Local Reasoning about Probabilistic Behaviour for Classical–Quantum Programs*

Yuxin Deng, Huiling Wu, and Ming Xu

Shanghai Key Laboratory of Trustworthy Computing
East China Normal University, Shanghai, China

Abstract. Verifying the functional correctness of programs with both classical and quantum constructs is a challenging task. The presence of probabilistic behaviour entailed by quantum measurements and unbounded while loops complicate the verification task greatly. We propose a new quantum Hoare logic for local reasoning about probabilistic behaviour by introducing distribution formulas to specify probabilistic properties. We show that the proof rules in the logic are sound with respect to a denotational semantics. To demonstrate the effectiveness of the logic, we formally verify the correctness of non-trivial quantum algorithms including the HHL and Shor's algorithms.

Keywords: Quantum computing · Program verification · Hoare logic· Separation logic · Local reasoning.

1 Introduction

Programming is an error-prone activity, and the situation is even worse for quantum programming, which is far less intuitive than classical computing. Therefore, developing verification and analysis techniques to ensure the correctness of quantum programs is an even more important task than that for classical programs.

Hoare logic [12] is probably the most widely used program logic to verify the correctness of programs. It is useful for reasoning about deterministic and probabilistic programs. A lot of efforts have been made to reuse the nice idea to verify quantum programs. Ying [28,29] was the first to establish a sound and relatively complete quantum Hoare logic to reason about pure quantum programs, i.e., quantum programs without classical variables. This work triggered a series of research in this direction. For example, Zhou et al. [34] proposed an applied quantum Hoare logic by only using projections as preconditions and postconditions, which makes the practical use of quantum Hoare logic much easier. Barthe et al. [2] extended quantum Hoare logic to relational verification by

* This work was supported by the National Natural Science Foundation of China under Grant Nos. 61832015, 62072176, 12271172 and 11871221, Shanghai Trusted Industry Internet Software Collaborative Innovation Center, and the "Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software under Grant No. 22510750100.

introducing a quantum analogue of probabilistic couplings. Li and Unruh [16] defined a quantum relational Hoare logic with expectations in pre- and post-conditions. Formalization of quantum Hoare logic in proof assistants such as Isabelle/HOL [20] and Coq [4] was accomplished in [17] and [33], respectively. Ying et al. [31] defined a class of disjoint parallel quantum programs and generalised the logic in [28] to this setting. As an extension of Hoare logic, separation logic turns out to be useful for verifying deterministic, concurrent, and probabilistic programs [22,21,5,25,3,1,15]. Zhou et al. [32] developed a quantum separation logic for local reasoning about quantum programs. However, all the work mentioned above cannot deal with programs that have both classical and quantum data.

In order to verify quantum programs found in almost all practical quantum programming frameworks such as Qiskit[1], $Q^{\#}$[2], Cirq[3], etc., we have to explicitly consider programs with both classical and quantum constructs. Verification techniques for this kind of hybrid programs have been put forward in the literature [6,13,26,27,10,7,9,14]. For example, Chadha et al. [6] proposed an ensemble exogenous quantum propositional logic for a simple quantum language with bounded iteration. The expressiveness of the language is very limited, and algorithms involving unbounded while loops such as the HHL and Shor's algorithms [11,24] cannot be described. Kakutani [13] presented a quantum Hoare logic for an imperative language with while loops, but the rule for them has no invariance condition. Instead, an infinite sequence of assertions has to be used. Unruh introduced a quantum Hoare logic with ghost variables to express properties such as that a quantum variable is disentangled with others [26] and a relational Hoare logic [27] for security analysis of post-quantum cryptography and quantum protocols. Deng and Feng [7] provided an abstract and a concrete proof system for classical–quantum programs, with the former being sound and relatively complete, while the latter being sound. Feng and Ying [10] introduced classical–quantum assertions, which are a class of mappings from classical states to quantum predicates, to analyse both classical and quantum properties. The approach was extended to verify distributed quantum programs in [9]. However, except for [14], all the work above offers no support for local reasoning, which is an obvious drawback. In the case that we have a large quantum register but only a few qubits are modified, it is awkward to always reason about the global states of the quantum register. Based on this observation, Le et al. [14] provided an interesting quantum interpretation of the separating conjunction, so to infuse separation logic into a Hoare-style framework and thus support local reasoning. However, a weakness of their approach is that it cannot handle probabilistic behaviour, which exists inherently in quantum programs, in a satisfactory way. Let us illustrate this with a simple example.

Example 1. The program **addM** defined below first initialises two qubits q_0 and q_1, and then applies the Hadamard gate H to each of them. By measuring

[1] `https://qiskit.org`
[2] `https://github.com/microsoft/qsharp-language`
[3] `https://github.com/quantumlib/Cirq`

them with the measurement operators $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$, we add the measurement results and assign the sum to the variable v.

$$\begin{aligned}\textbf{addM} \triangleq \quad & q_0 := |0\rangle\,;\ H[q_0];\\ & q_1 := |0\rangle\,;\ H[q_1];\\ & v_0 := M[q_0];\\ & v_1 := M[q_1];\\ & v := v_0 + v_1\end{aligned}$$

Since the classical variable v_0 takes either 0 or 1 with equal chance, and similarly for v_1, the probability that variable v is assigned to 1 should be exactly $\frac{1}{2}$. However, in the program logic in [14], this property cannot be specified. □

We propose a novel quantum Hoare logic for a classical–quantum language. Two distribution formulas $\oplus_{i\in I} p_i \cdot F_i$ and $\oplus_{i\in I} F_i$ are introduced. A program state μ, which is a partial density operator valued distribution (POVD) [7], satisfies the formula $\oplus_{i\in I} p_i \cdot F_i$ if μ can be split into the weighted sum of i parts called μ_i and each μ_i satisfies the formula F_i. A state μ satisfies the formula $\oplus_{i\in I} F_i$ if there exists a collection of probabilities $\{p_i\}_{i\in I}$ with $\sum_{i\in I} p_i = 1$ such that $\oplus_{i\in I} p_i \cdot F_i$ can be satisfied. In other words, the splitting of μ does not necessarily follow a fixed set of weights. With distribution formulas, we can conveniently reason about the probabilistic behaviour mentioned in Example 1 (more details will be discussed in Example 2), and give an invariance condition in the proof rule for while loops. In addition, we adopt the labelled Dirac notation emphasised in [33] to facilitate local reasoning. Our program logic is shown to be sound and can be used to prove the correctness of non-trivial quantum algorithms including the HHL and Shor's algorithms. Therefore, the main contributions of the current work include the following aspects:

- We propose to use distribution formulas in a new quantum Hoare logic to specify the probabilistic behaviour of classical–quantum programs. Distribution formulas are useful to give an invariance condition in the proof rule for while loops, so to avoid an infinite sequence of assertions in the rule.
- We prove the soundness of our logic that allows for local reasoning in the spirit of separation logic.
- We demonstrate the effectiveness of the logic by proving the correctness of the HHL and Shor's algorithms.

The rest of the paper is structured as follows. In Section 2 we recall some basic notations about quantum computing. In Section 3 we review the syntax and denotational semantics of a classical–quantum imperative language considered in [7]. In Section 4 we define an assertion language and propose a proof system for local reasoning about quantum programs. We also prove the soundness of the system. In Section 5 we apply our framework to verify the HHL and Shor's algorithms. Finally, we conclude in Section 6 and discuss possible future work. Missing proofs are given in [8].

2 Preliminaries

We briefly recall some basic notations from linear algebra and quantum mechanics which are needed in this paper. For more details, we refer to [19].

A *Hilbert space* $\mathcal{H}$ is a complete vector space 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. For any vector $|\psi\rangle\in\mathcal{H}$, its length $\||\psi\rangle\|$ is defined to be $\sqrt{\langle\psi|\psi\rangle}$, and it is said to be *normalised* if $\||\psi\rangle\| = 1$. 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 $A^\dagger A = AA^\dagger = I_{\mathcal{H}}$ 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 $\mathcal{H}$. A linear operator $A\in\mathcal{L}(\mathcal{H})$ is *positive* if $\langle\phi|A|\phi\rangle\geq 0$ for any vector $|\phi\rangle\in\mathcal{H}$. The *Löwner order* $\sqsubseteq$ on the set of Hermitian operators on $\mathcal{H}$ is defined by letting $A\sqsubseteq B$ if and only if $B-A$ is positive.

