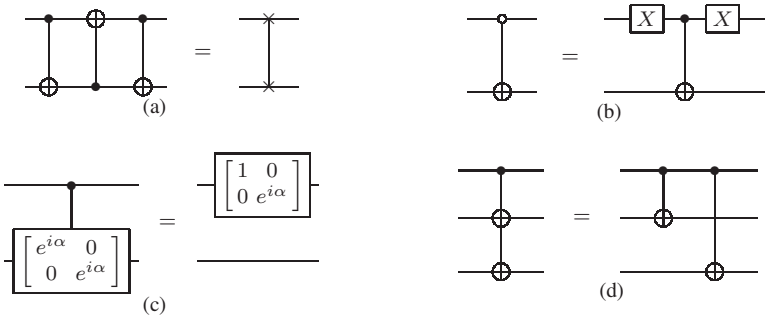


Fig. 3 Some equivalent circuits

```
Lemma unit_X : X × X = I_2.
Lemma unit_X' : forall v : Vector 2,  X × X × v = I_2 × v.
```

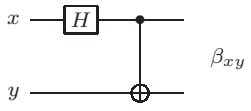
In the right column of Figure 2, the subscripts of X, Y, Z and H indicate on which qubits the quantum gates are applied. For example, X_2 means that the Pauli-X gate is applied on the second qubit. Thus, the operation $Y_1 \times X_2$ actually stands for $(Y \otimes I_2) \times (I_2 \otimes X)$.

In Figure 3, we display some equivalent circuits. In diagram (a), on the right of = is a schematic specification of swapping two qubits, which is implemented by the circuit on the left. In diagram (b), there is a controlled operation performed on the second qubit, conditioned on the first qubit being set to zero. It is equivalent to a CX gate enclosed by two Pauli-X gates on the first qubit. In diagram (c), the controlled phase shift gate on the left is equivalent to a circuit for two qubits on the right. In diagram (d), a controlled gate with two targets is equivalent to the concatenation of two CX gates. They are formalized as follows and all can be proved by using the strategy `operate_reduce` in conjunction with `MatrixEquiv_spec`.

```
Definition SWAP :=  B0 ⊗ B0 .+ B1 ⊗ B2 .+ B2 ⊗ B1 .+ B3 ⊗ B3.
Definition XC :=  X ⊗ B3 .+ I_2 ⊗ B0.
Lemma Eq1 : SWAP =  CX × XC × CX.

Definition not_CX := B0 ⊗ X .+ B3 ⊗ I_2.
Lemma Eq2 : not_CX = (X ⊗ I_2) × CX × (X ⊗ I_2).

Definition CE (u: R) := B0 ⊗ I_2 .+ B3 ⊗ (Cexp u .* B0 .+ Cexp u .* B3).
Lemma Eq3 : CE u = (B0 .+ Cexp u .* B3) ⊗ I_2.
```



$$|\beta_{00}\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle \otimes |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle \otimes |1\rangle$$
$$|\beta_{01}\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle \otimes |1\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle \otimes |0\rangle$$
$$|\beta_{10}\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle \otimes |0\rangle - \frac{1}{\sqrt{2}} \cdot |1\rangle \otimes |1\rangle$$
$$|\beta_{11}\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle \otimes |1\rangle - \frac{1}{\sqrt{2}} \cdot |1\rangle \otimes |0\rangle$$

Fig. 4 The Bell states

```
Definition CXX := B0 ⊗ I_2 ⊗ I_2 .+ B3 ⊗ X ⊗ X.
Definition CIX := B0 ⊗ I_2 ⊗ I_2 .+ B3 ⊗ I_2 ⊗ X.
Lemma Eq4 : CXX = CIX × (CX ⊗ I_2).
```

In Section 3 we have formalized the preparation of the 3-qubit GHZ state (cf. Figure 1). Now let us have a look at the Bell states. Depending on the input states, the circuit in Figure 4 gives four possible output states. The correctness of the circuit is validated by the four lemmas below, where the states are given in terms of density matrices and the circuit is described by a super-operator. It is easy to to prove them by using our strategy `super_reduce`.

```
Definition b00 := /√2 .* (|0,0⟩) .+ /√2 .* (|1,1⟩).
Definition b01 := /√2 .* (|0,1⟩) .+ /√2 .* (|1,0⟩).
Definition b10 := /√2 .* (|0,0⟩) .+ (-/√2) .* (|1,1⟩).
Definition b11 := /√2 .* (|0,1⟩) .+ (-/√2) .* (|1,0⟩).

Lemma pb00 : super (CX × (H ⊗ I_2)) (density |0,0⟩) = density b00.
Lemma pb01 : super (CX × (H ⊗ I_2)) (density |0,1⟩) = density b01.
Lemma pb10 : super (CX × (H ⊗ I_2)) (density |1,0⟩) = density b10.
Lemma pb11 : super (CX × (H ⊗ I_2)) (density |1,1⟩) = density b11.
```

5.2 Operator equivalence

An alternative and abstract way of interpreting a quantum circuit without measurements is to consider it as an operator that changes input quantum states to output ones. Therefore, we present a notion of operator equivalence used for circuit equivalence.

