

Verification of Nondeterministic Quantum Programs

Yuan Feng

Yuan.Feng@uts.edu.au
University of Technology Sydney
Australia

Yingte Xu

Yingte.Xu@student.uts.edu.au
University of Technology Sydney
Australia

ABSTRACT

Nondeterministic choice is a useful program construct that provides a way to describe the behaviour of a program without specifying the details of possible implementations. It supports the stepwise refinement of programs, a method that has proven useful in software development. Nondeterminism has also been introduced in quantum programming, and termination of nondeterministic quantum programs has been extensively analysed. In this paper, we go beyond termination analysis to investigate the verification of nondeterministic quantum programs where properties are given by sets of hermitian operators on the associated Hilbert space. Hoare-type logic systems for partial and total correctness are proposed which turn out to be both sound and relatively complete with respect to their corresponding semantic correctness. To show the utility of these proof systems, we analyse some quantum algorithms such as quantum error correction scheme, Deutsch algorithm, and a nondeterministic quantum walk. Finally, a proof assistant prototype is implemented to aid in the automated reasoning of nondeterministic quantum programs.

CCS CONCEPTS

• **Theory of computation** → **Denotational semantics; Axiomatic semantics; Hoare logic; Program verification; Pre- and post-conditions; Assertions.**

KEYWORDS

Quantum programming, nondeterminism, program verification, Hoare logic

ACM Reference Format:

Yuan Feng and Yingte Xu. 2023. Verification of Nondeterministic Quantum Programs. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '23), March 25–29, 2023, Vancouver, BC, Canada*. ACM, New York, NY, USA, 18 pages. <https://doi.org/10.1145/3582016.3582039>

1 INTRODUCTION

The introduction of nondeterminism into a programming language allows one to describe both a specification and its possible implementations within the same language. It naturally supports the technique of stepwise refinement of programs, which has proven

useful in the development of computer software [1, 17, 19]. From the verification perspective, it provides a way to reason about the correctness of an abstract specification before it is fully implemented [5], making it possible to detect errors in the earliest stages of software development.

Programs with both nondeterministic and probabilistic choices have been investigated in [8, 14, 15, 18]. Interestingly, as pointed out in [8], there are two natural models to describe the way nondeterminism interacts with probabilistic choice. This difference can be best illustrated by regarding the execution of a nondeterministic and probabilistic program as a game against a malicious adversary. In the relational model, the adversary makes his nondeterministic choice during runtime execution, taking advantage of all the probabilistic choices already made up to that point. By contrast, the adversary in the lifted model must make all the nondeterministic choices in compile-time, before any real execution of the program. It was shown in [8] that programs in the relational model enjoy many nice algebraic properties which are useful in static analysis and compiler design [9].

Nondeterminism was also introduced into quantum programming many years ago. Zuliani [30] extended the quantum Guarded-Command Language qGCL [22] with a nondeterministic choice construct, for the purpose of modelling and reasoning about counterfactual computations [16] and quantum mixed-state systems. Termination analysis of nondeterministic quantum programs was first studied in [12], where programs are described in a language-independent way: a nondeterministic quantum program simply consists of a set of deterministic quantum Markov chains (mathematically expressed as super-operators) with the same Hilbert space and a two-outcome measurement to determine whether the program has terminated. Later on, termination analysis was extended to a quantum programming language which involves both demonic and angelic nondeterminism, using the technique of linear ranking super-martingale [11].

Despite the termination analysis in [11, 12], the verification of nondeterministic quantum programs is rarely addressed in the literature. In this paper, we consider an extended while-language for quantum programs where a construct for binary (demonic) nondeterministic choice is included. The main contribution includes:

- A denotational semantics for nondeterministic quantum programs, which resembles the lifted model in the probabilistic setting. The rationale behind this design decision is an observation that the relational model cannot be satisfactorily defined in the quantum setting, partly due to the non-uniqueness of ensembles of pure states associated with a given density operator.
- The partial and total correctness of nondeterministic quantum programs based on their denotational semantics. Compared with [11, 12], our notions of correctness are much

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ASPLOS '23, March 25–29, 2023, Vancouver, BC, Canada

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-9918-0/23/03...\$15.00

<https://doi.org/10.1145/3582016.3582039>

broader, capable of describing a wider range of properties beyond termination of quantum programs.

- Hoare-type logic systems which are both sound and relatively complete with respect to partial/total correctness. Illustrating examples are explored and a prototyping tool is implemented which show the utility of these logic systems in proving the correctness of some quantum algorithms such as quantum error correction scheme, Deutsch algorithm, and a nondeterministic quantum walk.

The remaining part of the paper is organised as follows. In Sec. 2 we review some basic notions from quantum computing that will be used in later discussion. The syntax and denotational semantics of nondeterministic quantum programs considered in this paper are given in Sec. 3. Sec. 4 is the main part of the paper where we elaborate the notions of assertions for quantum states, and the partial and total correctness of nondeterministic quantum programs. Proof systems for these two correctness notions are proposed and shown to be both sound and relatively complete. Illustrative examples are explored in Sec. 5 to show the effectiveness of our proof systems. Sec. 6 is devoted to a proof assistant prototype implementing the verification techniques developed in this paper. Finally, Sec. 7 concludes the paper and points out some topics for future studies.

2 BACKGROUND ON QUANTUM COMPUTING

This section is devoted to fixing some notations from linear algebra and quantum mechanics that will be used in this paper. For a thorough introduction of relevant backgrounds, we refer to [20, Chapter 2].

Let \mathcal{H} be a finite-dimensional Hilbert space. Following the tradition of quantum computing, vectors in \mathcal{H} are denoted in the Dirac form $|\psi\rangle$. The inner and outer products of two vectors $|\psi\rangle$ and $|\phi\rangle$ are written as $\langle\psi|\phi\rangle$ and $|\psi\rangle\langle\phi|$ respectively. Let $\mathcal{L}(\mathcal{H})$ be the set of linear operators on \mathcal{H} . Denote by $\text{tr}(A) = \sum_{i \in I} \langle\psi_i|A|\psi_i\rangle$ the *trace* of $A \in \mathcal{L}(\mathcal{H})$ where $\{|\psi_i\rangle : i \in I\}$ is an (arbitrary) orthonormal basis of \mathcal{H} . The *adjoint* of A , denoted A^\dagger , is the unique linear operator in $\mathcal{L}(\mathcal{H})$ such that $\langle\psi|A|\phi\rangle = \langle\phi|A^\dagger|\psi\rangle^*$ for all $|\psi\rangle, |\phi\rangle \in \mathcal{H}$. Here, for a complex number z , z^* denotes its conjugate. An operator $A \in \mathcal{L}(\mathcal{H})$ is said to be (1) *hermitian* if $A^\dagger = A$; (2) *unitary* if $A^\dagger A = I_{\mathcal{H}}$, the identity operator on \mathcal{H} ; (3) *positive* if for all $|\psi\rangle \in \mathcal{H}$, $\langle\psi|A|\psi\rangle \geq 0$. Every hermitian operator A has a *spectral decomposition* form $A = \sum_{i \in I} \lambda_i |\psi_i\rangle\langle\psi_i|$ where $\{|\psi_i\rangle : i \in I\}$ constitute an orthonormal basis of \mathcal{H} . The Löwner (partial) order \sqsubseteq on $\mathcal{L}(\mathcal{H})$ is defined by letting $A \sqsubseteq B$ iff $B - A$ is positive.

Let \mathcal{H}_1 and \mathcal{H}_2 be two finite dimensional Hilbert spaces, and $\mathcal{H}_1 \otimes \mathcal{H}_2$ their tensor product. Let $A_i \in \mathcal{L}(\mathcal{H}_i)$. The tensor product of A_1 and A_2 , denoted $A_1 \otimes A_2$ is a linear operator in $\mathcal{L}(\mathcal{H}_1 \otimes \mathcal{H}_2)$ such that $(A_1 \otimes A_2)(|\psi_1\rangle \otimes |\psi_2\rangle) = (A_1|\psi_1\rangle) \otimes (A_2|\psi_2\rangle)$ for all $|\psi_i\rangle \in \mathcal{H}_i$. The definition extends linearly to $\mathcal{L}(\mathcal{H}_1 \otimes \mathcal{H}_2)$. To simplify notations, we often write $|\psi_1\rangle|\psi_2\rangle$ for $|\psi_1\rangle \otimes |\psi_2\rangle$.

A linear operator \mathcal{E} from $\mathcal{L}(\mathcal{H}_1)$ to $\mathcal{L}(\mathcal{H}_2)$ is called a *super-operator*. It is said to be (1) *positive* if it maps positive operators on \mathcal{H}_1 to positive operators on \mathcal{H}_2 ; (2) *completely positive* if $I_{\mathcal{H}_1} \otimes \mathcal{E}$ is positive for all finite dimensional Hilbert space \mathcal{H} , where $I_{\mathcal{H}}$ is the identity super-operator on $\mathcal{L}(\mathcal{H})$; (3) *trace-preserving* (resp.

trace-nonincreasing) if $\text{tr}(\mathcal{E}(A)) = \text{tr}(A)$ (resp. $\text{tr}(\mathcal{E}(A)) \leq \text{tr}(A)$) for any positive operator $A \in \mathcal{L}(\mathcal{H}_1)$.

From *Kraus representation theorem* [10], a super-operator \mathcal{E} from $\mathcal{L}(\mathcal{H}_1)$ to $\mathcal{L}(\mathcal{H}_2)$ is completely positive iff there are linear operators $\{E_i : i \in I\}$, called *Kraus operators* of \mathcal{E} , from \mathcal{H}_1 to \mathcal{H}_2 such that $\mathcal{E}(A) = \sum_{i \in I} E_i A E_i^\dagger$ for all $A \in \mathcal{L}(\mathcal{H}_1)$. Furthermore, \mathcal{E} is trace-preserving (resp. trace-nonincreasing) iff $\sum_{i \in I} E_i^\dagger E_i = I_{\mathcal{H}_1}$ (resp. $\sum_{i \in I} E_i^\dagger E_i \sqsubseteq I_{\mathcal{H}_1}$). The *adjoint* of \mathcal{E} , denoted \mathcal{E}^\dagger , is a completely positive super-operator from $\mathcal{L}(\mathcal{H}_2)$ back to $\mathcal{L}(\mathcal{H}_1)$ with Kraus operators being $\{E_i^\dagger : i \in I\}$. It is easy to check that for any $A \in \mathcal{L}(\mathcal{H}_1)$ and $B \in \mathcal{L}(\mathcal{H}_2)$, $\text{tr}(\mathcal{E}(A) \cdot B) = \text{tr}(A \cdot \mathcal{E}^\dagger(B))$.