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 as the following: 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 $\mathcal{H}_i$. Given $\mathcal{H}_1$ and $\mathcal{H}_2$, the *partial trace* with respect to $\mathcal{H}_2$, written $\mathrm{tr}_{\mathcal{H}_2}$, is a linear mapping from $\mathcal{L}(\mathcal{H}_1\otimes\mathcal{H}_2)$ to $\mathcal{L}(\mathcal{H}_1)$ such that for any $|\psi_1\rangle,|\phi_1\rangle\in\mathcal{H}_1$ and $|\psi_2\rangle,|\phi_2\rangle\in\mathcal{H}_2$,

$$\mathrm{tr}_{\mathcal{H}_2}(|\psi_1\rangle\langle\phi_1|\otimes|\psi_2\rangle\langle\phi_2|) = \langle\psi_2|\phi_2\rangle|\psi_1\rangle\langle\phi_1|.$$

By applying quantum gates to qubits, we can change their states. For example, the Hadamard gate (H gate) can be applied on a single qubit, while the

controlled-NOT gate (*CNOT* gate) can be applied on two qubits. Their representations in terms of matrices are given as

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \text{and} \quad CNOT = \begin{pmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,0\,1 \\ 0\,0\,1\,0 \end{pmatrix}.$$

According to von Neumann's formalism of quantum mechanics [18], 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 $\mathrm{tr}(\rho) = 1$. A *partial density operator* ρ is a positive linear operator with $\mathrm{tr}(\rho) \leq 1$.

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 ρ_1 and ρ_2, respectively, then $\rho_2 = U\rho_1 U^\dagger$ for some unitary operator U which depends only on t_1 and t_2.

A quantum *measurement* is described by a collection $\{M_m\}$ of positive operators, called 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 system is in 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 $M_m \rho M_m^\dagger / p(m)$.

3 A Classical–Quantum Language

We recall the simple classical–quantum imperative language **QIMP** as defined in [7]. It is essentially similar to a few imperative languages considered in the literature [23,30,27,10]. We introduce its syntax and denotational semantics.

3.1 Syntax

We assume three types of data in our language: `Bool` for booleans, `Int` for integers, and `Qbt` for quantum bits (qubits). Let $\mathbb{Z}$ be the set of integer constants, ranged over by n. Let **Cvar**, ranged over by $x, y, ...$, be the set of classical variables, and **Qvar**, ranged over by $q, q', ...$, the set of quantum variables. It is assumed that both **Cvar** and **Qvar** are countable. We assume a set **Aexp** of arithmetic expressions over `Int`, which includes **Cvar** as a subset and is ranged over by $a, a', ...$, and a set of boolean-valued expressions **Bexp**, ranged over by $b, b', ...$, with the usual boolean constants **true**, **false** and boolean connectives such as $\neg, \wedge$ and $\vee$. We assume a set of arithmetic functions (e.g. $+$, $-$, $*$, etc.)

(**Aexp**)	a	::=	$n \mid x, y, ... \mid f_m(a, ..., a)$
(**Bexp**)	b	::=	$\mathbf{true} \mid \mathbf{false} \mid P_m(a, ..., a) \mid b \wedge b \mid \neg b \mid \forall x.b$
(**Com**)	c	::=	$\mathbf{skip} \mid \mathbf{abort} \mid x := a \mid c; c$
			$\mid \mathbf{if}\ b\ \mathbf{then}\ c\ \mathbf{else}\ c\ \mathbf{fi} \mid \mathbf{while}\ b\ \mathbf{do}\ c\ \mathbf{od}$
			$\mid q := \|0\rangle \mid U[\bar{q}] \mid x := M[\bar{q}]$

Table 1. Syntax of quantum programs

ranged over by the symbol f_m, and a set of boolean predicates (e.g. $=$, $\leq$, $\geq$, etc.) ranged over by P_m, where m indicates the number of variables involved. We further assume that only classical variables can occur freely in both arithmetic and boolean expressions.

We let U range over unitary operators, which can be user-defined matrices or built in if the language is implemented. For example, a concrete U could be the 1-qubit Hadamard operator H, or the 2-qubit controlled-NOT operator $CNOT$, etc. Similarly, we write M for the measurement described by a collection $\{M_i\}$ of measurement operators, with each index i representing a measurement outcome. For example, to describe the measurement of the qubit referred to by variable q in the computational basis, we can write $M := \{M_0, M_1\}$, where $M_0 = |0\rangle_q\langle 0|$ and $M_1 = |1\rangle_q\langle 1|$.

Sometimes, we use metavariables which are primed or subscripted, e.g. x', x_0 for classical variables. We abbreviate a tuple of quantum variables $\langle q_1, ..., q_n\rangle$ as $\bar{q}$ if the length n of the tuple is not important. If two tuples of quantum variables $\overline{q}$ and $\overline{q'}$ are disjoint, where $\overline{q}=\langle q_1, ..., q_n\rangle$ and $\overline{q'} = \langle q_{n+1}, q_{n+2}, ..., q_{n+m}\rangle$, then their concatenation is a larger tuple $\overline{q}\overline{q'}=\langle q_1, ..., q_n, q_{n+1}, ..., q_{n+m}\rangle$. If no confusion arises, we occasionally use a tuple to stand for a set.

The formation rules for arithmetic and boolean expressions as well as commands are defined in Table 1. An arithmetic expression can be an integer, a variable, or built from other arithmetic expressions by some arithmetic functions. A boolean expression can be a boolean constant, built from arithmetic expressions by some boolean predicates or formed by using the usual boolean operations. A command can be a skip statement, an abort statement, a classical assignment, a conditional statement, or a while-loop, as in many classical imperative languages. The command **abort** represents the unsuccessful termination of programs. In addition, there are three commands that involve quantum data. The command $q := |0\rangle$ initialises the qubit referred to by variable q to be the basis state $|0\rangle$. The command $U[\bar{q}]$ applies the unitary operator U to the quantum system referred to by $\bar{q}$. The command $x := M[\bar{q}]$ performs a measurement M on $\bar{q}$ and assigns the measurement outcome to x. It differs from a classical assignment because the measurement M may change the quantum state of $\bar{q}$, besides the fact that the value of x is updated.

For convenience, we further define the following syntactic sugar for the initialization of a sequence of quantum variables: $\overline{q} = |0\rangle^{\otimes n}$, where $\overline{q} = \langle q_1, ..., q_n\rangle$,

is an abbreviation of the commands:

$$q_1 = |0\rangle\,; q_2 = |0\rangle\,; \ldots; q_n := |0\rangle\,.$$

3.2 Denotational Semantics

In the presence of classical and quantum variables, the execution of a **QIMP** program involves two types of states: classical states and quantum states.

As usual, a classical state is a function $\sigma : \mathbf{Cvar} \to \mathbb{Z}$ from classical variables to integers, where $\sigma(x)$ represents the value of classical variable x. For each quantum variable $q \in \mathbf{Qvar}$, we assume a 2-dimensional Hilbert space $\mathcal{H}_q$ to be the state space of the q-system. For any finite subset V of **Qvar**, we denote

$$\mathcal{H}_V \;=\; \bigotimes_{q \in V} \mathcal{H}_q.$$

That is, $\mathcal{H}_V$ is the Hilbert space spanned by tensor products of the individual state spaces of the quantum variables in V. Throughout the paper, when we refer to a subset of **Qvar**, it is assumed to be finite. Given $V \subseteq \mathbf{Qvar}$, the set of *quantum states* consists of all partial density operators in the space $\mathcal{H}_V$, denoted by $\mathcal{D}^-(\mathcal{H}_V)$. A *machine state* is a pair $\langle \sigma, \rho \rangle$ where σ is a classical state and ρ a quantum state. In the presence of measurements, we often need to consider an ensemble of states. For that purpose, we introduce a notion of distribution.

Definition 1. *[7] Suppose $V \subseteq$ **Qvar** and Σ is the set of classical states, i.e., the set of functions of type **Cvar** $\to \mathbb{Z}$. A* partial density operator valued distribution (POVD) *is a function $\mu : \Sigma \to \mathcal{D}^-(\mathcal{H}_V)$ with $\sum_{\sigma \in \Sigma} \mathrm{tr}(\mu(\sigma)) \leq 1$.*

Intuitively, a POVD μ represents a collection of machine states where each classical state σ is associated with a quantum state $\mu(\sigma)$. The notation of POVD is called classical–quantum state in [10]. If the collection has only one element σ, we explicitly write $(\sigma, \mu(\sigma))$ for μ. The support of μ, written $\lceil \mu \rceil$, is the set $\{\sigma \in \Sigma \mid \mu(\sigma) \neq 0\}$. We can also define the addition of two distributions by letting $(\mu_1 + \mu_2)(\sigma) = \mu_1(\sigma) + \mu_2(\sigma)$.