Formally, we define an operator for n-qubits as a square matrix of dimension 2^n and a state (in the vector form) for n-qubits as a 2^n-dimensional vector. Applying an operator to a state yields another state. The relations operator equivalence `OperatorEquiv` and state equivalence `StateEquiv` are then defined in terms of the notion of matrix equivalence $\approx$ introduced in Section 3, because global phases are ignored as far as quantum states are concerned.

```
Definition Operator n := Matrix (Nat.pow 2 n) (Nat.pow 2 n).
Definition State n := Matrix (Nat.pow 2 n) 1.
Definition Apply n (o: Operator n) (s: State n): State n :=
  @Mmult (Nat.pow 2 n) (Nat.pow 2 n) 1 o s.
Definition OperatorEquiv n (o1 o2: Operator n): Prop :=
  @phase_equiv (Nat.pow 2 n) (Nat.pow 2 n) o1 o2.
Definition StateEquiv n (s1 s2: State n): Prop :=
  @phase_equiv (Nat.pow 2 n) 1 s1 s2.
```

The following lemma provides a functional view of operator equivalence. It is a counterpart of `MatrixEquiv_spec`. Let A, B be two operators, we have $A \approx B$ if and only if $A|\psi\rangle \approx B|\psi\rangle$ for any state $|\psi\rangle$.

```
Lemma OperatorEquiv_spec: forall n (o1 o2: Operator n),
  OperatorEquiv o1 o2 <->
  (forall s: State n, StateEquiv (Apply o1 s) (Apply o2 s)).
```

Furthermore, two states are equivalent or equal modulo a global phase, i.e. $|\psi\rangle \approx |\phi\rangle$ if and only if their density matrices are exactly the same, i.e. $|\psi\rangle\langle\psi| = |\phi\rangle\langle\phi|$.

```
Lemma StateEquiv_spec: forall n (s1 s2: State n),
  StateEquiv s1 s2 <->
    @Mmult (Nat.pow 2 n) 1 (Nat.pow 2 n) s1 (s1 †) =
    @Mmult (Nat.pow 2 n) 1 (Nat.pow 2 n) s2 (s2 †).
```

Although both matrix equivalence $=$ and operator equivalence $\approx$ can be used for circuit equivalences, the former is strictly finer than the latter. Therefore, in the rest of the paper we relate circuits by $=$ whenever possible, as they are also related by $\approx$. Moreover, it is not difficult to see that $=$ is a congruence relation. For example, if A, B are two quantum gates and $A = B$, then we can add a control qubit to form controlled-A and controlled-B gates, which are still identified by $=$. However, the relation $\approx$ does not satisfy such kind of congruence property. In Section 6.1 we will see a concrete example of using $\approx$, where quantum states are identified by purposefully ignoring their global phases.

6 Case studies

To illustrate the power of our symbolic approach of reasoning about quantum circuits, we conduct a few case studies and compare the approach with the computational one in [10].

6.1 Deutsch's algorithm

Given a Boolean function $f : \{0,1\} \to \{0,1\}$, Deutsch [4] presented a quantum algorithm that can compute $f(0) \oplus f(1)$ in a single evaluation of f. The algorithm can tell whether $f(0)$ equals $f(1)$ or not, without giving any information about the two values individually. The quantum circuit in Figure 5 gives an implementation of the algorithm. It makes use of a quantum oracle that maps any state $|x\rangle \otimes |y\rangle$ to the state $|x\rangle \otimes |y \oplus f(x)\rangle$, where $x, y \in \{0,1\}$. More specifically, the unitary operator U_f can be in one of the following four forms:

- if $f(0) = f(1) = 0$, then $U_f = U_{f00} = I_2 \otimes I_2$;
- if $f(1) = f(1) = 1$, then $U_f = U_{f11} = I_2 \otimes X$;
- if $f(0) = 0$ and $f(1) = 1$, then $U_f = U_{f01} = CX$;
- if $f(0) = 1$ and $f(1) = 0$, then $U_f = U_{f10} = B_0 \otimes X + B_3 \otimes I_2$.

We formalize Deutsch's algorithm in Coq and use our symbolic approach to prove its correctness. Let us suppose that $|\psi_0\rangle = |01\rangle$ is the input state. There are three phases in this quantum circuit. The first phase applies the Hadamard gate to each of the two qubits. So we define the initial state and express the state after the first phase as follows:

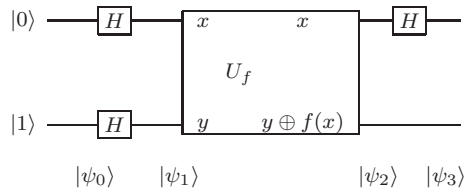


Fig. 5 Deutsch's algorithm

```
Definition ψ0 := |0⟩ ⊗ |1⟩.
Definition ψ1 := (H ⊗ H) × ψ0.
Lemma step1 : ψ1 = |+⟩ ⊗ |-⟩.
```

Lemma `step1` claims that the intermediate state after the first phase is $|+\rangle \otimes |-\rangle$. We can use the strategy `operate_reduce` designed in Section 3 to prove its correctness.

The second phase applies the unitary operator U_f to $|\psi_1\rangle$. Since U_f has four possible forms, we consider four cases.

```
Definition ψ20 := (I_2 ⊗ I_2) × ψ1.
Definition ψ21 := (I_2 ⊗ X) × ψ1.
Definition ψ22 := CX × ψ1.
Definition ψ23 := (B0 ⊗ X .+ B3 ⊗ I_2) × ψ1.

Lemma step20 : ψ20 = |+⟩ ⊗ |-⟩.
Lemma step21 : ψ21 = -1 .* |+⟩ ⊗ |-⟩.
Lemma step22 : ψ22 = |-⟩ ⊗ |-⟩.
Lemma step23 : ψ23 = -1 .* |-⟩ ⊗ |-⟩.
```

Each of the above four lemmas corresponds to one case. They claim that the intermediate state $|\psi_2\rangle$ is $\pm 1 \cdot |+\rangle \otimes |-\rangle$ after the second phase when $f(0) = f(1)$, and $\pm 1 \cdot |-\rangle \otimes |-\rangle$ when $f(0) \neq f(1)$. We prove the four lemmas by rewriting $|\psi_1\rangle$ with Lemma `step1` and using the strategy `operate_reduce` again.