In the following, for simplicity, all super-operators are assumed to be completely positive and trace-nonincreasing unless otherwise stated. Given the tensor product space $\mathcal{H}_1 \otimes \mathcal{H}_2$, the *partial trace* with respect to \mathcal{H}_2 , denoted $\text{tr}_{\mathcal{H}_2}$, is a super-operator from $\mathcal{L}(\mathcal{H}_1 \otimes \mathcal{H}_2)$ to $\mathcal{L}(\mathcal{H}_1)$ such that for any $|\psi_i\rangle, |\phi_i\rangle \in \mathcal{H}_i$, $i = 1, 2$,

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

Again, the definition extends linearly to $\mathcal{L}(\mathcal{H}_1 \otimes \mathcal{H}_2)$.

According to von Neumann's formalism of quantum mechanics [27], any quantum system with finite degrees of freedom is associated with a finite-dimensional Hilbert space \mathcal{H} called its *state space*. When the dimension of \mathcal{H} is 2, such a system is called a *qubit*, the analogy of bit in classical computing. A *pure state* of the system is described by a normalised vector in \mathcal{H} . When the system is in state $|\psi_i\rangle$ with probability p_i , $i \in I$, it is said to be in a *mixed state*, represented by the *density operator* $\sum_{i \in I} p_i |\psi_i\rangle\langle\psi_i|$ on \mathcal{H} . Obviously, a density operator is positive and has trace 1. In this paper, we follow Selinger's convention [23] to regard *partial density operators*, i.e. positive operators with traces not greater than 1 as (unnormalised) quantum states. Intuitively, a partial density operator ρ denotes a legitimate quantum state $\rho/\text{tr}(\rho)$ which is obtained with probability $\text{tr}(\rho)$. Denote by $\mathcal{D}(\mathcal{H})$ the set of partial density operators on \mathcal{H} . The state space of a composite system (e.g., a quantum system consisting of multiple qubits) is the tensor product of the state spaces of its components. For any ρ in $\mathcal{D}(\mathcal{H}_1 \otimes \mathcal{H}_2)$, the partial traces $\text{tr}_{\mathcal{H}_1}(\rho)$ and $\text{tr}_{\mathcal{H}_2}(\rho)$ are the reduced quantum states of ρ on \mathcal{H}_2 and \mathcal{H}_1 , respectively.

The *evolution* of a closed quantum system is described by a unitary operator on its state space: if the states of the system at t_1 and t_2 are ρ_1 and ρ_2 , respectively, then $\rho_2 = U\rho_1 U^\dagger$ for some unitary U . Typical unitary operators used throughout this paper include Pauli- X , Y , and Z , Hadamard H , and controlled-NOT (CNOT) operator CX , represented in the matrix form (with respect to the computational basis) respectively as follows:

$$X \triangleq \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y \triangleq \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, Z \triangleq \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

and

$$H \triangleq \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, CX \triangleq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

A (projective) quantum *measurement* \mathcal{M} is described by a collection $\{P_i : i \in O\}$ of projectors (hermitian operators with eigenvalues being either 0 or 1) in the state space \mathcal{H} , where O is the set of

measurement outcomes. It is required that the measurement operators P_i 's satisfy the completeness equation $\sum_{i \in \mathcal{O}} P_i = I_{\mathcal{H}}$. If the system was in state ρ before measurement, then the probability of observing outcome i is given by $p_i = \text{tr}(P_i \rho)$, and the state of the post-measurement system becomes $\rho_i = P_i \rho P_i / p_i$ whenever $p_i > 0$. Sometimes we use a hermitian operator M in $\mathcal{L}(\mathcal{H})$ called *observable* to represent a projective measurement. To be specific, let

$$M = \sum_{m \in \text{spec}(M)} m P_m$$

where $\text{spec}(M)$ is the set of eigenvalues of M , and P_m the projector onto the eigenspace associated with m . Then the projective measurement determined by M is $\{P_m : m \in \text{spec}(M)\}$. Note that by the linearity of the trace function, the expected value of outcomes when M is measured on state ρ is calculated as

$$\sum_{m \in \text{spec}(M)} m \cdot \text{tr}(P_m \rho) = \text{tr}(M \rho).$$

Finally, the dynamics that can occur in a (not necessarily closed) physical system are described by a trace-preserving super-operator. Typical examples include the unitary transformation $\mathcal{E}_U(\rho) \triangleq U \rho U^\dagger$ and the state transformation caused by a measurement, when all the post-measurement states are taken into account. More specifically, the evolution

$$\mathcal{E}_{\mathcal{M}}(\rho) \triangleq \sum_{i \in \mathcal{O}} p_i \rho_i = \sum_{i \in \mathcal{O}} P_i \rho P_i$$

is a super-operator.

Notation conventions. To simplify notations, we assume the following conventions throughout the paper.

- For a pure state $|\psi\rangle$ in a finite dimensional Hilbert space \mathcal{H} , we denote by $[|\psi\rangle]$ its corresponding (rank-1) density operator; that is, $[|\psi\rangle] \triangleq |\psi\rangle\langle\psi|$.
- We use subscripts to indicate the quantum systems on which a state, an operator, or a super-operator is acting. For example, $|\psi\rangle_q$ or $[|\psi\rangle_q]$ denotes a pure state $|\psi\rangle$ of qubit q , and CX_{q_1, q_2} denotes the CNOT operator with q_1 being its control qubit and q_2 the target qubit. Here $CX|x\rangle|y\rangle = |x\rangle|x \oplus y\rangle$ for any $x, y \in \{0, 1\}$ and \oplus denotes exclusive-or. Furthermore, for a d -dimensional Hilbert space, denote by $\{|i\rangle : 0 \leq i \leq d-1\}$ its standard (or computational) basis.
- Any super-operator is assumed to be identical to its cylinder extensions (the tensor product of it with the identity super-operator on the remaining subsystems) on larger Hilbert spaces. In other words, \mathcal{E} from $\mathcal{L}(\mathcal{H}_1)$ to $\mathcal{L}(\mathcal{H}_2)$ can be regarded as from $\mathcal{L}(\mathcal{H} \otimes \mathcal{H}_1)$ to $\mathcal{L}(\mathcal{H} \otimes \mathcal{H}_2)$ for any \mathcal{H} by identifying \mathcal{E} and $I_{\mathcal{H}} \otimes \mathcal{E}$. In particular, any value $p \in [0, 1]$ (a super-operator on the 0-dimensional space) can be regarded as $p \cdot I_V$ on $\mathcal{D}(\mathcal{H}_V)$ for any finite set V of qubits.
- Operations on individual elements are assumed to be extended to sets in an element-wise way. For example, let \mathbb{E} and \mathbb{F} be two sets of super-operators. Then

$$\mathbb{E} \circ \mathbb{E} + \mathbb{F} \circ \mathbb{F} \triangleq \{\mathcal{E}' \circ \mathcal{E} + \mathcal{F}' \circ \mathcal{F} : \mathcal{E}' \in \mathbb{E}, \mathcal{F}' \in \mathbb{F}\}$$

where \circ denotes the composition of super-operators; that is, $\mathcal{E} \circ \mathcal{F}(\rho) = \mathcal{E}(\mathcal{F}(\rho))$ for all ρ .

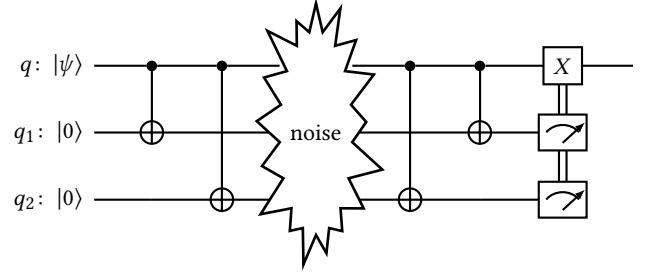


Figure 1: Quantum error correction scheme which corrects a bit-flip error on any of the three qubits.

3 NONDETERMINISTIC QUANTUM PROGRAMS

We extend the purely quantum while-language defined in [6, 28] to describe quantum programs with nondeterministic choices. Let \mathcal{V} , ranged over by q, r, \dots , be a finite set of (qubit-type) quantum variables. For any subset V of \mathcal{V} , let

$$\mathcal{H}_V \triangleq \bigotimes_{q \in V} \mathcal{H}_q,$$

where $\mathcal{H}_q \simeq \mathcal{H}_2$ is the 2-dimensional Hilbert space associated with q . As we use subscripts to distinguish Hilbert spaces associated with different quantum variables, their order in the tensor product is irrelevant.

3.1 Syntax

A nondeterministic quantum program is defined by the following rules:

$$\begin{aligned} S ::= & \text{skip} \mid \text{abort} \mid \bar{q} := 0 \mid \bar{q} * = U \mid S_0; S_1 \mid S_0 \square S_1 \mid \\ & \text{if } M[\bar{q}] \text{ then } S_1 \text{ else } S_0 \text{ end} \mid \text{while } M[\bar{q}] \text{ do } S \text{ end} \end{aligned}$$

where S, S_0 and S_1 are nondeterministic quantum programs, $\bar{q} \triangleq q_1, \dots, q_n$ an (ordered) tuple of distinct qubit-type variables, and U a unitary operator and $M = \{P_0, P_1\}$ a two-outcome projective measurement on $\mathcal{H}_{\bar{q}}$ respectively. Sometimes we also use \bar{q} to denote the (unordered) set $\{q_1, q_2, \dots, q_n\}$. Let $qv(S)$ be the set of quantum variables in S .