We interpret programs as POVD transformers. We write **POVD** for the set of POVDs called distribution states. Given an expression e, we denote its interpretation with respect to machine state (σ, ρ) by $[\![e]\!]_{(\sigma,\rho)}$. The denotational semantics of commands is displayed in Table 2, where we omit the denotational semantics of arithmetic and boolean expressions such as $[\![a]\!]_\sigma$ and $[\![b]\!]_\sigma$, which is almost the same as in the classical setting because the quantum part plays no role for those expressions. A state evolves into a POVD after some quantum qubits are measured, with the measurement outcomes assigned to a classical variable. Two other quantum commands, initialisation of qubits and unitary operations, are deterministic and only affect the quantum part of a state. As usual, we define the semantics of a loop (**while** b **do** c **od**) as the limit of its lower approximations, where the n-th lower approximation of $[\![\textbf{while } b \textbf{ do } c \textbf{ od}]\!]_{(\sigma,\rho)}$ is $[\![(\textbf{if } b \textbf{ then } c \textbf{ fi})^n; \textbf{if } b \textbf{ then abort fi}]\!]_{(\sigma,\rho)}$, where

$$
\begin{aligned}
[\![\mathbf{skip}]\!]_{(\sigma,\rho)} &= (\sigma,\rho)\\
[\![\mathbf{abort}]\!]_{(\sigma,\rho)} &= \varepsilon\\
[\![x := a]\!]_{(\sigma,\rho)} &= (\sigma[[\![a]\!]_\sigma/x],\rho)\\
[\![c_0; c_1]\!]_{(\sigma,\rho)} &= [\![c_1]\!]_{[\![c_0]\!]_{(\sigma,\rho)}}\\
[\![\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1\ \mathbf{fi}]\!]_{(\sigma,\rho)} &= \begin{cases} [\![c_0]\!]_{(\sigma,\rho)} \text{ if } [\![b]\!]_\sigma = \mathbf{true}\\ [\![c_1]\!]_{(\sigma,\rho)} \text{ if } [\![b]\!]_\sigma = \mathbf{false}\end{cases}\\
[\![\mathbf{while}\ b\ \mathbf{do}\ c\ \mathbf{od}]\!]_{(\sigma,\rho)} &= \lim_{n\to\infty} [\![(\mathbf{if}\ b\ \mathbf{then}\ c\ \mathbf{fi})^n; \mathbf{if}\ b\ \mathbf{then\ abort\ fi}]\!]_{(\sigma,\rho)}\\
[\![q := |0\rangle]\!]_{(\sigma,\rho)} &= (\sigma,\rho')\\
&\quad \text{where } \rho' := |0\rangle_q\langle 0|\rho|0\rangle_q\langle 0| + |0\rangle_q\langle 1|\rho|1\rangle_q\langle 0|\\
[\![U[\bar{q}]]\!]_{(\sigma,\rho)} &= (\sigma, U\rho U^\dagger)\\
[\![x := M[\bar{q}]]\!]_{(\sigma,\rho)} &= \mu\\
&\quad \text{where } M = \{M_i\}_{i\in I} \text{ and}\\
&\quad \mu(\sigma') = \textstyle\sum_i \{M_i\rho M_i^\dagger \mid \sigma[i/x] = \sigma'\}\\
[\![c]\!]_\mu &= \textstyle\sum_{\sigma\in\lceil\mu\rceil} [\![c]\!]_{(\sigma,\mu(\sigma))}.
\end{aligned}
$$

Table 2. Denotational semantics of commands

(**if** b **then** c **fi**) is shorthand for (**if** b **then** c **else skip fi**) and c^n is the command c iterated n times with $c^0 \equiv \mathbf{skip}$. The limit exists because the sequence $([\![(\mathbf{if}\ b\ \mathbf{then}\ c\ \mathbf{fi})^n; \mathbf{if}\ b\ \mathbf{then\ abort\ fi}]\!]_{(\sigma,\rho)})_{n\in\mathbb{N}}$ is increasing and bounded with respect to the Löwner order [7, Lemma 3.2]. We write ε for the special POVD whose support is the empty set.

We remark that the semantics $[\![c]\!]_{(\sigma,\rho)}$ of a command c in initial state (σ,ρ) is a POVD. The lifted semantics $[\![c]\!]_\mu$ of a command c in initial POVD μ is also a POVD. Furthermore, the function $[\![c]\!]$ is linear in the sense that

$$[\![c]\!]_{p_0\mu_0+p_1\mu_1} = p_0[\![c]\!]_{\mu_0} + p_1[\![c]\!]_{\mu_1}$$

where we write $p\mu$ for the POVD defined by $(p\mu)(\sigma) = p\cdot\mu(\sigma)$.

Similarly as in [14], we take advantage of the labelled Dirac notation throughout the paper with subscripts identifying the subsystem where a ket/bra/operator lies or operates. For example, the subscript in $|a\rangle_{\overline{q}}$ indicates the Hilbert space $\mathcal{H}_{\overline{q}}$ where the state $|a\rangle$ lies. The notation $|a\rangle\,|a\rangle$ and $|a\rangle_{\overline{q}}|a\rangle_{\overline{q}}$ are the abbreviations of $|a\rangle\otimes|a\rangle$ and $|a\rangle_{\overline{q}}\otimes|a\rangle_{\overline{q}}$ respectively. We also use operators with subscripts like $A_{\overline{q}}$ to identify the Hilbert space $\mathcal{H}_{\overline{q}}$ where the operator A is applied.

(Classical expression)	e	::=	$n \mid x, y, ... \mid f_m(e, ..., e)$
(Classical formula)	P	::=	$\mathbf{true} \mid \mathbf{false} \mid P_m(e, ..., e) \mid P \wedge P \mid \neg P \mid \forall x.P(x)$
(Quantum expression)	$\|s\rangle$	::=	$\|a\rangle_{\overline{q}} \mid \|s\rangle \otimes \|s\rangle$
(State formula)	F	::=	$P \mid \|s\rangle \mid F \odot F \mid F \wedge F \mid \neg F$
(Distribution formula)	D	::=	$\oplus_{i \in I}\, p_i \cdot F_i \mid \oplus_{i \in I}\, F_i$

Table 3. Syntax of assertion languages

4 Proof System

In this section we present a proof system for local reasoning about probabilistic behaviour of quantum programs. We first define an assertion language, then propose a Hoare-style proof system, and finally prove the soundness of the system.

4.1 Assertion Language

We now introduce an assertion language for our programs, whose syntax is given in Table 3. Classical expressions are arithmetic expressions with integer constants, ranged over by e. Classical formulas, ranged over by P, include the boolean constants **true** and **false**, boolean predicates in the form P_m and any P connected by boolean operators such as negation, conjunction, and universal quantification. They are intended to capture properties of classical states. Quantum expressions, ranged over by $|s\rangle$, include quantum states of the form $|a\rangle_{\overline{p}}$ and the tensor product $|s\rangle \otimes |s\rangle$ which we abbreviate as $|s\rangle\,|s\rangle$. Here $|a\rangle_{\overline{p}}$ can be any computational basis or their linear combinations in the Hilbert space $\mathcal{H}_{\overline{p}}$. State formulas, ranged over by F, are used to express properties on both classical and quantum states, which include the classical formula P, the quantum expression $|s\rangle$ and any expression connected by boolean operators such as negation and conjunction. In addition, we introduce a new connective $\odot$ to express an assertion of two separable systems. Following [14], we use $\mathrm{free}(F)$ to denote the set of all free classical and quantum variables in F. For example, $\mathrm{free}(|a_1\rangle_{\overline{q_1}} \otimes |a_2\rangle_{\overline{q_2}}) = \overline{q_1 q_2}$. Moreover, we use $\mathrm{qfree}(F)$ to denote the set of all quantum variables in F. For the formula $F_1 \odot F_2$ to be well defined, we impose the syntactical restriction that $\mathrm{free}(F_1) \cap \mathrm{free}(F_2) = \emptyset$. Intuitively, a quantum state satisfies $F_1 \odot F_2$ if the state mentions two disjoint subsystems whose states satisfy F_1 and F_2 respectively. Distribution formulas consist of some state formulas F_i connected by the connective $\oplus$ with the weights given by p_i satisfying $\sum_{i \in I} p_i = 1$ as well as the non-probabilistic formula $\oplus_{i \in I} F_i$. If there is a collection of distribution formulas $D_i = \oplus_j p_{ij} \cdot F_{ij}$ and a collection of probabilities p_i with $\sum_{i \in I} p_i = 1$, we sometimes write $\oplus_i p_i \cdot D_i$ to mean the formula $\oplus_{ij} p_i p_{ij} \cdot F_{ij}$.