The last phase applies the Hadamard gate to the first qubit of $|\psi_2\rangle$. So we still have four cases.

```
Definition ψ30 := (H ⊗ I_2) × ψ20.
...
Lemma step30 : ψ30 = |0⟩ ⊗ |-⟩.
Lemma step31 : ψ31 = -1 .* |0⟩ ⊗ |-⟩.
Lemma step32 : ψ32 = |1⟩ ⊗ |-⟩.
Lemma step33 : ψ33 = -1 .* |1⟩ ⊗ |-⟩.
```

Observe that the only difference between $|\psi_{30}\rangle$ and $|\psi_{31}\rangle$ lies in the global phase -1, which can be ignored. Similarly for $|\psi_{32}\rangle$ and $|\psi_{33}\rangle$. Formally, we can prove the following lemmas.

```
Lemma step31' : ψ31 ≈ |0⟩ ⊗ |-⟩.
Lemma step33' : ψ33 ≈ |1⟩ ⊗ |-⟩.
```

Therefore, after the last phase, we have $|\psi_3\rangle = |0\rangle \otimes |-\rangle$ when $f(0) = f(1)$, and $|\psi_3\rangle = |1\rangle \otimes |-\rangle$ when $f(0) \neq f(1)$. This is proved by using the intermediate results obtained in the first two phases and the strategy `operate_reduce`.

The above reasoning about the Deutsch's algorithm proceeds step by step and shows all the intermediate states in each phase. Alternatively, one may be only interested in the output state of the circuit once an input state is fed. In other words, we would like to show a property like

$$|\psi_{3ij}\rangle = (H \otimes I_2) \times U_{fij} \times (H \otimes H) \times |\psi_0\rangle.$$

Formally, we need to prove four equations depending on the forms of U_f.

```
Lemma deutsch00 :
(H ⊗ I_2) × (I_2 ⊗ I_2) × (H ⊗ H) × (|0⟩ ⊗ |1⟩) = |0⟩ ⊗ |-⟩ .
Lemma deutsch01 :
(H ⊗ I_2) × (I_2 ⊗ X) × (H ⊗ H) × (|0⟩ ⊗ |1⟩) = -1 .* |0⟩ ⊗ |-⟩ .
Lemma deutsch10 :
(H ⊗ I_2) × CX × (H ⊗ H) × (|0⟩ ⊗ |1⟩ = |1⟩ ⊗ |-⟩ .
Lemma deutsch11 :
(H ⊗ I_2) × (B0 ⊗ X .+ B3 ⊗ I_2) × (H ⊗ H) × (|0⟩ ⊗ |1⟩) = -1 .* |1⟩ ⊗ |-⟩.
```

The second and fourth equations can be written in a simpler form as follows.

```
Lemma deutsch01' :
(H ⊗ I_2) × (I_2 ⊗ X) × (H ⊗ H) × (|0⟩ ⊗ |1⟩) ≈ |0⟩ ⊗ |-⟩ .
Lemma deutsch11' :
(H ⊗ I_2) × (B0 ⊗ X .+ B3 ⊗ I_2) × (H ⊗ H) × (|0⟩ ⊗ |1⟩) ≈ |1⟩ ⊗ |-⟩ .
```

Using our symbolic reasoning, these lemmas can be easily proved. Thus, we know that the first qubit of the result state is $|0\rangle$ when $f(0) = f(1)$, and $|1\rangle$ otherwise. That is, $|\psi_{3ij}\rangle = |f(0) \oplus f(1)\rangle|-\rangle$ as expected.

In the above reasoning, states are described in vectors. We can also obtain a similar result if states are written in density matrix form.

6.2 Teleportation

Quantum teleportation [1] is one of the most important protocols in quantum information theory. It teleports an unknown quantum state by only sending classical information, by making use of a maximally entangled state.

Let the sender and the receiver be $Alice$ and Bob, respectively. The quantum teleportation protocol goes as follows, as illustrated by the quantum circuit in Figure 6.

1. $Alice$ and Bob prepare an EPR state $|\beta_{00}\rangle_{q_2,q_3}$ together. Then they share the qubits, $Alice$ holding q_2 and Bob holding q_3.
2. To transmit the state $|\psi\rangle$ of the quantum qubit q_1, $Alice$ applies a CX operation on q_1 and q_2 followed by an H operation on q_1.
3. $Alice$ measures q_1 and q_2 and sends the outcome x to Bob.
4. When Bob receives x, he applies appropriate Pauli gates on his qubit q_3 to recover the original state $|\psi\rangle$ of q_1.

We formalize the quantum teleportation algorithm in Coq and use our symbolic approach to prove its correctness. Let $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ be any vector used as a part of the input state. The other part is $|\beta_{00}\rangle$, which needs extra preparation. For simplicity, we directly represent $|\beta_{00}\rangle$ with a combination of $|0\rangle$ and $|1\rangle$. So we define the input state $|\psi_0\rangle$ in Coq as follows.