The program constructs are standard, and their meaning will be clear when the denotational semantics is given later. Intuitively, **skip** is a no-op statement, **abort** halts the computation with no proper state reached, $\bar{q} := 0$ initialises each qubit in system \bar{q} into $|0\rangle$, $\bar{q} * = U$ applies the unitary operator U on system \bar{q} , $S_0; S_1$ is the sequential composition of S_0 and S_1 , and $S_0 \square S_1$ chooses S_0 or S_1 to execute non-deterministically. The conditional statement **if $M[\bar{q}]$ then S_1 else S_0 end** and the while statement **while $M[\bar{q}]$ do S end** behave similarly to their classical counterparts, except that they use the outcome of measuring M on \bar{q} to determine subsequent operations. These quantum measurements often change the state of the system they are applied on. This is in contrast with classical programs, where such side-effects do not exist.

EXAMPLE 3.1 (QUANTUM ERROR CORRECTION SCHEME AS A NON-DETERMINISTIC PROGRAM). *Error correction codes are widely used to protect quantum information from noise in quantum communication*

channels or during quantum computation. The simplest error correction code, called three-qubit bit-flip code, encodes each qubit state $\alpha_0|0\rangle + \alpha_1|1\rangle$ into a three-qubit state $\alpha_0|000\rangle + \alpha_1|111\rangle$ (see the left part of Fig. 1 for a circuit implementation of the encoding process). All the three qubits then pass through a noisy quantum channel, which flips the qubit (that is, applies a Pauli-X operator on it which turns $|0\rangle$ into $|1\rangle$ and $|1\rangle$ into $|0\rangle$ simultaneously) with some small probability. For simplicity, we assume that at most one of the three qubits is flipped, but we do not know which one it is. Interestingly, by applying a properly designed error correction procedure (shown in the right part of Fig. 1), the error can be detected and corrected perfectly.

If we model the unknown noise with a nondeterministic choice, the quantum error correction scheme, including the processes of encoding, error introduction, and decoding, can be written as a nondeterministic quantum program as follows:

$$\begin{aligned} \text{ErrCorr} &\triangleq \\ & q_1, q_2 := 0; \\ & q, q_1 * = CX; q, q_2 * = CX; \\ & \text{skip} \square q * = X \square q_1 * = X \square q_2 * = X; \\ & q, q_2 * = CX; q, q_1 * = CX; \\ & \text{if } \mathcal{M}[q_2] \text{ then} \\ & \quad \text{if } \mathcal{M}[q_1] \text{ then} \\ & \quad \quad q * = X \\ & \quad \text{end} \\ & \text{end} \end{aligned}$$

Here CX is the CNOT gate, \mathcal{M} is the measurement according to the computational basis $\{|0\rangle, |1\rangle\}$, and the command **if** $\mathcal{M}[r]$ **then** S **end** denotes the abbreviation for **if** $\mathcal{M}[r]$ **then** S **else skip end**. The nondeterministic statement models four different possibilities of error occurring: no error, bit-flip error on the first, second, and third qubit, respectively. Similar to the sequential composition, we assume (and will justify after the formal semantics is given) that the nondeterministic choice is both left- and right-associative.

3.2 Denotational Semantics

This subsection is devoted to a denotational semantics for nondeterministic quantum programs which is obtained by lifting the semantics of deterministic programs presented in [6, 28].

Given a finite dimensional Hilbert space \mathcal{H} , let $\mathcal{S}(\mathcal{H})$ be the set of super-operators on \mathcal{H} . Note that $\mathcal{S}(\mathcal{H})$ is a complete partial order (CPO) with respect to the order \leq defined as follows: $\mathcal{E} \leq \mathcal{F}$ iff there exists a super-operator \mathcal{G} such that $\mathcal{F} = \mathcal{E} + \mathcal{G}$. The following lemma shows that this partial order coincides with the one defined by lifting pointwise the Löwner order over density operators on extended Hilbert spaces.

LEMMA 3.1. *For any $\mathcal{E}, \mathcal{F} \in \mathcal{S}(\mathcal{H})$, $\mathcal{E} \leq \mathcal{F}$ iff for any auxiliary Hilbert space \mathcal{H}' and any $\rho \in \mathcal{D}(\mathcal{H} \otimes \mathcal{H}')$,*

$$\mathcal{E}(\rho) \sqsubseteq \mathcal{F}(\rho).$$

PROOF. The necessity part is easy. For the sufficiency part, note that the mapping $\mathcal{G} \triangleq \mathcal{F} - \mathcal{E}$ is linear, trace non-increasing, and completely positive. Thus it is also a super-operator. \square

$$\begin{aligned} \llbracket \text{skip} \rrbracket &= \{1\} \\ \llbracket \text{abort} \rrbracket &= \{0\} \\ \llbracket \bar{q} := 0 \rrbracket &= \{\text{Set}_{\bar{q}}^0\} \\ \llbracket \bar{q} * = U \rrbracket &= \{\mathcal{U}_{\bar{q}}\} \\ \llbracket S_0; S_1 \rrbracket &= \llbracket S_1 \rrbracket \circ \llbracket S_0 \rrbracket \\ \llbracket S_0 \square S_1 \rrbracket &= \llbracket S_0 \rrbracket \cup \llbracket S_1 \rrbracket \end{aligned}$$

$$\llbracket \text{if } \mathcal{M}[\bar{q}] \text{ then } S_1 \text{ else } S_0 \text{ end} \rrbracket = \llbracket S_0 \rrbracket \circ \mathcal{P}_{\bar{q}}^0 + \llbracket S_1 \rrbracket \circ \mathcal{P}_{\bar{q}}^1$$

$$\llbracket \text{while } \mathcal{M}[\bar{q}] \text{ do } S \text{ end} \rrbracket =$$

$$\left\{ \sum_{i=0}^{\infty} \mathcal{P}_{\bar{q}}^0 \circ \eta_i \circ \mathcal{P}_{\bar{q}}^1 \circ \dots \circ \eta_1 \circ \mathcal{P}_{\bar{q}}^1 : \eta \in \llbracket S \rrbracket^{\mathbb{N}} \right\}$$

Figure 2: Denotational semantics for nondeterministic quantum programs.

Let Prog be the set of all nondeterministic quantum programs. For any $S \in \text{Prog}$, the denotational semantics $\llbracket S \rrbracket$ of S is a set of super-operators in $\mathcal{S}(\mathcal{H}_{qv(S)})$ defined inductively in Fig. 2, where $\mathcal{M} = \{P^0, P^1\}$, $\mathcal{P}_{\bar{q}}^i$, $i = 0, 1$, is a super-operator such that $\mathcal{P}_{\bar{q}}^i(\rho) = P_{\bar{q}}^i \rho P_{\bar{q}}^i$, and $\text{Set}_{\bar{q}}^0$ and $\mathcal{U}_{\bar{q}}$ with $|\bar{q}| = n$ are super-operators such that $\text{Set}_{\bar{q}}^0(\rho) = \sum_{i=0}^{2^n-1} |0\rangle_{\bar{q}} \langle i| \rho |i\rangle_{\bar{q}} \langle 0|$ and $\mathcal{U}_{\bar{q}}(\rho) = U_{\bar{q}} \rho U_{\bar{q}}^\dagger$ for all $\rho \in \mathcal{D}(\mathcal{H}_{\mathcal{V}})$. Recall that the semantics of a deterministic quantum program is a single super-operator. We borrow the idea widely used in classical programming theory to employ sets of super-operators to represent nondeterminism in the semantics of nondeterministic quantum programs.

A more elegant way to define the denotational semantics of nondeterministic quantum programs is to consider the power domain whose elements are subsets of super-operators with certain closure properties. The semantics of a while loop, say, is then simply the least fixed point of some Scott-continuous function over this power domain. However, since the main goal of this paper is to develop verification techniques for nondeterministic quantum programs, we decided to adopt the current definition, which is more intuitive and accessible for ordinary readers without a background in domain theory.

Note that with our notational convention, $\llbracket S \rrbracket$ can also be regarded as a subset of $\mathcal{S}(\mathcal{H}_{\mathcal{V}})$ for any $V \supseteq qv(S)$. For example, the number 1 in $\llbracket \text{skip} \rrbracket$ can be regarded as the identity super-operator $I_{\mathcal{H}_{\mathcal{V}}}$ for any $V \subseteq \mathcal{V}$. Similarly, the number 0 in $\llbracket \text{abort} \rrbracket$ can represent the zero super-operator on any qubit system. Furthermore, note that $0(\rho)$ is the zero density operator for any input ρ . This accurately captures the intuition that **abort** applied to any input cannot produce any valid quantum state.

As the four basic commands **skip**, **abort**, $\bar{q} := 0$, $\bar{q} * = U$ are all deterministic, their semantic sets contain only a single super-operator representing the corresponding quantum operation. Recall also that we use the subscript \bar{q} to denote the qubit system on which the quantum operation is applied. The semantics of $S_0; S_1$ is defined as the (element-wise) composition of $\llbracket S_1 \rrbracket$ and $\llbracket S_0 \rrbracket$. This follows the lifted model proposed in [8] for nondeterministic

probabilistic programs. The reason why we adopt this lifted model (rather than the relational one) will be elaborated in Sec. 3.3.

For $[[\text{if } M[\bar{q}] \text{ then } S_1 \text{ else } S_0 \text{ end}]]$, we have to combine the semantics of S_0 and S_1 using the corresponding measurement super-operators. Specifically, for any input state ρ , if the measurement returns 0, then the post-measurement state becomes $\mathcal{P}_{\bar{q}}^0(\rho)$, and one of the super-operators in $[[S_0]]$ will be applied on this state. The case where the measurement result is 1 is similar. Finally, the (partial) density operators obtained from the two branches must be added together to form the output state of the entire statement, since the two branches are chosen probabilistically (due to the measurement), rather than non-deterministically.