We use the notation $\mu \models F$ to indicate that the state μ satisfies the assertion F. The satisfaction relation $\models$ is defined in Table 4. When writing $(\sigma, \rho) \models F$, we mean that (σ, ρ) is a machine state and ρ is its quantum part representing

$$
\begin{array}{lll}
(\sigma,\rho) \models P & \text{if} & [\![P]\!]_\sigma = \mathbf{true} \\
(\sigma,\rho) \models |s\rangle & \text{if} & \frac{\rho}{\mathrm{tr}(\rho)}|_{\overline{q}} = |s\rangle\langle s| \quad \text{where } \overline{q} = \mathrm{free}(|s\rangle) \\
(\sigma,\rho) \models F_1 \wedge F_2 & \text{if} & (\sigma,\rho) \models F_1 \wedge (\sigma,\rho) \models F_2 \\
(\sigma,\rho) \models \neg F & \text{if} & (\sigma,\rho) \not\models F \\
(\sigma,\rho) \models F_1 \odot F_2 & \text{if} & (\sigma,\rho|_{\mathrm{qfree}(F_1)}) \models F_1 \wedge (\sigma,\rho|_{\mathrm{qfree}(F_2)}) \models F_2 \\
\mu \models F & \text{if} & \forall\, \sigma \in \lceil \mu \rceil.\ (\sigma, \mu(\sigma)) \models F \\
\mu \models \oplus_{i\in I} p_i \cdot F_i & \text{if} & \exists\, \mu_1 \cdots \exists\, \mu_m.\ [(\bigwedge_{i\in I} \mu_i \models F_i) \wedge \mu = \sum_{i\in I} p_i \cdot \mu_i] \\
 & & \text{for } I = \{1,\ldots,m\} \\
\mu \models \oplus_{i\in I} F_i & \text{if} & \exists\, p_1 \cdots \exists p_m.\ [(\bigwedge_{i\in I} p_i \geq 0) \wedge \mu \models \oplus_{i\in I} p_i \cdot F_i] \\
 & & \text{for } I = \{1,\ldots,m\}
\end{array}
$$

Table 4. Semantics of assertions

the status of the whole quantum system in a program under consideration. We use $[\![P]\!]_\sigma$ to denote the evaluation of the classical predicate P with respect to the classical state σ. If V is set of quantum variables and $V' \subseteq V$, we write $\rho|_{V'}$ for the reduced density operator $\mathrm{tr}_{V\setminus V'}(\rho)$ obtained by restricting ρ to V'. A machine state (σ,ρ) satisfies the formula $|s\rangle$ if the reduced density operator obtained by first normalising ρ and then restricting it to $\mathrm{free}(|s\rangle)$ becomes $|s\rangle\langle s|$. The state (σ,ρ) satisfies the formula $F_1 \odot F_2$ if $\mathrm{free}(F_1)$ and $\mathrm{free}(F_2)$ are disjoint and the restrictions of ρ to $\mathrm{qfree}(F_1)$ and $\mathrm{qfree}(F_2)$ satisfy the two sub-formulas F_1 and F_2. The assertion F holds on a distribution μ when F holds on each pure state in the support of μ. A distribution state μ satisfies the distribution formula $\oplus_{i\in I} p_i \cdot F_i$ if μ is a linear combination of some μ_i with weights p_i and each μ_i satisfies F_i. In the special case that μ is a pure state, we have that $(\sigma,\rho) \models \oplus_{i\in I} p_i \cdot F_i$ means $(\sigma,\rho) \models F_i$ for every $i \in I$. The formula $\oplus_{i\in I} F_i$ is a nondeterministic version of the distribution formulas in the form $\oplus_{i\in I} p_i \cdot F_i$ without fixing the weights p_i, so the weights can be arbitrarily chosen as long as their sum is 1. For other assertions, the semantics should be self-explanatory.

From the relation $\models$, we can derive a quantitative definition of satisfaction, where a state satisfies a predicate only to certain degree.

Definition 2. *Let F be an assertion and $p \in [0,1]$ a real number. We say that the probability of a state μ satisfying F is p, written by*

$$\mathbb{P}_\mu(F) = p,$$

if there exist two states μ_1 and μ_2 such that $\mu = p\mu_1 + (1-p)\mu_2$, $\mu_1 \models F$ and $\mu_2 \models \neg F$, and moreover, p is the maximum probability for this kind of decomposition of μ.

In [14], assertions with disjunctions are used as the postconditions of measurement statements. However, this approach is awkward when reasoning about probabilities. Let us take a close look at the problem in Example 2.

Example 2. We revisit the program **addM** discussed in Example 1, which measures the variables q_0 and q_1, both in the state $|+\rangle\langle+|$, and subsequently adds their results of measurements. Here M is a projective measurement $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$.

$$\begin{aligned}\mathbf{addM} \triangleq \quad & q_0 := |0\rangle\,; H[q_0];\\ & q_1 := |0\rangle\,; H[q_1];\\ & v_0 := M[q_0];\\ & v_1 := M[q_1];\\ & v := v_0 + v_1\end{aligned}$$

After the measurements on q_0 and q_1, the classical variables v_0 and v_1 are assigned to either 0 or 1 with equal probability. By executing the last command, we assign the sum of v_0 and v_1 to v, and obtain the following POVD μ.

$$\begin{array}{rcl} & v_0v_1v & \quad q_0q_1 \\ & 000 & \mapsto \frac{1}{4}\,|00\rangle\langle 00| \\ \mu: & 011 & \mapsto \frac{1}{4}\,|01\rangle\langle 01| \\ & 101 & \mapsto \frac{1}{4}\,|10\rangle\langle 10| \\ & 112 & \mapsto \frac{1}{4}\,|11\rangle\langle 11| \end{array}$$

The second column represents the four classical states while the last column shows the four quantum states. For example, we have $\sigma_1(v_0v_1v) = 000$ and $\rho_1 = |00\rangle\langle 00|$. Let $\mu_i = (\sigma_i, \rho_i)$ for $1 \leq i \leq 4$. Then $\mu = \frac{1}{2}\mu_{14} + \frac{1}{2}\mu_{23}$, where $\mu_{14} = \frac{1}{2}\mu_1 + \frac{1}{2}\mu_4$ and $\mu_{23}{=}\frac{1}{2}\mu_2 + \frac{1}{2}\mu_3$. Since $\mu_i \models (v = 1)$ for $i = 2, 3$, it follows that $\mu_{23} \models (v = 1)$. On the other hand, we have $\mu_{14} \models (v \neq 1)$. Therefore, it follows that $\mathbb{P}_\mu(v = 1) = \frac{1}{2}$. That is, the probability for μ to satisfy the assertion $v = 1$ is $\frac{1}{2}$.

Alternatively, we can express the above property as a distribution formula. Let $D = \frac{1}{2} \cdot (v = 1) \oplus \frac{1}{2} \cdot (v \neq 1)$. We see that $\mu \models D$ due to the fact that $\mu = \frac{1}{2}\mu_{23} + \frac{1}{2}\mu_{14}$, $\mu_{23} \models (v = 1)$ and $\mu_{14} \models (v \neq 1)$.

In [14], there is no distribution formula. The best we can do is to use a disjunctive assertion to describe the postcondition of the above program.

$$\begin{aligned} F \triangleq\ & \tfrac{1}{4} \cdot (v = 0 \wedge |00\rangle_{q_0q_1}) \vee \tfrac{1}{4} \cdot (v = 1 \wedge |01\rangle_{q_0q_1}) \vee \\ & \tfrac{1}{4} \cdot (v = 1 \wedge |10\rangle_{q_0q_1}) \vee \tfrac{1}{4} \cdot (v = 2 \wedge |11\rangle_{q_0q_1}). \end{aligned}$$

This assertion does not take the mutually exclusive correlations between different branches into account. For example, it is too weak for us to prove that $\mathbb{P}(v = 1) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$, as discussed in more details in [14]. From this example, we see that distribution formulas give us a more accurate way of describing the behaviour of measurement statements than disjunctive assertions. □