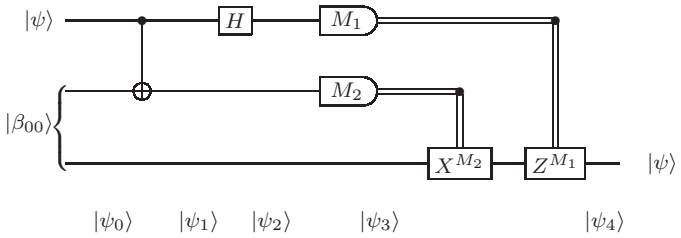


Fig. 6 Teleportation [2]

```
Definition ψ (a b : C) : Vector 2 := a .* |0⟩ .+ b .* |1⟩.
Definition bell00 := /√2 .* (|0⟩ ⊗ |0⟩) .+ /√2 .* (|1⟩ ⊗ |1⟩).
Definition ψ0 := ψ a b ⊗ bell00.
```

The input state goes through the quantum circuit that comprises four phases. We define the quantum states immediately after phases 2, 3 and 4 as follows.

$$\begin{aligned}|\psi_2\rangle &:= (H \otimes I_2 \otimes I_2) \times (CX \otimes I_2) \times (|\psi\rangle \otimes |\beta_{00}\rangle)\\ |\psi_{3ij}\rangle &:= (N_i \otimes N_j \otimes I_2) \times |\psi_2\rangle\\ |\psi_{4ij}\rangle &:= (I_2 \otimes I_2 \otimes Z^i) \times (I_2 \otimes I_2 \otimes X^j) \times |\psi_{3ij}\rangle\end{aligned}$$

In the third phase, due to the measurement with measurement operators $\{N_0, N_1\}$, where $N_0 = B_0$ and $N_1 = B_3$, there are four possible cases for the state $|\psi_3\rangle$, and the probability for each case can be calculated as

$$\langle\psi_2| \times (N_i \otimes N_j \otimes I_2)^\dagger \times (N_i \otimes N_j \otimes I_2) \times |\psi_2\rangle \;=\; 1/4.$$

Let $i, j \in \{0, 1\}$ be the measurement outcomes for the top two qubits. The quantum state after the fourth phase becomes $|\psi_{4ij}\rangle$. Using the simplification strategies discussed in Section 3, it is easy to prove that $|\psi_{4ij}\rangle$ can be simplified to be $\frac{1}{2}|i\rangle \otimes |j\rangle \otimes |\psi\rangle$, which is the correct final state without normalization.

Instead of a step-by-step reasoning, we may be only concerned with the final state after the circuit and would like to show the following equality

$$\begin{aligned}|\psi_{4ij}\rangle = (I_2 \otimes I_2 \otimes Z^i) \times (I_2 \otimes I_2 \otimes X^j) \times (N_i \otimes N_j \otimes I_2) \times\\ (H \otimes I_2 \otimes I_2) \times (CX \otimes I_2) \times (|\psi\rangle \otimes |\beta_{00}\rangle).\end{aligned}$$

In the formal proof, we have four cases to consider. They correspond to the four lemmas below.

```
Lemma tele00 :  forall (a b : C),
  (N0 ⊗ N0 ⊗ I_2) × (H ⊗ I_2 ⊗ I_2) × (CX ⊗ I_2) × (ψ a b ⊗ bell00)
  = / 2 .* |0⟩ ⊗ |0⟩ ⊗ (ψ a b) .
Lemma tele01 :  forall (a b : C),
  (I_2 ⊗ I_2 ⊗ X) × (N0 ⊗ N1 ⊗ I_2) × (H ⊗ I_2 ⊗ I_2) × (CX ⊗ I_2)
  × (ψ a b ⊗ bell00) = / 2 .* |0⟩ ⊗ |1⟩ ⊗ (ψ a b) .
Lemma tele10 :  forall (a b : C),
  (I_2 ⊗ I_2 ⊗ Z) × (N1 ⊗ N0 ⊗ I_2) × (H ⊗ I_2 ⊗ I_2) × (CX ⊗ I_2)
  × (ψ a b ⊗ bell00) = / 2 .* |1⟩ ⊗ |0⟩ ⊗ (ψ a b) .
Lemma tele11 :  forall (a b : C),
  (I_2 ⊗ I_2 ⊗ Z) × (I_2 ⊗ I_2 ⊗ X) × (N1 ⊗ N1 ⊗ I_2)
  × (H ⊗ I_2 ⊗ I_2) × (CX ⊗ I_2) × (ψ a b ⊗ bell00)
  = / 2 .* |1⟩ ⊗ |1⟩ ⊗ (ψ a b) .
```

The above lemmas can be quickly proved using our symbolic approach. They show that the third qubit of the result state is always equal to $|\psi\rangle$, the state to be teleported from Alice to Bob.

If the input state is given by a density matrix as follows, we can also prove that the final output state of the circuit is in a correct form. The lemma below is a counterpart of Lemma `tele00`, with states in density matrices instead of vectors.

```
Lemma Dtele00 :  forall (a b : C),
  super ((N0 ⊗ N0 ⊗ I_2) × (H ⊗ I_2 ⊗ I_2) × (CX ⊗ I_2))
  (density (ψ a b ⊗ bell00)) = density (/ 2 .* |0⟩ ⊗ |0⟩ ⊗ (ψ a b)) .
```

6.3 Simon's Algorithm

The Simon's problem was raised in 1994 [28]. Although it is an artificial problem, it inspired Shor to discover a polynomial time algorithm to solve the integer factorization problem.

Given a function $f : \{0,1\}^n \to \{0,1\}^n$, suppose there exists a string $s \in \{0,1\}^n$ such that the following property is satisfied:

$$f(x) = f(y) \;\Leftrightarrow\; x = y \text{ or } x \oplus y = s \tag{5}$$

for all $x, y \in \{0,1\}^n$. Here $\oplus$ is the bit-wise modulo 2 addition of two n bit-strings. The goal of Simon's algorithm is to find the string s. The algorithm consists of iterating the quantum circuit and then performing some classical post-processing.