Finally, in $[[\text{while } M[\bar{q}] \text{ do } S \text{ end}]]$, since the loop body S may contain nondeterministic choices, we use a *scheduler* η to specify which super-operator in $[[S]]$ is taken in each iteration of the while loop. To simplify the notation, we write η_i for $\eta(i)$, the super-operator taken in the i -th iteration. For each $n \geq 0$, the super-operator

$$\mathcal{F}_n^\eta \triangleq \sum_{i=0}^n \mathcal{P}_{\bar{q}}^0 \circ \eta_i \circ \mathcal{P}_{\bar{q}}^1 \circ \dots \circ \eta_1 \circ \mathcal{P}_{\bar{q}}^1 \quad (1)$$

is obtained from the first n executions of the loop body S under the scheduler η . It is evident that the sequence \mathcal{F}_n^η , $n \geq 0$, is non-decreasing under \leq . Thus the least upper bound $\bigvee_{n \geq 0} \mathcal{F}_n^\eta$ is well-defined, and it is the semantics of $\text{while } M[\bar{q}] \text{ do } S \text{ end}$ when η is used to resolve the nondeterminism in S .

EXAMPLE 3.2. We revisit the quantum error correction scheme presented in Example 3.1. First, it is easy to see that

$$\begin{aligned} [[\text{ErrCorr}]] = & \left\{ \left(X_q \circ \mathcal{P}_{q_1, q_2}^{11} + \mathcal{P}_{q_1, q_2}^{\neq 11} \right) \circ CX_{q, q_1} \circ CX_{q, q_2} \circ \mathcal{U} \right. \\ & \left. \circ CX_{q, q_2} \circ CX_{q, q_1} \circ \text{Set}_{q_1, q_2}^0 : \mathcal{U} \in \{I, X_q, X_{q_1}, X_{q_2}\} \right\} \end{aligned}$$

where X and CX are the super-operators corresponding to the unitary operators X and CX , respectively. Furthermore,

$$\mathcal{P}^{11}(\rho) = |11\rangle\langle 11| \rho |11\rangle\langle 11|$$

and

$$\mathcal{P}^{\neq 11}(\rho) = \sum_{(i,j) \neq (1,1)} |ij\rangle\langle ij| \rho |ij\rangle\langle ij|$$

for all $\rho \in \mathcal{D}(\mathcal{H}_2 \otimes \mathcal{H}_2)$. That is, $\mathcal{P}_{q_1, q_2}^{11}$ denotes the super-operator corresponding to the case where the measurements $M[q_2]$ and $M[q_1]$ both obtain 1, while $\mathcal{P}_{q_1, q_2}^{\neq 11}$ represents the other cases.

From any input state $|\psi\rangle_q |\psi'\rangle_{q_1, q_2}$ where $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ and $|\psi'\rangle \in \mathcal{H}_2 \otimes \mathcal{H}_2$, although there are four different super-operators in the denotational semantics $[[\text{ErrCorr}]]$, when applied on $|\psi\rangle_q |\psi'\rangle_{q_1, q_2}$, the final quantum states $|\psi\rangle_q |ij\rangle_{q_1, q_2}$, $i, j = 0, 1$, have the same reduced state $|\psi\rangle$ on qubit q , which is exactly the input state on q . From this observation, we conclude that the error correction scheme has successfully corrected the potential bit-flip error occurred in any of the three qubits.

The next lemma gives a recursive description of the denotational semantics of while loops. For any scheduler $\eta \in [[S]]^{\mathbb{N}}$, we define $\eta^{\rightarrow} \in [[S]]^{\mathbb{N}}$ to be the suffix of η starting from η_2 ; that is, $\eta_i^{\rightarrow} = \eta_{i+1}$ for all $i \geq 1$.

LEMMA 3.2. Let $\text{while} \triangleq \text{while } M[\bar{q}] \text{ do } S \text{ end}$. Then for any $\eta \in [[S]]^{\mathbb{N}}$ and $n \geq 0$,

$$\mathcal{F}_{n+1}^\eta = \mathcal{P}_{\bar{q}}^0 + \mathcal{F}_n^{\eta^{\rightarrow}} \circ \eta_1 \circ \mathcal{P}_{\bar{q}}^1. \quad (2)$$

where \mathcal{F}_n^η is defined as in Eq. (1). Consequently,

$$[[\text{while}]] = \mathcal{P}_{\bar{q}}^0 + [[\text{while}]] \circ [[S]] \circ \mathcal{P}_{\bar{q}}^1. \quad (3)$$

PROOF. Eq. (2) can be easily proved by induction on n , and then the ' \leq ' part of Eq. (3) follows. For the ' \supseteq ' part, let $\mathcal{G} \in \mathcal{P}_{\bar{q}}^0 + [[\text{while}]] \circ [[S]] \circ \mathcal{P}_{\bar{q}}^1$. Then there exists $\eta \in [[S]]^{\mathbb{N}}$ and $\mathcal{E} \in [[S]]$ such that $\mathcal{G} = \mathcal{P}_{\bar{q}}^0 + \bigvee_{n \geq 0} \mathcal{F}_n^\eta \circ \mathcal{E} \circ \mathcal{P}_{\bar{q}}^1$. Let $\eta' \in [[S]]^{\mathbb{N}}$ with $\eta'_1 = \mathcal{E}$ and $\eta'_{i+1} = \eta_i$ for all $i \geq 1$. Then it is easy to prove by induction that for any $n \geq 1$,

$$\mathcal{F}_n^{\eta'} = \mathcal{P}_{\bar{q}}^0 + \mathcal{F}_n^\eta \circ \mathcal{E} \circ \mathcal{P}_{\bar{q}}^1.$$

Thus $\mathcal{G} = \bigvee_{n \geq 0} \mathcal{F}_n^{\eta'}$ is in $[[\text{while}]]$ as well. \square

3.3 Different Approaches for Semantics of Quantum Programs

This subsection is devoted to an explanation of the design decisions we made in defining the semantics of nondeterministic quantum programs.

3.3.1 *Pure-State v.s. Mixed-State Semantics.* Note that there are two different approaches in defining semantics of deterministic quantum programs in the literature. One is based on pure states [2, 21, 22, 24, 26], in which the meaning of a program is defined assuming that it is applied on pure states. The semantics is then extended to mixed states using, say, spectral decomposition. To be specific, if $\rho = \sum_{i \in I} p_i |\psi_i\rangle\langle \psi_i|$ and a program S maps $|\psi_i\rangle$ to $|\psi'_i\rangle$, then the final (mixed) state of executing S on ρ is $\sum_{i \in I} p_i |\psi'_i\rangle\langle \psi'_i|$. In contrast, the other approach is based on mixed states [4, 6, 23, 28]. That is, the semantics of a program is defined directly on mixed states without extension.

The choice of pure-state v.s. mixed-state semantics does not make any difference when the programs considered are deterministic. The reason is that the semantics of a deterministic quantum program is a super-operator which is linear with respect to convex combination of density operators. Consequently, the two approaches indeed obtain the same semantics for deterministic quantum programs when applied on mixed states. However, this is not true for nondeterministic quantum programs, since now a program often corresponds to a set of super-operators, instead of a single one. As a result, convex combination of pure-state semantics does not give the same result as the mixed-state semantics. To make this point clear, let us see a simple example.

EXAMPLE 3.3. Let S be a nondeterministic quantum program defined as follows:

$$S \triangleq \text{skip} \square q * = X$$

Then we have from Fig. 2 that $[[S]] = \{1, X_q\}$. Consequently,

$$\begin{aligned} [[S]](|0\rangle)_q &= \{|0\rangle_q, |1\rangle_q\} \\ [[S]](|1\rangle)_q &= \{|0\rangle_q, |1\rangle_q\} \\ [[S]](|+\rangle)_q &= \{|+\rangle_q\} \end{aligned} \quad (4)$$

$$\llbracket S \rrbracket(\llbracket |-\rangle \rrbracket_q) = \{ \llbracket |-\rangle \rrbracket_q \}$$

where $|\pm\rangle \triangleq \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$, and the last equation follows from the fact that $[X|-\rangle] = [-|-\rangle] = [|-\rangle]$; that is, the global phase disappears in the density operator representation of quantum states.

Suppose we had adopted the pure-state approach to define our denotational semantics, and extended it to mixed states by taking the convex combination of pure-state semantics. Recall that in \mathcal{H}_2 ,

$$\frac{I}{2} = \frac{1}{2} (\llbracket |0\rangle \rrbracket + \llbracket |1\rangle \rrbracket) = \frac{1}{2} (\llbracket |+\rangle \rrbracket + \llbracket |-\rangle \rrbracket). \quad (5)$$

Then from the first two equations in Eq. (4) we would derive

$$\llbracket S \rrbracket(I_q/2) = \{ \llbracket |0\rangle \rrbracket_q, \llbracket |1\rangle \rrbracket_q, I_q/2 \},$$

while from the last two equations in Eq. (4) we would have

$$\llbracket S \rrbracket(I_q/2) = \{ I_q/2 \}$$

instead. That is, the extended semantics for mixed states would not be well-defined.

Note that this inconsistency does not exist for classical programs because, unlike the fact that a density operator can represent different ensembles of pure states, exemplified in Eq. (5) for $I/2$, any probability distribution of classical states has a unique representation (as a convex combination of underlying states).