4.2 Proof System

In this subsection, we present a series of inference rules for classical–quantum programs, which will be proved to be sound in the next section. As usual, we

$$\frac{}{\{D\}\ \mathbf{skip}\ \{D\}}[\text{Skip}] \qquad \frac{}{\{D\}\ \mathbf{abort}\ \{\mathbf{false}\}}[\text{Abort}]$$

$$\frac{}{\{D[a/x]\}\ x := a\ \{D\}}[\text{Assgn}] \qquad \frac{\{D_0\}\ c_0\ \{D_1\} \quad \{D_1\}\ c_1\ \{D_2\}}{\{D_0\}\ c_0; c_1 \{D_2\}}[\text{Seq}]$$

$$\frac{\{F_1 \wedge b\}\ c_1\ \{F_1'\} \quad \{F_2 \wedge \neg b\}\ c_2\ \{F_2'\}}{\{p(F_1 \wedge b) \oplus (1-p)(F_2 \wedge \neg b)\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \mathbf{fi}\ \{pF_1' \oplus (1-p)F_2'\}}[\text{Cond}]$$

$$\frac{}{\{\mathbf{false}\}\ c\ \{D\}}[\text{Absurd}] \qquad \frac{D_0 \Rightarrow D_1 \quad \{D_1\}\ c\ \{D_2\} \quad D_2 \Rightarrow D_3}{\{D_0\}\ c\ \{D_3\}}[\text{Conseq}]$$

$$\frac{D = (F_0 \wedge b) \oplus (F_1 \wedge \neg b) \quad \{F_0 \wedge b\}\ c\ \{D\}}{\{D\} \quad \mathbf{while}\ b\ \mathbf{do}\ c\ \mathbf{od}\ \{F_1 \wedge \neg b\}}[\text{While}]$$

$$\frac{\{F_1\}\ c\ \{F_1'\} \quad \{F_2\}\ c\ \{F_2'\}}{\{F_1 \wedge F_2\}\ c\ \{F_1' \wedge F_2'\}}[\text{Conj}]$$

$$\frac{\{F_1\}\ c\ \{F_2\} \quad \text{free}(F_3) \cap \text{mod}(c) = \emptyset}{\{F_1 \odot F_3\}\ c\ \{F_2 \odot F_3\}}[\text{QFrame}]$$

$$\frac{\forall i \in I.\ \{D_i\}\ c\ \{D_i'\} \quad \sum_{i\in I} p_i = 1}{\{\oplus_{i\in I} p_i \cdot D_i\}\ c\ \{\oplus_{i\in I} p_i \cdot D_i'\}}[\text{Sum}]$$

Table 5. Inference rules for classical statements

$$\frac{}{\{\mathbf{true}\}\ \overline{q} := |0\rangle\ \ \{|0\rangle_{\overline{q}}\}}[\text{QInit}] \qquad \frac{}{\{U_{\overline{q}}^{\dagger} F\}\ U[\overline{q}]\ \{F\}}[\text{QUnit}]$$

$$\frac{M = \{M_i\}_{i\in I} \quad p_i = \|M_{i\overline{q}}|v\rangle\|^2}{\{\wedge_i (P_i[i/x] \wedge |v\rangle_{\overline{q}\overline{q'}})\}\ x := M[\overline{q}]\ \{\oplus_i p_i \cdot ((P_i \wedge M_{i\overline{q}}|v\rangle_{\overline{q}\overline{q'}}/\sqrt{p_i}))\}}[\text{QMeas}]$$

Table 6. Inference rules for quantum statements

use the Hoare triple $\{D_1\}\ S\ \{D_2\}$ to express the correctness of our programs, where S is a program and D_1, D_2 are the assertions specified in Table 3.

Table 5 lists the rules for classical statements, with most of them being standard and thus self-explanatory. The assertion $D[a/x]$ is the same as D except that all the free occurrences of variable x in D are replaced by a. The assertion $D_1 \Rightarrow D_2$ indicates that D_1 logically implies D_2. The quantum frame rule [QFrame] in Table 5 is introduced for local reasoning, which allows us to add assertion F_3 to the pre/post-conditions of the local proof $\{F_1\}\ S\ \{F_2\}$. We use $\text{mod}(c)$ to denote the set of all classical and quantum variables modified by c. Then $\text{free}(F_3) \cap \text{mod}(c) = \emptyset$ indicates that all the free classical and quantum variables in F_3 are not modified by program c. Note that when F_3 is a classical assertion, $\odot$ can be replaced by $\wedge$. The rule [Sum] allows us to reason about a probability distribution by considering each pure state individually.

Table 6 displays the inference rules for quantum statements. In rule [QInit] we see that the execution of the command $\overline{q} := |0\rangle_{\overline{q}}$ sets the quantum system $\overline{q}$

$$\frac{}{F \vdash \mathbf{true}}[\mathrm{PT}] \qquad \frac{}{F \odot \mathbf{true} \dashv\vdash F}[\mathrm{OdotE}]$$

$$\frac{}{F_1 \odot F_2 \dashv\vdash F_2 \odot F_1}[\mathrm{OcotC}] \qquad \frac{}{F_1 \odot (F_2 \odot F_3) \dashv\vdash (F_1 \odot F_2) \odot F_3}[\mathrm{OdotA}]$$

$$\frac{}{P_1 \odot P_2 \dashv\vdash P_1 \wedge P_2}[\mathrm{OdotO}] \qquad \frac{}{P \odot F \dashv\vdash P \wedge F}[\mathrm{OdotOP}]$$

$$\frac{}{P \wedge (F_1 \odot F_2) \dashv\vdash (P \wedge F_1) \odot F_2}[\mathrm{OdotOA}]$$

$$\frac{}{F_1 \odot (F_2 \wedge F_3) \dashv\vdash (F_1 \odot F_2) \wedge (F_1 \odot F_3)}[\mathrm{OdotOC}]$$

$$\frac{}{|u\rangle_{\overline{p}}|v\rangle_{\overline{q}} \dashv\vdash |v\rangle_{\overline{q}}|u\rangle_{\overline{p}}}[\mathrm{ReArr}] \qquad \frac{}{|u\rangle_{\overline{p}}|v\rangle_{\overline{q}} \dashv\vdash |u \otimes v\rangle_{\overline{pq}}}[\mathrm{Separ}]$$

$$\frac{}{|u\rangle_{\overline{p}}|v\rangle_{\overline{q}} \dashv\vdash |u\rangle_{\overline{p}} \odot |v\rangle_{\overline{q}}}[\mathrm{OdotT}]$$

$$\frac{}{p_0 \cdot F \oplus p_1 \cdot F \oplus p_2 \cdot F' \dashv\vdash (p_0 + p_1) \cdot F \oplus p_2 \cdot F'}[\mathrm{OMerg}]$$

$$\frac{}{\oplus_{i \in I} p_i \cdot F_i \vdash \oplus_{i \in I} F_i}[\mathrm{Oplus}] \qquad \frac{\forall i \in I,\ F_i \vdash F_i'}{\oplus_{i \in I} p_i \cdot F_i \vdash \oplus_{i \in I} p_i \cdot F_i'}[\mathrm{OCon}]$$

Table 7. Inference rules for entailment reasoning

to $|0\rangle$, no matter what is the initial state. In rule [QUnit], for the postcondition F we have the precondition $U_{\overline{q}}^{\dagger}F$. Here $U_{\overline{q}}^{\dagger}$ distributes over those connectives of state formulas and eventually applies to quantum expressions. For example, if $F = |v\rangle_{\overline{qq'}} \wedge P$, then $U_{\overline{q}}^{\dagger}F = U_{\overline{q}}^{\dagger}(|v\rangle_{\overline{qq'}}) \wedge P$ and $U_{\overline{q}}^{\dagger}(\neg F) = \neg(U_{\overline{q}}^{\dagger}F)$. In rule [QMeas], the combined state of the variables $\overline{q}\overline{q'}$ is specified because there may be an entanglement between the subsystems for $\overline{q}$ and $\overline{q'}$. In that rule, we write $P_i[i/x]$ for the assertion obtained from P_i by replacing the variable x with value i. The postcondition is a distribution formula, with each assertion $P_i \wedge (M_{i[\overline{q}]}|v\rangle / \sqrt{p_i})$ assigned probability p_i, i.e., the probability of obtaining outcome i after the measurement M.

Table 7 presents several rules for entailment reasoning about quantum predicates. The notation $D_1 \vdash D_2$ says that D_1 proves D_2. Intuitively, it means that any state satisfying D_1 also satisfies D_2. We write $D_1 \dashv\vdash D_2$ if the other direction also holds.