1. Set an initial state $|0^n\rangle \otimes |0^n\rangle$, and apply Hadamard gates to the first n qubits respectively.
2. Apply an oracle U_f to all the $2n$ qubits, where $U_f : |x\rangle|y\rangle \mapsto |x\rangle|f(x) \oplus y\rangle$.
3. Apply Hadamard gates to the first n qubits again and then measure them.

When $s \neq 0^n$, the probability of obtaining each string $y \in \{0,1\}^n$ is

$$p_y = \begin{cases} 2^{-(n-1)} & \text{if} \quad s \cdot y = 0 \\ 0 & \text{if} \quad s \cdot y = 1. \end{cases}$$

Therefore, it can be seen that the result string y must satisfy $s \cdot y = 0$ and be evenly distributed. Repeating this process $n-1$ times, we will get $n-1$ strings $y_1, \cdots, y_{n-1}$ so that $y_i \cdot s = 0$ for $1 \leq i \leq n-1$. Thus we have $n-1$ linear equations with n unknowns (n is the number of bits in s). The goal is to solve this system of equations to get s. We can get a unique non-zero solution s if we are lucky and $y_1, ..., y_{n-1}$ are linearly independent. Otherwise, we repeat the entire process and will find a linearly independent set with a high probability.

As an example, we consider the Simon's algorithm with $n = 2$. The quantum circuit is displayed in Figure 7. We design the oracle as the gates in the dotted box $U_f = (I_2 \otimes CX \otimes I_2) \times (CIX \otimes X)$, where the gate CIX is defined in page 13. For this oracle, $s = 11$ satisfies property (5). The change of states can be seen as follows:

$$\begin{aligned} |0000\rangle &\xrightarrow{H\otimes H\otimes I_2\otimes I_2} |++\rangle|00\rangle \\ &\xrightarrow{U_f} \tfrac{1}{2}[(|00\rangle + |11\rangle)|01\rangle + (|01\rangle + |10\rangle)|11\rangle] \\ &\xrightarrow{H\otimes H\otimes I_2\otimes I_2} \tfrac{1}{2}[(|00\rangle + |11\rangle)|01\rangle + (|00\rangle - |11\rangle)|11\rangle] \end{aligned}$$

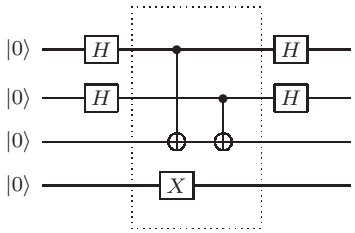


Fig. 7 Simon's algorithm with $n = 2$ and $s = 11$

We can establish the following lemma with our symbolic approach:

```
Lemma simon : super ((H ⊗ H ⊗ I_2 ⊗ I_2) × (I_2 ⊗ CX ⊗ I_2) ×
(CIX ⊗ X) × (H ⊗ H ⊗ I_2 ⊗ I_2)) (density |0,0,0,0⟩) = density
(/2 .* |0,0,0,1⟩ .+ /2 .* |1,1,0,1⟩ .+ /2 .* |0,0,1,1⟩ .+ -/2 .* |1,1,1,1⟩).
```

We analyze the cases where the last two qubits are in the state $|01\rangle$ or $|11\rangle$. The corresponding first two qubits are in $|00\rangle$ or $|11\rangle$, each occurs with equal probability. By property (5), it means that $x \oplus y = 00$ or 11, so we obtain that $s = 11$.

6.4 Grover's algorithm

In this section we consider Grover's search algorithm . The algorithm starts from the initial state $|0\rangle^{\otimes n}$. It first uses $H^{\otimes n}$ (the H gate applied to each of the n qubits) to obtain a uniform superposition state, and then applies the Grover iteration repeatedly. An implementation of the Grover iteration has four steps:

1. Apply the oracle O.
2. Apply the Hadamard transform $H^{\otimes n}$.
3. Perform a conditional phase shift on $|x\rangle$, if $|x\rangle \neq |0\rangle$.
4. Apply the Hadamard transform $H^{\otimes n}$ again.

Here the conditional phase-shift unitary operator in the third step is $2|0\rangle\langle 0| - I$. We can merge the last three steps as follows:

$$H^{\otimes n} \times (2|0\rangle\langle 0| - I) \times H^{\otimes n} = 2|\phi\rangle\langle\phi| - I$$

where $|\phi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$, where $N = 2^n$. Therefore, the Grover iteration becomes $G = (2|\phi\rangle\langle\phi| - I) \times O$.

As a concrete example, we consider the Grover's algorithm with two qubits. The size of the search space of this algorithm is four. So we need to consider four search cases with $x^* = 0, 1, 2, 3$. The oracle must satisfy that if $x = x^*$, then $f(x^*) = 1$, otherwise $f(x) = 0$. So in accordance with $x^* = 0, 1, 2, 3$, we design four oracles $ORA_0, ..., ORA_3$, which are implemented by the four circuits in Figure 8.

```
Definition ORA0 := B0 ⊗ (B0 ⊗ X .+ B3 ⊗ I_2) .+ B3 ⊗ I_2 ⊗ I_2.
Definition ORA1 := B0 ⊗ CX .+ B3 ⊗ I_2 ⊗ I_2.
Definition ORA2 := B0 ⊗ I_2 ⊗ I_2 .+ B3 ⊗ (B0 ⊗ X .+ B3 ⊗ I_2).
Definition ORA3 := B0 ⊗ I_2 ⊗ I_2 .+ B3 ⊗ CX.
```