3.3.2 Relational v.s. Lifted Model. As pointed out in [8], nondeterministic probabilistic programs can naturally be given two different semantic models: a relational one and a lifted one. Intuitively, in the lifted model, a nondeterministic program is semantically regarded as a set of deterministic programs, while in the relational model, a dedicated semantics is given by taking into account the interference between probability and nondeterminism. The difference between these two models can be best illustrated in their treatment of sequential composition of programs. Specifically, let Σ be the classical state space, ranged over by s, t , etc. For simplicity, assume that Σ is countable. Let S and T be nondeterministic probabilistic programs. Recall that the semantics $\llbracket S \rrbracket^{\mathbf{r}}$ of S in the relational model is a mapping from states in Σ to sets of probability distributions over Σ . The semantics of the sequential composition $S; T$ is defined for any $s \in \Sigma$ as

$$\llbracket S; T \rrbracket^{\mathbf{r}}(s) = \left\{ \sum_{t \in \Sigma} \mu(t) \cdot \nu_t : \mu \in \llbracket S \rrbracket^{\mathbf{r}}(s), \forall t. \nu_t \in \llbracket T \rrbracket^{\mathbf{r}}(t) \right\}. \quad (6)$$

That is, each probability distribution in $\llbracket S; T \rrbracket^{\mathbf{r}}(s)$ is generated by first choosing a distribution μ in $\llbracket S \rrbracket^{\mathbf{r}}(s)$, then for each state t choosing a distribution ν_t in $\llbracket T \rrbracket^{\mathbf{r}}(t)$, and finally taking the convex combination of ν_t 's where the weight of ν_t is given by $\mu(t)$. In contrast, the semantics $\llbracket S \rrbracket^{\mathbf{l}}$ of S in the lifted model is a set of deterministic distribution-transformers which map states in Σ to probability distributions over Σ . The lifted semantics of $S; T$ is defined to be

$$\llbracket S; T \rrbracket^{\mathbf{l}} = \left\{ g \circ f : f \in \llbracket S \rrbracket^{\mathbf{l}}, g \in \llbracket T \rrbracket^{\mathbf{l}} \right\} \quad (7)$$

where $g \circ f$ is defined in the standard way: $(g \circ f)(s) = \sum_{t \in \Sigma} f(s)(t) \cdot g(t)$. Although these two models are both well-motivated, it was argued in [8] that the relational model is preferable since it enjoys more nice algebraic properties.

Obviously, our definition in Fig. 2 follows the same style of the lifted model in Eq. (7). To illustrate why it is problematic to have a relational semantics for nondeterministic quantum programs, let us take a close look at the definition of $\llbracket S; T \rrbracket^{\mathbf{r}}$ in Eq. (6) from the perspective of a game. Intuitively, for each $t \in \Sigma$, ν_t is chosen from $\llbracket T \rrbracket^{\mathbf{r}}(t)$ by the adversary if the current program state is t . Note that this only makes sense under the assumption that the program state at any given moment can be determined exactly during the runtime so that the adversary can use this information to make the choice. However, this assumption does not necessarily hold in the quantum setting, also due to the fact that a density operator can represent different ensembles of pure states. To make this point more clear, let us consider the following example.

EXAMPLE 3.4. Let S be defined as in Example 3.3 and

$$\begin{aligned} T &\triangleq q := 0; q * H; \text{measure } q \\ T_{\pm} &\triangleq q := 0; \text{measure}_{\pm} q \end{aligned}$$

Here the command **measure** q is the abbreviation for **if** $M_{0,1}[q]$ **then skip else skip end** and $M_{0,1}$ is the measurement according to the computational basis $\{|0\rangle, |1\rangle\}$. Similarly, **measure** $_{\pm}$ q is the abbreviation for **if** $M_{\pm}[q]$ **then skip else skip end** and M_{\pm} is the measurement according to the orthonormal basis $\{|+\rangle, |-\rangle\}$. Note that both T and T_{\pm} are deterministic. Obviously, for any input state ρ in $\mathcal{D}(\mathcal{H}_q)$ with $\text{tr}(\rho) = 1$, the output state of T is the ensemble $\left(|0\rangle : \frac{1}{2}, |1\rangle : \frac{1}{2} \right)$, which can also be regarded as a probability distribution taking $|0\rangle$ or $|1\rangle$ uniformly. Similarly, the output state of T_{\pm} for the same input ρ is the ensemble $\left(|+\rangle : \frac{1}{2}, |-\rangle : \frac{1}{2} \right)$, a probability distribution taking $|+\rangle$ or $|-\rangle$ uniformly. If we adopt the relational semantics $\llbracket \cdot \rrbracket^{\mathbf{r}}$ by regarding mixed quantum states as probability distributions over pure states, then we may have

$$\begin{aligned} \llbracket T \rrbracket^{\mathbf{r}}(\rho) &= \left\{ \left(|0\rangle : \frac{1}{2}, |1\rangle : \frac{1}{2} \right) \right\}, \\ \llbracket T_{\pm} \rrbracket^{\mathbf{r}}(\rho) &= \left\{ \left(|+\rangle : \frac{1}{2}, |-\rangle : \frac{1}{2} \right) \right\}. \end{aligned}$$

Note that these two ensembles, although different in the probability distribution form, are physically indistinguishable; see Eq. (5). Consequently, $\llbracket T \rrbracket^{\mathbf{r}} = \llbracket T_{\pm} \rrbracket^{\mathbf{r}}$.

Now, consider the sequential composition of T and T_{\pm} with S respectively. Similar to Eq. (6), we may have

$$\begin{aligned} \llbracket T; S \rrbracket^{\mathbf{r}}(\rho) &= \frac{1}{2} \cdot \llbracket S \rrbracket^{\mathbf{r}}(\llbracket |0\rangle \rrbracket) + \frac{1}{2} \cdot \llbracket S \rrbracket^{\mathbf{r}}(\llbracket |1\rangle \rrbracket) \\ &= \frac{1}{2} \cdot \{ |0\rangle, |1\rangle \} + \frac{1}{2} \cdot \{ |0\rangle, |1\rangle \} \\ &= \left\{ (|0\rangle : 1), (|1\rangle : 1), \left(|0\rangle : \frac{1}{2}, |1\rangle : \frac{1}{2} \right) \right\} \end{aligned}$$

while

$$\begin{aligned} \llbracket T_{\pm}; S \rrbracket^{\mathbf{r}}(\rho) &= \frac{1}{2} \cdot \llbracket S \rrbracket^{\mathbf{r}}(\llbracket |+\rangle \rrbracket) + \frac{1}{2} \cdot \llbracket S \rrbracket^{\mathbf{r}}(\llbracket |-\rangle \rrbracket) \\ &= \frac{1}{2} \cdot \{ |+\rangle \} + \frac{1}{2} \cdot \{ |-\rangle \} = \left\{ |+\rangle : \frac{1}{2}, |-\rangle : \frac{1}{2} \right\}. \end{aligned}$$

In the language of density operators, we can say that starting from ρ , $\llbracket T; S \rrbracket^{\mathbf{r}}$ may output $|0\rangle$, $|1\rangle$, or $\frac{1}{2} \left(\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| \right)$ non-deterministically while $\llbracket T_{\pm}; S \rrbracket^{\mathbf{r}}$ can only output $\frac{1}{2}$

$(= \frac{1}{2}|+\rangle\langle+| + \frac{1}{2}|-\rangle\langle-|)$ although the nondeterministic choice also exists in it. In other words, $[[T;S]]^F \neq [[T_{\pm};S]]^F$ although $[[T]]^F = [[T_{\pm}]]^F$. This is highly undesirable because semantic composability is a natural requirement of language design.

4 VERIFICATION OF NONDETERMINISTIC QUANTUM PROGRAMS

Recall that a common practice in the verification of (deterministic or nondeterministic) probabilistic programs is to take expectations, that is, linear functions $f : \Sigma \rightarrow [0, 1]$ where Σ is the set of classical states, as assertions. Then a program can be regarded as an expectation-transformer which maps a post-expectation to its greatest pre-expectation. The reason why deterministic and nondeterministic probabilistic programs can share the same form of assertions is that the set of expectations constitutes a complete lattice with respect to the pointwise partial order, where the join and meet are also defined pointwisely. Consequently, it is closed under taking infimum, the operation corresponding to the (demonic) nondeterminism.

To describe desirable properties of a quantum state, we follow the approach of regarding hermitian operators from

$$\mathcal{P}(\mathcal{H}_{\mathcal{V}}) \triangleq \{M \in \mathcal{L}(\mathcal{H}_{\mathcal{V}}) : 0 \sqsubseteq M \sqsubseteq I\}.$$

as quantum predicates in the verification of deterministic quantum programs [4, 6, 28]. Note that any hermitian operator $M \in \mathcal{P}(\mathcal{H}_{\mathcal{V}})$ induces a linear function $f_M : \mathcal{D}(\mathcal{H}_{\mathcal{V}}) \rightarrow [0, 1]$ by setting $f_M(\rho) = \text{tr}(M\rho)$ for any ρ . Remarkably, recall from Sec. 2 that $\text{tr}(M\rho)$ is exactly the expected value of outcomes when measuring the observable M on state ρ . It is naturally interpreted as the degree of ρ satisfying M if M represents some desired property of the quantum system.

However, the set $\mathcal{P}(\mathcal{H}_{\mathcal{V}})$, although being a CPO, does not form a lattice. Thus it is not expressive enough for the verification of nondeterministic quantum programs. In this paper, we simply take $\mathcal{A} \triangleq 2^{\mathcal{P}(\mathcal{H}_{\mathcal{V}})}$, the collection of subsets of $\mathcal{P}(\mathcal{H}_{\mathcal{V}})$, ranged over by Θ, Ψ, \dots , as our set of quantum assertions. It is obviously a complete lattice in the usual subset order. When $\Theta = \{M\}$ is a singleton we simply write Θ as M . We further extend the operations applied on hermitian operators to quantum assertions in an element-wise way. For example, let \mathcal{E} be a super-operator. Then by $\mathcal{E}(\Theta)$ we denote the set $\{\mathcal{E}(M) : M \in \Theta\}$.

DEFINITION 4.1. Given a density operator $\rho \in \mathcal{D}(\mathcal{H}_{\mathcal{V}})$ and a quantum assertion $\Theta \in \mathcal{A}$, the expectation of ρ satisfying Θ is defined to be $\text{Exp}(\rho \models \Theta) \triangleq \inf_{M \in \Theta} \text{tr}(M\rho)$.

The infimum taken in the above definition of $\text{Exp}(\rho \models \Theta)$ reflects a pessimistic view of the satisfaction: it provides a *guaranteed* expected satisfaction of Θ by ρ in the presence of possibly demonic nondeterminism.