The connective $\odot$ is commutative and associative, according to the rules [OdotC] and [OdotA]. If one or two assertions are classical, the rules [OdotO] and [OdotOP] replace $\odot$ with $\wedge$. The rule [OdotOA] replaces $P \wedge (F_1 \odot F_2)$ with $(P \wedge F_1) \odot F_2$ and vice versa. The rule [OdotOC] assists us to distribute $\odot$ into conjunctive assertions. The rule [ReArr] allows us to rearrange quantum expressions while [Separ] allows us to split/join the quantum expressions, given that $\overline{p}$ and $\overline{q}$ are not entangled with each other. The rules [ReArr] and [Separ] can be obtained naturally via the properties of tensor products. The rule [OdotT] replaces the $\odot$ connective with $\otimes$ when both assertions are state expressions. The rule [OMerg] allows us to merge the probabilities of two branches in a distribution formula if the two branches are the same. The rule [Oplus] is easy

to be understood as $\oplus_{i\in I} F_i$ is essentially a relaxed form of $\oplus_{i\in I} p_i \cdot F_i$. The rule [OCon] says that if each F_i entails F_i', then the entailment relation is preserved between the combinations $\oplus_{i\in I} p_i \cdot F_i$ and $\oplus_{i\in I} p_i \cdot F_i'$.

We use the notation $\vdash \{D_1\}\, c\, \{D_2\}$ to mean that the Hoare triple $\{D_1\}\, c\, \{D_2\}$ is provable by applying the rules in Tables 5–7.

Example 3. Suppose there are two separable systems $\bar{q}$ and $\bar{q'}$. They satisfy the precondition $\frac{1}{2}|u_1\rangle_{\bar{q}}|v_1\rangle_{\bar{q'}} \oplus \frac{1}{2}|u_2\rangle_{\bar{q}}|v_2\rangle_{\bar{q'}}$. After applying the operator $U[\bar{q}]$, they satisfy the postcondition $\frac{1}{2}(U_{\bar{q}}|u_1\rangle_{\bar{q}})|v_1\rangle_{\bar{q'}} \oplus \frac{1}{2}(U_{\bar{q}}|u_2\rangle_{\bar{q}})|v_2\rangle_{\bar{q'}}$. In other words, the following Hoare triple holds.

$$\begin{array}{l} \vdash \{\frac{1}{2}|u_1\rangle_{\bar{q}}|v_1\rangle_{\bar{q'}} \oplus \frac{1}{2}|u_2\rangle_{\bar{q}}|v_2\rangle_{\bar{q'}}\} \\ \quad U[\bar{q}] \\ \quad \{\frac{1}{2}(U_{\bar{q}}|u_1\rangle_{\bar{q}})|v_1\rangle_{\bar{q'}} \oplus \frac{1}{2}(U_{\bar{q}}|u_2\rangle_{\bar{q}})|v_2\rangle_{\bar{q'}}\} \end{array} \tag{1}$$

This Hoare triple can be proved as follows. Firstly, we apply the rule [QUnit] to obtain

$$\vdash \{|u_1\rangle_{\bar{q}}\}\ U[\bar{q}]\ \{|U_{\bar{q}}|u_1\rangle_{\bar{q}}\}. \tag{2}$$

Then we use the rules [QFrame] and [OdotT] to get

$$\vdash \{|u_1\rangle_{\bar{q}}|v_1\rangle_{\overline{q'}}\}\ U[\bar{q}]\ \{|U_{\bar{q}}|u_1\rangle_{\bar{q}}|v_1\rangle_{\overline{q'}}\}. \tag{3}$$

Similarly, we have

$$\vdash \{|u_2\rangle_{\bar{q}}|v_2\rangle_{\overline{q'}}\}\ U[\bar{q}]\ \{|U_{\bar{q}}|u_2\rangle_{\bar{q}}|v_2\rangle_{\overline{q'}}\}. \tag{4}$$

Combining (3) with (4) by rule [Sum], we obtain the Hoare triple in (1). □

Example 4. Let us consider the program **addM** and the distribution formula $D = \frac{1}{2}\cdot(v=1) \oplus \frac{1}{2}\cdot(v\neq 1)$ discussed in Example 2. It can be formally proved that

$$\{\mathbf{true}\}\ \mathbf{addM}\ \{D\}\ .$$

A proof outline is given in [8]. Following [14], we use the following notations to highlight the application of the frame rule [QFrame]:

$$\Longleftrightarrow \{\ F_1\ \}\ c\ \{\ F_2\ \} \qquad \text{or equivalently} \qquad \begin{array}{c} \Longrightarrow \{\ F_1\ \} \\ c \\ \Longleftarrow \{\ F_2\ \} \end{array}$$

Both notations indicate that $\{F_1\}c\{F_2\}$ is a local proof for c and is useful for long proofs. The frame assertion F_3 can be deduced from the assertions before F_1 or after F_2. □

Our inference system is sound in the following sense: any Hoare triple in the form $\{D_1\}\ c\ \{D_2\}$ derived from the inference system is valid, denoted by $\vDash \{D_1\}c\{D_2\}$, meaning that for any state μ we have $\mu \models D_1$ implies $[\![c]\!]_\mu \models D_2$.

Theorem 1 (Soundness). *If* $\vdash \{D_1\}\ c\ \{D_2\}$ *then* $\vDash \{D_1\}\ c\ \{D_2\}$.

5 Examples

We apply the proof system to verify the functional correctness of two non-trivial algorithms: the HHL and Shor's algorithms in Subsections 5.1 and 5.2, respectively.

Notice that the correctness of the HHL algorithm was verified in [34] where projections are used as assertions. The Shor's algorithm was verified in [10] with a quantitative interpretation of assertions. As we will see, our verification for both algorithms employs a qualitative reasoning, which is more in the style of the classical Hoare logic.

5.1 HHL Algorithm

The HHL algorithm [11] aims to obtain a vector x such that $Ax = b$, where A is a given Hermitian operator and b is a given vector. Suppose that A has the spectral decomposition $A{=}\Sigma_j \lambda_j |j\rangle \langle j|$, where each λ_j is an eigenvalue and $|j\rangle$ is the corresponding eigenvector of A. On the basis $\{|j\rangle\}_{j\in J}$, we have $A^{-1} = \Sigma_j \lambda_j^{-1} |j\rangle \langle j|$ and $|b\rangle{=}\Sigma_j b_j |j\rangle$. Then the vector x can be expressed as $|x\rangle{=}A^{-1} |b\rangle = \Sigma_j \lambda_j^{-1} b_j |j\rangle$. Here we require that $|j\rangle, |b\rangle$ and $|x\rangle$ are all normalized vectors. Hence, we have

$$\sum_j |\lambda_j^{-1} b_j|^2 = 1. \tag{5}$$

A quantum program implementing the HHL algorithm is presented in Table 8. The n-qubit subsystem $\overline{p}$ is used as a control system in the phase estimation step with $N = 2^n$. The m-qubit subsystem $\overline{q}$ stores the vector $|b\rangle = \sum_i b_i |i\rangle$. The one-qubit subsystem r is used to control the while loop. The measurement $M = \{M_0, M_1\}$ in the loop is the simplest two-value measurement: $M_0 = |0\rangle_r \langle 0|$ and $M_1 = |1\rangle_r \langle 1|$. The results of measurements will be assigned to the classical variable v, which is initialized to be 0. If the value of v is 0, then the while loop is repeated until it is 1. The unitary operator U_b is assumed to map $|0\rangle^{\otimes m}$ to $|b\rangle$. The controlled unitary operator U_f has a control system $\overline{p}$ and a target system $\overline{q}$, that is,

$$U_f = \sum_{\tau=0}^{N-1} |\tau\rangle_p \langle \tau| \otimes U^\tau,$$

where $U = e^{iAt}$. Equivalently, we have $U |j\rangle{=}e^{i\lambda_j t} |j\rangle$. We denote $\phi_j{=}\frac{\lambda_j t}{2\pi}$ and $\widetilde{\phi_j}{=}\phi_j{\cdot}N$, then we have $U |j\rangle{=}e^{2\pi i \phi_j} |j\rangle$. A given controlled unitary operator U_c has control system $\overline{p}$ and target system r, more precisely,

$$U_c |0\rangle_{\overline{p}} |0\rangle_r = |0\rangle_{\overline{p}} |0\rangle_r, \qquad U_c |j\rangle_{\overline{p}} |0\rangle_r = |j\rangle_{\overline{p}} (\sqrt{1 - \tfrac{C^2}{j^2}} |0\rangle + \tfrac{C}{j} |1\rangle)_r,$$

where $1 \le j \le N-1$ and C is a given parameter.

The symbol $H^{\otimes n}$ represents n Hadamard gates applied to the variables $\overline{q}$; QFT and QFT^{-1} are the quantum Fourier transform and the inverse quantum Fourier transform acting on the variables in $\overline{p}$.