(a) ORA_0 (b) ORA_1 (c) ORA_2 (d) ORA_3

Fig. 8 The quantum circuit of different Oracle

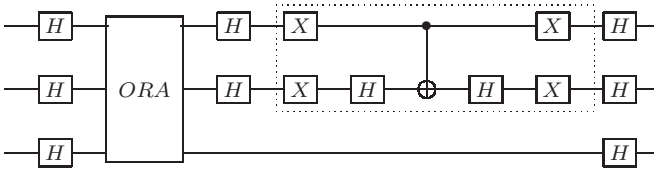


Fig. 9 the Grover's algorithm with two qubits

	Deutsch	Simon	Teleportation	Secret sharing	QFT	Grover
Symbolic	2860	36560	40712	58643	34710	363160
Computational	25190	183230	46450	168680	68730	966140

Table 2 Comparison of two approaches with verification time in milliseconds

The whole algorithm is illustrated by the circuit in Figure 9. The gates in the dotted box perform the conditional phase shift operation $2|0\rangle\langle 0| - I$. We then merge the front and back $H \otimes H$ gates to it and get the operation CPS as follows.

```
Definition MI := (B0 .+ B1 .+ B2 .+ B3) ⊗ (B0 .+ B1 .+ B2 .+ B3).
Definition CPS := (((/2 .* MI) .+ (-1) .* (I_2 ⊗ I_2)) ⊗ I_2).
```

So we have the Grover iteration $G = (2|\phi\rangle\langle\phi| - I) \times O = CPS \times ORA_i$. Let the initial state be $|0\rangle \otimes |0\rangle \otimes |1\rangle$. After the Hadamard transform $H^{\otimes 3}$, we only perform Grover iteration once to get the search solution. In summary, we formalize the Grover's algorithm with two qubits in the vector form as follows, and use our symbolic approach to prove them. The reasoning using density matrices can also be done.

```
Lemma Gro0:
  (I_2 ⊗ I_2 ⊗ H) × CPS × ORA0 × (H ⊗ H ⊗ H) × |0,0,1⟩ = |0,0,1⟩.
Lemma Gro1:
  (I_2 ⊗ I_2 ⊗ H) × CPS × ORA1 × (H ⊗ H ⊗ H) × |0,0,1⟩ = |0,1,1⟩.
Lemma Gro2:
  (I_2 ⊗ I_2 ⊗ H) × CPS × ORA2 × (H ⊗ H ⊗ H) × |0,0,1⟩ = |1,0,1⟩.
Lemma Gro3:
  (I_2 ⊗ I_2 ⊗ H) × CPS × ORA3 × (H ⊗ H ⊗ H) × |0,0,1⟩ = |1,1,1⟩.
```

6.5 Experiments

We have conducted experiments on Deutsch's algorithm, Simon's algorithm, quantum teleportation, quantum secret sharing protocol, quantum Fourier transform (QFT)

with three qubits, and Grover's search algorithm with two qubits. In Table 2, we record the execution time of those examples in milliseconds in CoqIDE 8.10.0 running in a PC with Intel Core i5-7200 CPU and 8 GB RAM. As we can see in the table, our symbolic approach always outperforms the computational one in [10].

The computational approach is slow because of the explicit representation of matrices and inefficient tactics for evaluating matrix multiplications. Let us consider a simple example. In the computational approach, the Hadamard gate H is defined by `ha` below:

```
Definition ha : Matrix 2 2 :=
  fun x y => match x, y with
    | 0, 0 => (1 / √2)
    | 0, 1 => (1 / √2)
    | 1, 0 => (1 / √2)
    | 1, 1 => -(1 / √2)
    | _, _ => 0
  end.
```

Since H is unitary, we have $HH = I$ and the following property becomes straightforward.

```
Lemma H3_ket0: (ha ⊗ ha ⊗ ha) × (ha ⊗ ha ⊗ ha) × (|0,0,0⟩) = (|0,0,0⟩).
```

However, to prove the above lemma with the computational approach is far from being trivial. To see this, we literally go through a few steps. Firstly, we apply the associativity of matrix multiplication on the left hand side of the equation so to rewrite it into

$$(H \otimes H \otimes H) \times ((H \otimes H \otimes H) \times (|0\rangle \otimes |0\rangle \otimes |0\rangle)).$$

Secondly, each explicitly represented matrix is converted into a two-dimensional list and matrix multiplications are calculated in order. Finally, we need to show that each of the eight elements in the vector on the left is equal to the corresponding element on the right. Let $A_0 = (H \otimes H \otimes H) \times (|0\rangle \otimes |0\rangle \otimes |0\rangle)$ and $A_1 = (H \otimes H \otimes H) \times A_0$. With the computational approach, obvious simplifications such as multiplication and addition with 0 and 1 are carried out for the elements in A_0 and A_1, and no more complicated simplification is effectively handled. So A_0 is a two-dimensional list with each element in the form $\frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}}$ and A_1 is a two-dimensional list whose first element is

$$\begin{aligned}
&(\tfrac{1}{\sqrt{2}} \times \tfrac{1}{\sqrt{2}} \times \tfrac{1}{\sqrt{2}} \times (\tfrac{1}{\sqrt{2}} \times \tfrac{1}{\sqrt{2}} \times \tfrac{1}{\sqrt{2}}))\\
&+(\tfrac{1}{\sqrt{2}} \times \tfrac{1}{\sqrt{2}} \times \tfrac{1}{\sqrt{2}} \times (\tfrac{1}{\sqrt{2}} \times \tfrac{1}{\sqrt{2}} \times \tfrac{1}{\sqrt{2}}))\\
&+ \ldots\\
&+(\tfrac{1}{\sqrt{2}} \times \tfrac{1}{\sqrt{2}} \times \tfrac{1}{\sqrt{2}} \times (\tfrac{1}{\sqrt{2}} \times \tfrac{1}{\sqrt{2}} \times \tfrac{1}{\sqrt{2}})),
\end{aligned}$$

which is a summation of eight identical summands with $\frac{1}{\sqrt{2}}$ multiplied with itself six times; other elements are in similar forms. From this simple example, we can already see that the explicit matrix representation and ineffective simplification in matrix multiplication make the intermediate expressions very cumbersome.