4.1 Correctness Formula

As usual, program correctness is expressed by *correctness formulas* with the form $\{\Theta\} S \{\Psi\}$ where S is a quantum program, and Θ and Ψ are quantum assertions in \mathcal{A} .

DEFINITION 4.2. Let $S \in \text{Prog}$, and $\Theta, \Psi \in \mathcal{A}$.

(1) We say the correctness formula $\{\Theta\} S \{\Psi\}$ is true in the sense of total correctness, written $\models_{\text{tot}} \{\Theta\} S \{\Psi\}$, if for any $\rho \in \mathcal{D}(\mathcal{H}_{\mathcal{V}})$,

$$\text{Exp}(\rho \models \Theta) \leq \inf \{ \text{Exp}(\sigma \models \Psi) : \sigma \in [[S]](\rho) \}.$$

(2) We say the correctness formula $\{\Theta\} S \{\Psi\}$ is true in the sense of partial correctness, written $\models_{\text{par}} \{\Theta\} S \{\Psi\}$, if for any $\rho \in \mathcal{D}(\mathcal{H}_{\mathcal{V}})$,

$$\text{Exp}(\rho \models \Theta) \leq \inf \{ \text{Exp}(\sigma \models \Psi) + \text{tr}(\rho) - \text{tr}(\sigma) : \sigma \in [[S]](\rho) \}.$$

Intuitively, $\models_{\text{tot}} \{\Theta\} S \{\Psi\}$ iff starting from any initial state, the guaranteed expected satisfaction of postcondition Ψ by any possible final state is lower bounded by the guaranteed expected satisfaction of precondition Θ by the initial state. For the case of partial correctness, we relax the lower bound by taking the non-termination probability $\text{tr}(\rho) - \text{tr}(\sigma)$ into account. It is easy to see that when S is deterministic, and both Θ and Ψ contain only one quantum predicate, the above definition reduces to the corresponding one in [28] for deterministic quantum programs.

EXAMPLE 4.1. The correctness of quantum error correction scheme for bit flip in Example 3.1 can be stated as follows: for any $|\psi\rangle \in \mathcal{H}_q$,

$$\models_{\text{tot}} \{ |\psi\rangle_q \langle \psi| \} \text{ErrCorr} \{ |\psi\rangle_q \langle \psi| \}. \quad (8)$$

To see why Eq.(8) captures the intuition that the (arbitrary) state of qubit q has been successfully protected from the possible bit-flip error, note that Eq.(8) implies

$$1 = \langle \psi | \psi \rangle \langle \psi | \psi \rangle \leq \inf \{ \langle \psi | \sigma | \psi \rangle : \sigma \in [[\text{ErrCorr}]](|\psi\rangle\langle\psi|) \}.$$

Thus $\sigma = |\psi\rangle\langle\psi|$ for all $\sigma \in [[\text{ErrCorr}]](|\psi\rangle\langle\psi|)$.

Note that for $* \in \{tot, par\}$, if there exists $M \in \Theta$ such that $\models_* \{M\} S \{N\}$ for all $N \in \Psi$, then $\models_* \{\Theta\} S \{\Psi\}$. However, the reverse is not true. A counterexample is as follows. Let $\mathcal{V} \triangleq \{q\}$, $\Theta \triangleq \{|0\rangle_q \langle 0|, |1\rangle_q \langle 1|\}$, $\Psi \triangleq \{I_q/2\}$, and $S \triangleq \text{skip}$. Note that for any $\rho \in \mathcal{D}(\mathcal{H}_{\mathcal{V}})$, $\text{tr}(|0\rangle_q \langle 0| \rho) + \text{tr}(|1\rangle_q \langle 1| \rho) = \text{tr}(\rho)$. Thus

$$\begin{aligned} \text{Exp}(\rho \models \Theta) &= \min \{ \text{tr}(|0\rangle_q \langle 0| \rho), \text{tr}(|1\rangle_q \langle 1| \rho) \} \\ &\leq \frac{\text{tr}(\rho)}{2} = \text{Exp}(\rho \models \Psi), \end{aligned}$$

and so $\models_* \{\Theta\} S \{\Psi\}$. However, neither $\models_* \{|0\rangle_q \langle 0|\} S \{I_q/2\}$ nor $\models_* \{|1\rangle_q \langle 1|\} S \{I_q/2\}$ holds. To see the former one, we note that

$$\text{Exp}(|0\rangle_q \langle 0| \models |0\rangle_q \langle 0|) = 1 \not\leq 1/2 = \text{Exp}(|0\rangle_q \langle 0| \models I_q/2).$$

For the latter one, we can take $|1\rangle_q \langle 1|$ as the initial state.

The following are some basic facts about total and partial correctness.

LEMMA 4.1. Let $S \in \text{Prog}$, Θ and Ψ be quantum assertions.

- (1) If $\models_{\text{tot}} \{\Theta\} S \{\Psi\}$ then $\models_{\text{par}} \{\Theta\} S \{\Psi\}$;
- (2) $\models_{\text{tot}} \{0\} S \{\Psi\}$ and $\models_{\text{par}} \{\Theta\} S \{I\}$;
- (3) If $\models_* \{\Theta_i\} S \{\Psi_i\}$ for all i , then $\models_* \{\cup_i \Theta_i\} S \{\cup_i \Psi_i\}$, where $*$ $\in \{tot, par\}$.

PROOF. Clause (1) follows directly from the fact that $\text{tr}(\sigma) \leq \text{tr}(\rho)$ for all $\sigma \in [[S]](\rho)$, clause (3) from the definition, and clause (2) from the observation that for any $M \in \Theta$, $\text{tr}(M\rho) \leq \text{tr}(\rho)$. \square

(Skip)	$\{\Theta\} \text{ skip } \{\Theta\}$
(Abort)	$\{I\} \text{ abort } \{0\}$
(Init)	$\left\{ \sum_{i=0}^{2^n-1} i\rangle_{\bar{q}} \langle 0 \Theta 0\rangle_{\bar{q}} \langle i \right\} \bar{q} := 0 \{\Theta\}$
(Unit)	$\left\{ U_{\bar{q}}^\dagger \Theta U_{\bar{q}} \right\} \bar{q} * = U \{\Theta\}$
(Seq)	$\frac{\{\Theta\} S_0 \{\Theta'\}, \{\Theta'\} S_1 \{\Psi\}}{\{\Theta\} S_0; S_1 \{\Psi\}}$
(NDet)	$\frac{\{\Theta\} S_0 \{\Psi\}, \{\Theta\} S_1 \{\Psi\}}{\{\Theta\} S_0 \sqcap S_1 \{\Psi\}}$
(Meas)	$\frac{\{\Theta_1\} S_1 \{\Psi\}, \{\Theta_0\} S_0 \{\Psi\}}{\left\{ \mathcal{P}_{\bar{q}}^0(\Theta_0) + \mathcal{P}_{\bar{q}}^1(\Theta_1) \right\} \text{ if } \mathcal{M}[\bar{q}] \text{ then } S_1 \text{ else } S_0 \text{ end } \{\Psi\}}$
(While)	$\frac{\{\Theta\} S \left\{ \mathcal{P}_{\bar{q}}^0(\Psi) + \mathcal{P}_{\bar{q}}^1(\Theta) \right\}}{\left\{ \mathcal{P}_{\bar{q}}^0(\Psi) + \mathcal{P}_{\bar{q}}^1(\Theta) \right\} \text{ while } \mathcal{M}[\bar{q}] \text{ do } S \text{ end } \{\Psi\}}$
(Imp)	$\frac{\Theta \sqsubseteq_{\text{inf}} \Theta', \{\Theta'\} S \{\Psi'\}, \Psi' \sqsubseteq_{\text{inf}} \Psi}{\{\Theta\} S \{\Psi\}}$
(Union)	$\frac{\{\Theta_i\} S \{\Psi_i\} \text{ for all } i \in I}{\{\bigcup_{i \in I} \Theta_i\} S \{\bigcup_{i \in I} \Psi_i\}}$

Figure 3: Proof system for partial correctness.

To conclude this subsection, we define a pre-order between quantum assertions by letting $\Theta \sqsubseteq_{\text{inf}} \Psi$ if for any ρ , $\inf_{M \in \Theta} \text{tr}(M\rho) \leq \inf_{N \in \Psi} \text{tr}(N\rho)$. The following lemma is useful in our later discussion.

LEMMA 4.2. (1) *Let \mathcal{E} be a super-operator and $\Theta \sqsubseteq_{\text{inf}} \Psi$. Then $\mathcal{E}^\dagger(\Theta) \sqsubseteq_{\text{inf}} \mathcal{E}^\dagger(\Psi)$. Recall that \mathcal{E}^\dagger is the adjoint super-operator of \mathcal{E} .*

(2) *If for all i , $\Theta_i \sqsubseteq_{\text{inf}} \Psi_i$, then $\bigcup_i \Theta_i \sqsubseteq_{\text{inf}} \bigcup_i \Psi_i$.*

PROOF. Easy from the definition of \sqsubseteq_{inf} . □

4.2 Proof Systems

The core of Hoare logic is a proof system consisting of axioms and proof rules which enable syntax-oriented and modular reasoning of program correctness. In this section, we propose a Hoare logic for nondeterministic quantum programs.

Partial Correctness. We propose in Fig. 3 a proof system for partial correctness of quantum programs, which is a natural extension of the Hoare logic system introduced in [28] for deterministic quantum programs.

To help understand the rules presented in Fig. 3, let us compare them with their counterparts for classical programs. The rules (Skip), (Abort), (Seq), (NDet), (Imp) have the same form as the corresponding classical ones. Note that for any pure state $|\psi\rangle$, $\text{tr}(I \cdot$

$|\psi\rangle\langle\psi|) = 1$ and $\text{tr}(0 \cdot |\psi\rangle\langle\psi|) = 0$. Thus, the quantum predicates I and 0 play similar roles as **true** and **false** do respectively in classical assertions. The rules (Init) and (Unit) are the counterparts of the classical assignment rule, and they can be better understood in a backwards fashion. For example, (Unit) means that to guarantee postcondition Ψ , it suffices to have $U_{\bar{q}}^\dagger \Theta U_{\bar{q}}$ as the precondition.