HHL $\triangleq$

$1: v := 0$
$2:$ **while** $v = 0$ **do**
$3:$ $\overline{p} := |0\rangle^{\otimes n}$;
$4:$ $\overline{q} := |0\rangle^{\otimes m}$;
$5:$ $r := |0\rangle$;
$6:$ $U_b[\overline{q}]$;
$7:$ $H^{\otimes n}[\overline{p}]$;
$8:$ $U_f[\overline{pq}]$;
$9:$ $\mathrm{QFT}^{-1}[\overline{p}]$;
$10:$ $U_c[\overline{p}r]$;
$11:$ $\mathrm{QFT}[\overline{p}]$;
$12:$ $U_f^{\dagger}[\overline{pq}]$;
$13:$ $H^{\otimes n}[\overline{p}]$;
$14:$ $v := M[r]$ **od**

Table 8. A quantum program for the HHL algorithm

The correctness of the HHL algorithm can be specified by the Hoare triple:

$$\{\ \mathbf{true}\ \}\ \mathbf{HHL}\ \{\ |x\rangle_{\overline{q}}\ \}\ .$$

Now let $D \triangleq (v = 0) \oplus ((|0\rangle_{\overline{p}}^{\otimes n} |x\rangle_{\overline{q}} |1\rangle_r) \wedge (v = 1))$, and S the body of the while loop of the HHL algorithm. The following Hoare triple can be proved.

$$\{\ v = 0\ \}\ S\ \{\ D\ \}\ .$$

So D is an invariant of the while loop of the HHL algorithm. Then by rule [While] we obtain that

$$\{\ D\ \}\ \mathbf{while}\ (v = 0)\ \mathbf{do}\ S\ \mathbf{od}\ \{\ |0\rangle_{\overline{p}}^{\otimes n} |x\rangle_{\overline{q}} |1\rangle_r \wedge (v = 1)\ \}\ .$$

Finally, we can establish the correctness of the HHL algorithm as given in Table 9, where we highlight the invariant of the while loop in red.

5.2 Shor's Algorithm

Shor's algorithm relies on the order-finding algorithm [24]. So we first verify the correctness of the latter. Given two co-prime positive integers x and N, the smallest positive integer r that satisfies the equation $x^r = 1(\mathrm{mod})N$ is called the order of x modulo N, denoted by $\mathrm{ord}(x, N)$. The problem of order-finding is to find the order r defined above, which is solved by the program presented in Table 10. Let $L \triangleq \lceil \log(N) \rceil$, $\epsilon \in (0, 1)$ and $t \triangleq 2L + 1 + \lceil \log(2 + 1/2\epsilon) \rceil$. The

$\{$ **true** $\}$
$\{$ $(v = 0)[0/v]$ $\}$ by rule [Conseq]
$v := 0;$
$\{$ $v = 0$ $\}$ by rule [Assgn]
$\{$ $(v = 0) \oplus (|0\rangle_{\overline{p}}^{\otimes n} |x\rangle_{\overline{q}} |1\rangle_r \wedge (v = 1))$ $\}$ by rule [Oplus]
while $v = 0$ **do**
  $\{$ $v = 0$ $\}$
  S
  $\{$ $(v = 0) \oplus (|0\rangle_{\overline{p}}^{\otimes n} |x\rangle_{\overline{q}} |1\rangle_r \wedge (v = 1))$ $\}$
od
$\{$ $|0\rangle_{\overline{p}}^{\otimes n} |x\rangle_{\overline{q}} |1\rangle_r \wedge (v = 1)$ $\}$ by rule [While]
$\{$ $|0\rangle_{\overline{p}}^{\otimes n} |x\rangle_{\overline{q}} |1\rangle_r$ $\}$ by rule [Conseq]
$\{$ $|0\rangle_{\overline{p}}^{\otimes n} \odot |x\rangle_{\overline{q}} \odot |1\rangle_r$ $\}$ by rule [OdotT]
$\{$ **true** $\odot |x\rangle_{\overline{q}} \odot$ **true** $\}$ by rule [PT]
$\{$ $|x\rangle_{\overline{q}}$ $\}$ by rule [OdotE]

Table 9. Proof outline of the HHL algorithm

order-finding algorithm can successfully obtain the order of x with probability at least $(1-\epsilon)/2\log(N)$, by using $\mathrm{O}(L^3)$ operations as discussed in [19].

The variables in $\overline{q}$ correspond to a t-qubit subsystem while $\overline{p}$ represents an L-qubit subsystem. We introduce the variable z to store the order computed by the program **OF**, and initialize it to 1. The unitary operator U_+ maps $|0\rangle$ to $|1\rangle$, that is $U_+ |0\rangle = |1\rangle$. The notation $H^{\otimes t}$ means t Hadamard gates applied to the system $\overline{p}$ and QFT^{-1} is the inverse quantum Fourier transform. The function $f(x)$ stands for the continued fraction algorithm which returns the minimal denomination n of all convergents m/n of the continued fraction for x with $|m/n - x| < 1/(2n^2)$ [10]. The unitary operator CU is the controlled-U, with $\overline{q}$ being the control system and $\overline{p}$ the target system, that is $\mathrm{CU}\ |i\rangle_{\overline{q}} |j\rangle_{\overline{p}} = |i\rangle_{\overline{q}}\, U^i\, |j\rangle_{\overline{p}}$, where for each $0 \leq y \leq 2^L$,

$$U\,|y\rangle = \begin{cases} |xy \bmod N\rangle & \text{if } y \leq N \\ |y\rangle & \text{otherwise.} \end{cases}$$

Note that the states defined by

$$|u_s\rangle \triangleq \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i s k/r}\, |x^k \bmod N\rangle$$

for integer $0 \leq s \leq r-1$ are eigenstates of U and $\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = 1$.

The variable b stores the value of $(x^z \bmod N)$, and the operator $(a \bmod b)$ computes the modulo of a divided by b. If the value of b is not equal to 1, which

$\mathbf{OF}(x,N) :\equiv$

1 : $z := 1$;
2 : $b := x^z \pmod N$;
3 : **while** $(b \neq 1)$ **do**
4 : $\quad \overline{q} := |0\rangle^{\otimes t}$;
5 : $\quad \overline{p} = |0\rangle^{\otimes L}$;
6 : $\quad H^{\otimes t}[\overline{q}]$;
7 : $\quad U_+[\overline{p}]$;
8 : $\quad \mathrm{CU}[\overline{qp}]$;
9 : $\quad \mathrm{QFT}^{-1}[\overline{q}]$;
10 : $\quad z' := M[\overline{q}]$;
11 : $\quad z := f(\frac{z'}{2^t})$;
12 : $\quad b := x^z \pmod N$ **od**

Table 10. A quantum program for the order-finding algorithm

means that the value of z computed by **OF** is not equal to the actual order of x modulo N, then the program will repeat the body of the while loop until $b = 1$. The while loop in the **OF** program exhibits probabilistic behaviour due to a measurement in the loop body.

The correctness of the order-finding algorithm can be specified as

$$\{\ 2 \leq x \leq N-1 \wedge \gcd(x,N) = 1 \wedge N \bmod 2\ \neq 0\ \}\ \mathbf{OF}\ \{\ z = r\ \}\ .$$

Now let $D' \triangleq (z = r \wedge b = 1) \oplus (z \neq r \wedge b \neq 1)$, and S' the body of the while loop **OF**. We can establish the correctness of S' as follows:

$$\{\ z \neq r \wedge b \neq 1\ \}\ S'\ \{\ D'\ \}\ .$$

So the invariant of the while loop of **OF** can be D', and by rule [While] we have

$$\{\ D'\ \}\ \mathbf{while}\ (b \neq 1)\ \mathbf{do}\ S'\ \mathbf{od}\ \{\ z = r \wedge b = 1\ \}\ .$$

Finally, a proof outline of **OF** is given in Table 11 .

Then we introduce Shor's algorithm in Table 12. The function random(a,b) is used to randomly generate a number between a and b. The function gcd(a,b) returns the greatest common divisor of a and b. The operator $\equiv_N$ represents identity modulo N. **OF**(x,N) is the order-finding algorithm given before, which will return the order of x modulo N and assign the order to the classical variable z. The classical variable y stores one of the divisors of N and we use $y|N$ to represent that N is divisible by y.