On the contrary, in the symbolic approach we have

$$\begin{aligned}
A_1 &= (H \otimes H \otimes H) \times (H \otimes H \otimes H) \times (|0\rangle \otimes |0\rangle \otimes |0\rangle)\\
&= (H \times H \times |0\rangle) \otimes (H \times H \times |0\rangle) \otimes (H \times H \times |0\rangle)\\
&= (H \times |+\rangle) \otimes (H \times |+\rangle) \otimes (H \times |+\rangle)\\
&= |0\rangle \otimes |0\rangle \otimes |0\rangle.
\end{aligned}$$

Notice that here we have kept the structure of tensor products rather than to eliminate them. In fact, we lazily evaluate tensor products because they are expensive to calculate and preserving more higher-level structures opens more opportunities for rewriting. The symbolic reasoning not only renders the intermediate expressions more readable, but also greatly reduces the time cost of arithmetic calculations.

In general, in the computational approach a multiplication of two $N \times N$ matrices of $O(k)$-length expressions results in a matrix of $O(Nk)$-length expressions, and those expressions are not effectively simplified. At the end of the computation, a matrix of $O(N^m)$-length expressions is obtained if $m + 1$ matrices of size $N \times N$ are multiplied together, which takes exponential time to simplify. In our approach, we represent matrices symbolically and simplify intermediate expressions effectively on the fly, which has a much better performance.

7 Related work

Formal verification in quantum computing has been growing rapidly, especially in Coq. Boender et al. [11] presented a framework for modeling and analyzing quantum protocols using Coq. They made use of the Coq repository C-CoRN [12] and built a matrix library with dependent types. Cano et al. [13] specifically designed CoqEAL, a library built on top of ssreflect [14] to develop efficient computer algebra programs with proofs of correctness. They represented a matrix as a list of lists for efficient generic matrix computation in Coq but they did not consider optimizations specific for matrices commonly used in quantum computation. Rand et al. [10] defined a quantum circuit language Qwire in Coq, and formally verified some quantum programs expressed in that language [5, 15]. Reasoning using their matrix library usually requires explicit computation, which does not scale well, as discussed in Section 6. Hietala et al. [16] developed a quantum circuit compiler VOQC in Coq, which uses several peephole optimization techniques such as replacement, propagation, and cancellation as proposed by Nam et al. [17] to reduce the number of unitary transformations. It is very different from our symbolic approach of simplifying matrix operations using the Dirac notation. Mahmoud et al. [18] formalized the semantics of Proto-Quipper in Coq and formally proved the type soundness property. They developed a linear logical framework within the Hybrid system [19] and used it to represent and reason about the linear type system of Quipper [20]. Note that although sparse matrix computation is well studied in other areas of Computer Science, we are not aware of any library in Coq dedicated to sparse matrices. We consider the symbolic approach proposed in the current work as a contribution in this perspective.

Apart from Coq, other proof assistants have also been used to verify quantum circuits and programs. Liu et al. [22] used the theorem prover Isabelle/HOL [23] to formalize a quantum Hoare logic [25] and verify its soundness and completeness for partial correctness. Unruh [6] developed a relational quantum Hoare logic and implemented an Isabelle-based tool to prove the security of post-quantum cryptography and quantum protocols. Beillahi et al. [26] verified quantum circuits with up to 190 two-qubit gates in HOL Light. It relies on the formalization of Hilbert spaces in HOL Light proposed by Mahmoud et al. in [27], where a number of laws about complex

functions and linear operators are proved. Although linear operators correspond to matrices in the finite-dimension case, our results are not implied by those in [26,27] because we are in a different setting, i.e. Coq. Furthermore, one of our main contributions is to represent sparse matrices using Dirac notation, which is convenient for readability and cancelling zero matrices due to orthogonality of basic vectors.

Notice that the laws in Table 1 play an important role in our symbolic reasoning of quantum circuits. Although they resemble to some laws in a ring, the matrices under our consideration can be of various dimensions and they do not form a ring. It is also critical that the multiplication of two matrices, e.g. a row vector and a column vector, could be a scalar number (and even zero). Thus, rings are not enough here. The proof-by-reflection technique for rings might be useful but are usually hard to develop. We have shown that the tactic-based method is already efficient in our application scenario, and also flexible for both fully-automated and interactive proofs.

8 Conclusion and future work

We have proposed a symbolic approach to reasoning about quantum circuits in Coq. It is based on a small set of equational laws which are exploited to design some simplification strategies. According to our case studies, the approach scales better than the usual one of explicitly representing matrices and is well suited to be automated in Coq.

Dealing with quantum circuits is our intermediate goal. More interesting algorithms such as the Shor's algorithm [7] also require classical computation. In the near future, we plan to formalize in Coq the semantics of a quantum programming language with both classical and quantum features.