Recall the proof rule for classical conditional statements

$$\frac{\{p \wedge B\} S_1 \{q\}, \{p \wedge \neg B\} S_0 \{q\}}{\{p\} \text{ if } B \text{ then } S_1 \text{ else } S_0 \text{ end } \{q\}}. \quad (9)$$

Due to the lack of conjunction of quantum assertions, our proof rule (Meas) takes an alternative form, adding the preconditions of the two branches of **if** $\mathcal{M}[\bar{q}]$ **then** S_1 **else** S_0 **end** after applying the corresponding measurement super-operators. Note that $p = (p \wedge B) \vee (p \wedge \neg B)$. So (Meas) is indeed a generalisation of the rule in Eq. (9). Inspired by [28], the quantum assertion $\mathcal{P}_{\bar{q}}^0(\Psi) + \mathcal{P}_{\bar{q}}^1(\Theta)$ in rule (While) serves as a *loop invariant* of the while loop. Finally, we introduce rule (Union) to combine different correctness formulas for the same quantum program, mimicking the conjunction rule for classical programs.

Denote by $\vdash_{\text{par}} \{\Theta\} S \{\Psi\}$ if the correctness formula $\{\Theta\} S \{\Psi\}$ can be derived from the proof system.

THEOREM 4.1. *The proof system in Fig. 3 is both sound and relatively complete with respect to the partial correctness of nondeterministic quantum programs.*

PROOF. (Sketch) Similar to [28], the key idea for the proof here is to define the notion of weakest liberal precondition for quantum programs. Specially, for any nondeterministic quantum program S and postcondition $\Psi \in \mathcal{A}$, we construct a quantum assertion, denoted $wlp.S.\Psi$, which is the weakest (or largest in terms of \sqsubseteq_{inf}) one among all valid preconditions. In other words, for any $\Theta \in \mathcal{A}$, $\vdash_{\text{par}} \{\Theta\} S \{\Psi\}$ iff $\Theta \sqsubseteq_{\text{inf}} wlp.S.\Psi$.

Now, to prove the soundness of our logic system, it suffices to show by induction on the structure of S that whenever $\vdash_{\text{par}} \{\Theta\} S \{\Psi\}$, it holds $\Theta \sqsubseteq_{\text{inf}} wlp.S.\Psi$. For the completeness part, we need to prove, again by structural induction, that $\vdash_{\text{par}} \{wlp.S.\Psi\} S \{\Psi\}$. For more details, please refer to Appendix B. □

Total Correctness. Ranking functions play a central role in proving total correctness of while loop programs. Recall that in the classical case, a ranking function maps each reachable state during the execution of the loop body to an element of a well-founded ordered set (say, the set of nonnegative integers), such that the value decreases strictly after each iteration of the loop. Our proof rule for total correctness of while loops also heavily relies on the notion of ranking assertion. The following definition is inspired by the corresponding concept proposed in [7]. However, here we have to take into account possible nondeterministic choices in the loop body.

DEFINITION 4.3. *Let $\widehat{\Theta}$ be a quantum assertion. A set of quantum predicates $\{R_i^\eta : i \geq 0, \eta \in \llbracket S \rrbracket^{\mathbb{N}}\}$ is called a $\widehat{\Theta}$ -ranking assertion for **while** $\mathcal{M}[\bar{q}]$ **do** S **end** if for each η ,*

- (1) $\widehat{\Theta} \sqsubseteq_{\text{inf}} R_0^\eta$;
- (2) *the sequence $R_i^\eta, i \geq 0$, is decreasing with respect to \sqsubseteq ; that is, $R_0^\eta \sqsupseteq R_1^\eta \sqsupseteq \dots$. Furthermore, $\bigwedge_i R_i^\eta = 0$;*

(3) for any $i \geq 0$ and $\rho \in \mathcal{D}(\mathcal{H}_V)$,

$$\text{tr} \left(R_i^{\eta \rightarrow} \cdot \left(\eta_1 \circ \mathcal{P}_{\bar{q}}^1(\rho) \right) \right) \leq \text{tr} \left(R_{i+1}^{\eta} \cdot \rho \right). \quad (10)$$

This can also be more compactly written as

$$\mathcal{P}_{\bar{q}}^1 \circ \eta_1^\dagger \left(R_i^{\eta \rightarrow} \right) \sqsubseteq R_{i+1}^{\eta}. \quad (11)$$

As can be seen from the rule (WhileT) below, $\widehat{\Theta}$ is usually taken to be the precondition of the correctness formula we are concerned with, which is also a loop invariant of **while** $M[\bar{q}]$ **do** S **end**. The basic idea of $\widehat{\Theta}$ -ranking assertion is to provide a sequence of upper bounds on $\widehat{\Theta}$. We will explain it in more detail below.

For simplicity, assume $\widehat{\Theta} = \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta)$ for some quantum predicate M and assertion Θ . Let $M_0^\eta = 0$ and M_i^η , $i \geq 1$, be the quantum predicate representing the weakest precondition of M if the loop body is only executed $i - 1$ times; that is, for any ρ ,

$$\text{tr} \left(M_i^\eta \cdot \rho \right) = \text{tr} \left(M \cdot \mathcal{F}_{i-1}^\eta(\rho) \right)$$

where \mathcal{F}_i^η is defined in Eq. (1). The goal of $\widehat{\Theta}$ -ranking assertion R_i^η is then to make sure that

$$\mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta) \sqsubseteq_{\text{inf}} M_i^\eta + R_i^\eta. \quad (12)$$

Clause (1) of Definition 4.3 establishes the bound for $i = 0$, while clause (3), together with the assumption that $\mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta)$ is a loop invariant, guarantee that Eq. (12) holds inductively for all i . This can be shown by using Lemma 3.2. Finally, because of clause (2), when i tends to infinity, M_i^η alone acts as the upper bound, which in turn implies that $\mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta)$ is a valid precondition of M with respect to **while** $M[\bar{q}]$ **do** S **end** in the total correctness sense.

With the notion of ranking assertion, we can state the rule (WhileT) for while loops in total correctness as follows:

$$\text{(WhileT)} \quad \frac{\{\Theta\} S \left\{ \mathcal{P}_{\bar{q}}^0(\Psi) + \mathcal{P}_{\bar{q}}^1(\Theta) \right\} \text{ while } M[\bar{q}] \text{ do } S \text{ end } \{\Psi\}}{\left\{ \mathcal{P}_{\bar{q}}^0(\Psi) + \mathcal{P}_{\bar{q}}^1(\Theta) \right\} \text{ while } M[\bar{q}] \text{ do } S \text{ end } \{\Psi\}}$$

The proof system for total correctness is then defined as for partial correctness, except that the rule (While) is replaced by (WhileT), and rule (Abort) replaced by

$$\text{(AbortT)} \quad \{0\} \text{ abort } \{0\}.$$

We write $\vdash_{\text{tot}} \{\Theta\} S \{\Psi\}$ if the correctness formula $\{\Theta\} S \{\Psi\}$ can be derived using the proof system for total correctness. Again, we can prove the soundness and relative completeness of the proof system for total correctness.

THEOREM 4.2. *The proof system for total correctness is both sound and relatively complete with respect to the total correctness of nondeterministic quantum programs.*

PROOF. (Sketch) Again, the key idea for the proof here is to define the notion of weakest precondition for quantum programs.

The basic idea of using a ranking assertion to establish the soundness of rule (WhileT) has been intuitively discussed below Definition 4.3. For the completeness part, we can prove that the set of quantum predicates

$$R_i^\eta = \sum_{k=i}^{\infty} \mathcal{P}_{\bar{q}}^1 \circ \eta_1^\dagger \circ \dots \circ \mathcal{P}_{\bar{q}}^1 \circ \eta_k^\dagger \circ \mathcal{P}_{\bar{q}}^0(I),$$

representing the termination probability of the while loop after i iterations of the loop body, constitute a proper ranking assertion to prove that $\vdash_{\text{tot}} \{wp.\text{while}.\Theta\} \text{ while } \{\Theta\}$. For more details, please refer to Appendix B. \square

5 CASE STUDIES

To illustrate the effectiveness of the proof systems presented in the last section, we employ them to verify some simple quantum algorithms and protocols.

5.1 Three Qubit Quantum Error Correction Scheme

Recall from Example 4.1 that the correctness of the three-qubit quantum error correction scheme can be stated as follows: for any $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \in \mathcal{H}_2$,

$$\vdash_{\text{tot}} \{|\psi\rangle_q \langle \psi|\} \text{ ErrCorr } \{|\psi\rangle_q \langle \psi|\}. \quad (13)$$

First, we note that from rules (Skip) and (Unit),

$$\begin{aligned} & \vdash_{\text{tot}} \{[\alpha_0|000\rangle + \alpha_1|111\rangle\} \text{ skip } \{[\alpha_0|000\rangle + \alpha_1|111\rangle\} \\ & \vdash_{\text{tot}} \{[\alpha_0|000\rangle + \alpha_1|111\rangle\} q \text{ } * = X \{[\alpha_0|100\rangle + \alpha_1|011\rangle\} \\ & \vdash_{\text{tot}} \{[\alpha_0|000\rangle + \alpha_1|111\rangle\} q_1 \text{ } * = X \{[\alpha_0|010\rangle + \alpha_1|101\rangle\} \\ & \vdash_{\text{tot}} \{[\alpha_0|000\rangle + \alpha_1|111\rangle\} q_2 \text{ } * = X \{[\alpha_0|001\rangle + \alpha_1|110\rangle\}. \end{aligned}$$

Let M_i , $1 \leq i \leq 4$, be the four postconditions presented above respectively. Then from (Imp), we have for each $S \in \{\text{skip}, q \text{ } * = X, q_1 \text{ } * = X, q_2 \text{ } * = X\}$,

$$\{[\alpha_0|000\rangle + \alpha_1|111\rangle\}_{q,q_1,q_2} \ S \ \{M_1 + M_2 + M_3 + M_4\}.$$