$\{2 \leq x \leq N-1\}$
$\Longrightarrow \{(z=1)[1/z]\}$
$z := 1;$
$\Longleftarrow \{z=1\}$ by rule [Assgn]
$\Longrightarrow \{(z=1 \wedge b = x^1 \bmod N)[(x^z \bmod \mathrm{N})/b]\}$
$b := x^z \bmod N;$
$\Longleftarrow \{z=1 \wedge b = x^1 \bmod N\}$ by rule [Assgn]
$\Longrightarrow \{z \neq \mathrm{ord}(x,N) \wedge b \neq 1\}$ by rule [Conseq]
$\{(z=\mathrm{ord}(x,N) \wedge b=1) \oplus (z \neq \mathrm{ord}(x,N) \wedge b \neq 1)\}$ by rule [Conseq]
while $(b \neq 1)$ **do**
  $\{z \neq \mathrm{ord}(x,N) \wedge b \neq 1\}$
  S'
  $\{(z=\mathrm{ord}(x,N) \wedge b=1) \oplus (z \neq \mathrm{ord}(x,N) \wedge b \neq 1)\}$
od
$\Longleftarrow \{z=\mathrm{ord}(x,N) \wedge b=1\}$ by rule [While]
$\{2 \leq x \leq N-1 \wedge z=\mathrm{ord}(x,N)\}$ by rule [Conseq]

Table 11. Proof outline of the **OF** program

The correctness of Shor's algorithm can be specified by the Hoare triple:

$$\{\ \mathrm{cmp}(N)\ \}\ \mathbf{Shor}\ \{\ y|N \wedge y \neq 1 \wedge y \neq N\ \}$$

where $\mathrm{cmp}(N)$ is a predicate stating that N is a composite number greater than 0. The invariant of the while loop in Shor's algorithm can be

$$y|N \wedge y \neq N\ .$$

A proof outline for the correctness of the algorithm is given in [8].

6 Conclusion and Future Work

We have presented a new quantum Hoare logic for classical–quantum programs. It includes distribution formulas for specifying probabilistic properties of classical assertions naturally, and at the same time allows for local reasoning. We have proved the soundness of the logic with respect to a denotational semantics and exhibited its usefulness in reasoning about the functional correctness of the HHL and Shor's algorithms, which are non-trivial algorithms involving probabilistic behaviour due to quantum measurements and unbounded while loops.

We have not yet precisely delimited the expressiveness of our logic. It is unclear whether the logic is relatively complete, which is an interesting future

```
1:  if 2 | N then
2:      y := 2;
3:  else
4:      x := random(2, N − 1);
5:      y := gcd(x, N);
6:      while y = 1 do
7:          z := OF(x, N);
8:          if 2 | z and x^{z/2} ≢_N −1 then
9:              y′ := gcd(x^{z/2} − 1, N);
10:             if 1 < y′ < N then
11:                 y := y′;
12:             else
13:                 y := gcd(x^{z/2} + 1, N);
14:             fi
15:         else
16:             x := random(2, N − 1);
17:             y := gcd(x, N);
18:         fi
19:     od
20: fi
```

Table 12. A program for Shor's algorithm

work to consider. We would also like to embed the logic into a proof assistant so to alleviate the burden of manually reasoning about quantum programs as done in Section 5.

Usually there are two categories of program logics when dealing with probabilistic behaviour: satisfaction-based or expectation-based [7]. Our logic belongs to the first category. In an expectation-based logic, e.g. the logic in [28,29], the Hore triple $\{P\}c\{Q\}$ is valid in the sense that the expectation of an initial state satisfying P is a lower bound of the expectation of the final state satisfying Q. It would be interesting to explore local reasoning in expectation-based logics for classical–quantum programs such as those proposed in [10,9].

References

1. Barthe, G., Hsu, J., Liao, K.: A probabilistic separation logic. Proc. ACM Program. Lang. **4**(POPL), 55:1–55:30 (2020)
2. Barthe, G., Hsu, J., Ying, M., Yu, N., Zhou, L.: Relational proofs for quantum programs. Proceedings of the ACM on Programming Languages **4**(POPL), 1–29 (2019)
3. Batz, K., Kaminski, B.L., Katoen, J., Matheja, C., Noll, T.: Quantitative separation logic: a logic for reasoning about probabilistic pointer programs. Proc. ACM Program. Lang. **3**(POPL), 34:1–34:29 (2019)
4. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions. Springer (2004)

5. Brookes, S.: A semantics for concurrent separation logic. Theoretical Computer Science **375**(1-3), 227–270 (2007)
6. Chadha, R., Mateus, P., Sernadas, A.: Reasoning about imperative quantum programs. Electronic Notes in Theoretical Computer Science **158**, 19–39 (2006)
7. Deng, Y., Feng, Y.: Formal semantics of a classical-quantum language. Theoretical Computer Science **913**, 73–93 (2022)
8. Deng, Y., Wu, H., Xu, M.: Local reasoning about probabilistic behaviour for classical–quantum programs (full version), `https://arxiv.org/abs/2308.04741`
9. Feng, Y., Li, S., Ying, M.: Verification of distributed quantum programs. ACM Trans. Comput. Log. **23**(3), 19:1–19:40 (2022)
10. Feng, Y., Ying, M.: Quantum Hoare logic with classical variables. ACM Transactions on Quantum Computing **2**(4), 16:1–16:43 (2021)
11. Harrow, A.W., Hassidim, A., Lloyd, S.: Quantum algorithm for linear systems of equations. Physical review letters **103**(15), 150502 (2009)
12. Hoare, C.A.R.: An axiomatic basis for computer programming. Communications of the ACM **12**(10), 576–580 (1969)
13. Kakutani, Y.: A logic for formal verification of quantum programs. In: Proceedings of the 13th Asian Computing Science Conference. Lecture Notes in Computer Science, vol. 5913, pp. 79–93. Springer (2009)
14. Le, X.B., Lin, S.W., Sun, J., Sanan, D.: A quantum interpretation of separating conjunction for local reasoning of quantum programs based on separation logic. Proc. ACM Program. Lang. **6**(POPL) (jan 2022)
15. Li, J.M., Ahmed, A., Holtzen, S.: Lilac: A modal separation logic for conditional probability. Proc. ACM Program. Lang. **7**(PLDI), 148–171 (2023)
16. Li, Y., Unruh, D.: Quantum Relational Hoare Logic with Expectations. In: 48th International Colloquium on Automata, Languages, and Programming. Leibniz International Proceedings in Informatics (LIPIcs), vol. 198, pp. 136:1–136:20 (2021)
17. Liu, J., Zhan, B., Wang, S., Ying, S., Liu, T., Li, Y., Ying, M., Zhan, N.: Formal verification of quantum algorithms using quantum Hoare logic. In: International conference on computer aided verification. Lecture Notes in Computer Science, vol. 11562, pp. 187–207. Springer (2019)
18. von Neumann, J.: States, Effects and Operations: Fundamental Notions of Quantum Theory. Princeton University Press (1955)
19. Nielsen, M., Chuang, I.: Quantum computation and quantum information. Cambridge university press (2000)
20. Nipkow, T., Wenzel, M., Paulson, L.C.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic. Springer (2002)
21. O'Hearn, P.W.: Resources, concurrency, and local reasoning. Theoretical Computer Science **375**(1-3), 271–307 (2007)
22. Reynolds, J.C.: Separation logic: A logic for shared mutable data structures. In: Proceedings of the 17th IEEE Symposium on Logic in Computer Science. pp. 55–74. IEEE Computer Society (2002)
23. Selinger, P.: Towards a quantum programming language. Mathematical Structures in Computer Science **14**(4), 527–586 (2004)
24. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science. pp. 124–134. IEEE Computer Society (1994)
25. Tassarotti, J., Harper, R.: A separation logic for concurrent randomized programs. Proc. ACM Program. Lang. **3**(POPL), 64:1–64:30 (2019)

26. Unruh, D.: Quantum Hoare logic with ghost variables. In: Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 1–13. IEEE (2019)
27. Unruh, D.: Quantum relational Hoare logic. Proceedings of the ACM on Programming Languages **3**(POPL), 1–31 (2019)
28. Ying, M.: Floyd–Hoare logic for quantum programs. ACM Transactions on Programming Languages and Systems **33**(6), 1–49 (2012)
29. Ying, M.: Foundations of Quantum Programming. Morgan Kaufmann (2016)
30. Ying, M., Feng, Y.: A flowchart language for quantum programming. IEEE Trans. Software Eng. **37**(4), 466–485 (2011)
31. Ying, M., Zhou, L., Li, Y., Feng, Y.: A proof system for disjoint parallel quantum programs. Theor. Comput. Sci. **897**, 164–184 (2022)
32. Zhou, L., Barthe, G., Hsu, J., Ying, M., Yu, N.: A quantum interpretation of bunched logic & quantum separation logic. In: Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 1–14. IEEE (2021)
33. Zhou, L., Barthe, G., Strub, P.Y., Liu, J., Ying, M.: Coqq: Foundational verification of quantum programs. Proc. ACM Program. Lang. **7**(POPL) (2023)
34. Zhou, L., Yu, N., Ying, M.: An applied quantum Hoare logic. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 1149–1162 (2019)