References

1. C.H. Bennett, G. Brassard, C. Crepeau, R. Jozsa, A. Peres, and W. Wootters. Tele- porting an unknown quantum state via dual classical and EPR channels. *Physical Review Letters*, 70:1895–1899, 1993.
2. Nielsen M A, Chuang I L. Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, 2011.
3. Coq Development Team. The Coq Proof Assistant Reference Manual. Electronic resource, available from `http://coq.inria.fr`.
4. David Deutsch. Quantum theory, the Church-Turing principle, and the Universal Quantum Computer. *Proceedings of the Royal Society of London A*, 400:97-117, 1985.
5. Robert Rand, Jennifer Paykin, Steve Zdancewic. QWIRE Practice: Formal Verification of Quantum Circuits in Coq. In *Proceedings of the 14th International Conference on Quantum Physics and Logic*, EPTCS 266: 119-132, 2018.
6. Dirac P A M. A New Notation for Quantum Mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society* 35(3): 416-418, 1939.
7. Shor P W. Algorithms for Quantum Computation: Discrete Log and Factoring. In *Proc. FOCS 1994*, 124-133, IEEE Computer Society, 1994.
8. Sylvie Boldo, Catherine Lelay, Guillaume Melquiond. Coquelicot. Available at `http://coquelicot.saclay.inria.fr/`.
9. Daniel M. Greenberger, Michael A. Horne, Anton Zeilinger. Bell's theorem, Quantum Theory, and Conceptions of the Universe. pp. 73-76, Kluwer Academics, Dordrecht, The Netherlands 1989.
10. Jennifer Paykin, Robert Rand, Steve Zdancewic. QWIRE: A Core Language for Quantum Circuits. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, 52: 846-858, 2017.

11. Jaap Boender, Florian Kammüller, Rajagopal Nagarajan. Formalization of Quantum Protocols using Coq. In *Proceedings of the 12th International Workshop on Quantum Physics and Logic*, EPTCS 195:71-83, 2015.
12. Cruz-Filipe, Herman Geuvers, Freek Wiedijk. C-CoRN, the Constructive Coq Repository at Nijmegen. *Mathematical Knowledge Management*, Lecture Notes in Computer Science 3119: 88-103, 2004.
13. Guillaume Cano, Cyril Cohen, Maxime Dénès, Anders Mörtberg, Vincent Siles. CoqEAL - The Coq Effective Algebra Library. `https://github.com/CoqEAL/CoqEAL`, 2016.
14. Mathematical Components team. Mathematical Components. `https://math-comp.github.io`
15. Robert Rand, Jennifer Paykin, Dong-Ho Lee, Steve Zdancewic. ReQWIRE: Reasoning about Reversible Quantum Circuits. In *Proceedings of the 15th International Conference on Quantum Physics and Logic*, EPTCS 287: 299-312, 2019.
16. Kesha Hietala, Robert Rand, Shih-Han Hung, Xiaodi Wu, Michael Hicks. Verified Optimization in a Quantum Intermediate Representation. In *Proceedings of the 16th International Conference on Quantum Physics and Logic*, CoRR abs/1904.06319, 2019.
17. Yunseong Nam, Neil J. Ross, Yuan Su, Andrew M. Childs, Dmitri Maslov. Automated Optimization of Large Quantum Circuits with Continuous Parameters. *npj Quantum Information*, 4(1): 23, 2018.
18. Mahmoud M Y, Felty A P. Formalization of Metatheory of the Quipper Quantum Programming Language in a Linear Logic. *Journal of Automated Reasoning*, 63(4): 967-1002, 2019.
19. Felty A P, Momigliano A. Hybrid: A Definitional Two-Level Approach to Reasoning with Higher-Order Abstract Syntax. *Journal of Automated Reasoning*, 48(1): 43-105, 2012
20. Green A S, Lumsdaine P L, Ross N J, Selinger P, Valiron B. Quipper: A Scalable Quantum Programming Language. *Acm Sigplan Notices*, 48(6): 333-342, 2013.
21. Anticoli L, Piazza C, Taglialegne L, Zuliani P. Towards Quantum Programs Verification: From Quipper Circuits to QPMC. In *International Conference on Reversible Computation*, LNCS 9720: 213-219, 2016.
22. Junyi Liu, Bohua Zhan, Shuling Wang, Shenggang Ying, Tao Liu, Yangjia Li, Mingsheng Ying, Naijun Zhan. Formal Verification of Quantum Algorithms Using Quantum Hoare Logic. In *Proc. CAV 2019*, LNCS 11562: 187-207. Springer, 2019.
23. Tobias Nipkow, Lawrence C. Paulson, Markus Wenzel. Isabelle/HOL: A Proof Assistant for Higher-Order Logic. Lecture Notes in Computer Science. Springer, 2002.
24. J. von Neumann. States, Effects and Operations: Fundamental Notions of Quantum Theory. Princeton University Press, 1955.
25. Mingsheng Ying. Foundations of Quantum Programming. Morgan Kaufmann, 2016.
26. Beillahi S M, Mahmoud M Y, Tahar S. A Modeling and Verification Framework for Optical Quantum Circuits. *Formal Aspects of Computing* 31: 321-351, 2019.
27. Mahmoud M Y, Aravantinos Y, Tahar S. Formalization of Infinite Dimension Linear Spaces with Application to Quantum Theory. In *Nasa Formal Methods Symposium*, LNCS 7871: 413-427, 2013.
28. Daniel R. Simon, On the power of quantum computation, SIAM Journal on Computing 26 (5), 1474 - 1483, 1997.