Thus we have the following proof outline:

$$\begin{aligned} & \{[\alpha_0|0\rangle + \alpha_1|1\rangle\}_q \\ & q_1, q_2 := 0; \\ & \{[\alpha_0|000\rangle + \alpha_1|100\rangle\}_{q,q_1,q_2} \quad (\text{Init}) \\ & q, q_1 \text{ } * = CX; \\ & \{[\alpha_0|000\rangle + \alpha_1|110\rangle\}_{q,q_1,q_2} \quad (\text{Unit}) \\ & q, q_2 \text{ } * = CX; \\ & \{[\alpha_0|000\rangle + \alpha_1|111\rangle\}_{q,q_1,q_2} \quad (\text{Unit}) \\ & \text{skip } \square q \text{ } * = X \ \square q_1 \text{ } * = X \ \square q_2 \text{ } * = X; \\ & \{[\alpha_0|000\rangle + \alpha_1|111\rangle\} + [\alpha_0|001\rangle + \alpha_1|110\rangle \\ & \quad + [\alpha_0|010\rangle + \alpha_1|101\rangle] + [\alpha_0|100\rangle + \alpha_1|011\rangle\} \quad (\text{NDet}) \\ & q, q_2 \text{ } * = CX; \\ & \{[\alpha_0|000\rangle + \alpha_1|110\rangle\} + [\alpha_0|001\rangle + \alpha_1|111\rangle \\ & \quad + [\alpha_0|010\rangle + \alpha_1|100\rangle] + [\alpha_0|101\rangle + \alpha_1|011\rangle\} \quad (\text{Unit}) \\ & q, q_1 \text{ } * = CX; \\ & \{[\alpha_0|0\rangle + \alpha_1|1\rangle\}_q \otimes ([|00\rangle] + [|01\rangle] + [|10\rangle])_{q_1,q_2} \end{aligned}$$

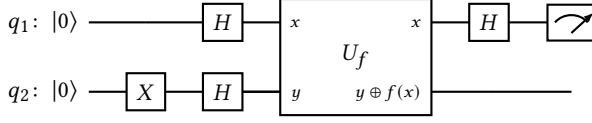


Figure 4: Quantum circuit for Deutsch algorithm.

```

+[\alpha_0|1\rangle + \alpha_1|0\rangle]_q \otimes [|11\rangle]_{q_1, q_2} \quad (Unit)
if  $\mathcal{M}[q_2]$  then
  \{[\alpha_0|0\rangle + \alpha_1|1\rangle\} \otimes [|0\rangle] + \{[\alpha_0|1\rangle + \alpha_1|0\rangle\} \otimes [|1\rangle\}
  if  $\mathcal{M}[q_1]$  then
    \{[\alpha_0|1\rangle + \alpha_1|0\rangle\}_q
    q *= X
    \{[\alpha_0|0\rangle + \alpha_1|1\rangle\}_q \quad (Unit)
  end
  \{[\alpha_0|0\rangle + \alpha_1|1\rangle\}_q \quad (Meas)
end
\{[\alpha_0|0\rangle + \alpha_1|1\rangle\}_q \quad (Meas)

```

With the soundness of our logic system, this completes the proof of Eq. (13).

5.2 Deutsch Algorithm

Deutsch algorithm [3] is one of the first quantum algorithms which demonstrate a speedup brought by quantum computing. Given a boolean function $f : \{0, 1\} \rightarrow \{0, 1\}$, Deutsch algorithm can tell whether $f(0)$ equals $f(1)$ or not with just a single evaluation of f . In the traditional description of the algorithm, a quantum oracle U_f that maps any state $|x\rangle \otimes |y\rangle$ to the state $|x\rangle \otimes |y \oplus f(x)\rangle$, where $x, y \in \{0, 1\}$, is employed. The circuit for Deutsch algorithm is shown in Figure 4. It claims that f is constant (meaning $f(0) = f(1)$) if the measurement outcome is 0; otherwise, it claims f is balanced (meaning $f(0) \neq f(1)$).

We now show how to describe Deutsch algorithm as a nondeterministic quantum program, and how to verify the correctness of it. To this end, we note that

$$U_f = \begin{cases} I_2 \otimes I_2 & \text{if } f(0) = f(1) = 0; \\ I_2 \otimes X & \text{if } f(0) = f(1) = 1; \\ CX & \text{if } f(0) = 0 \text{ and } f(1) = 1; \\ C^0X & \text{if } f(0) = 1 \text{ and } f(1) = 0. \end{cases}$$

where CX is the CNOT gate, and $C^0X = (X \otimes I_2) \cdot CX \cdot (X \otimes I_2)$ which applies X gate on the second qubit conditioning on the first qubit being $|0\rangle$. In the first two cases f is constant while in the last two cases f is balanced. Then Deutsch algorithm can be written as

```

Deutsch  $\triangleq$ 
  q1, q2 := 0;
  q1 *= H; q2 *= X; q2 *= H;
  if  $\mathcal{M}_{0,1}[q]$  then
    (q1, q2 *= CX)  $\square$  (q1, q2 *= C0X)
  else

```

```

  skip  $\square$  (q2 *= X)
end
  q1 *= H;
  measure q1

```

Here we introduce an auxiliary qubit q with unknown initial state and use the measurement outcome on q to choose f (or equivalently, U_f). Note that for each outcome we have two possibilities for U_f , denoted by a nondeterministic choice between them. The statement **measure** q_1 is defined as in Example 3.4. It is easy to prove from rules (Unit) and (Meas) that the following correctness formula is valid for any quantum assertion Θ :

$$\{|0\rangle_{q_1} \langle 0| \Theta |0\rangle_{q_1} \langle 0| + |1\rangle_{q_1} \langle 1| \Theta |1\rangle_{q_1} \langle 1|\} \text{ **measure** } q_1 \{ \Theta \}.$$

Note that at the end of the algorithm, both q and q_1 are in one of the computational basis $\{|0\rangle, |1\rangle\}$. The correctness of Deutsch algorithm can then be stated as follows:

$$\models_{tot} \{I\} \text{ Deutsch } \{(|00\rangle\langle 00| + |11\rangle\langle 11|)_{q, q_1}\}. \quad (14)$$

That is, the (classical) information encoded in q_1 when the program terminates coincides with that encoded in q , thus indicating correctly whether $f(0)$ equals $f(1)$ or not.

To prove Eq. (14), we first note from (Unit) that

$$\begin{aligned} & \vdash_{tot} \{|0\rangle_q |+\rangle_{q_1, q_2}\} \text{ skip } \{|0\rangle_q |+\rangle_{q_1, q_2}\} \\ & \vdash_{tot} \{|0\rangle_q |+\rangle_{q_1, q_2}\} q_2 *= X \{|0\rangle_q |+\rangle_{q_1, q_2}\} \\ & \vdash_{tot} \{|1\rangle_q |+\rangle_{q_1, q_2}\} q_1, q_2 *= CX \{|1\rangle_q |-\rangle_{q_1, q_2}\} \\ & \vdash_{tot} \{|1\rangle_q |+\rangle_{q_1, q_2}\} q_1, q_2 *= C^0X \{|1\rangle_q |-\rangle_{q_1, q_2}\} \end{aligned}$$

where $| \pm \rangle \triangleq \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$. Then we have the following proof outline:

```

\{I\}
q1, q2 := 0;
\{[|00\rangle]_{q_1, q_2}\} \quad (Init), (Seq)
q1 *= H; q2 *= X; q2 *= H;
\{[|0+\rangle]_{q, q_1, q_2} + [|1+\rangle]_{q, q_1, q_2}\} \quad (Unit), (Seq)
if  $\mathcal{M}_{0,1}[q]$  then
  \{[|1+\rangle]_{q, q_1, q_2}\}
  (q1, q2 *= CX)  $\square$  (q1, q2 *= C0X)
  \{[|1-\rangle]_{q, q_1, q_2}\} \quad (NDet)
else
  \{[|0+\rangle]_{q, q_1, q_2}\}
  skip  $\square$  (q2 *= X)
  \{[|0+\rangle]_{q, q_1, q_2}\} \quad (NDet)
end
\{[|0+\rangle]_{q, q_1, q_2} + [|1-\rangle]_{q, q_1, q_2}\} \quad (Imp), (Meas)
\{[|0+\rangle]_{q, q_1} + [|1-\rangle]_{q, q_1}\} \quad (Imp)
q1 *= H;
\{[|00\rangle]_{q, q_1} + [|11\rangle]_{q, q_1}\} \quad (Unit)
measure q1

```

$$\{[|00\rangle]_{q_1} + [|11\rangle]_{q_1}\} \quad (\text{Meas})$$

Finally, Eq. (14) follows from the soundness of our logic system.

5.3 A Nondeterministic Quantum Walk

To illustrate the utility of our logic system regarding quantum loops, let us consider a revised version of the nondeterministic quantum walk on a circle with four vertices presented in [12]. The walk uses a two-qubit system consisting of q_1 and q_2 as its principle system. It starts in the initial state $|00\rangle$, and at each step of walk, the following two unitary walk operators

$$W_1 \triangleq \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 1 & 0 & -1 \\ 1 & -1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & -1 & 1 \end{pmatrix}, \quad W_2 \triangleq \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 1 & 0 & 1 \\ -1 & 1 & -1 & 0 \\ 0 & 1 & 1 & -1 \\ 1 & 0 & -1 & -1 \end{pmatrix}$$

are applied on q_1 and q_2 consecutively. Here W_1 and W_2 are written with respect to the computational basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ of \mathcal{H}_{q_1, q_2} . However, the order in which these walk operators are applied is nondeterministically chosen. We further assume that there exists an absorbing boundary at the subspace spanned by $|10\rangle$; that is, after each walk step, a projective measurement $\mathcal{M} = \{P_0, P_1\}$ where

$$P_0 \triangleq |10\rangle\langle 10|, \quad P_1 \triangleq I_4 - |10\rangle\langle 10|$$

is applied. If the outcome corresponding to P_0 is observed, the whole process terminates; otherwise the next walk step continues.

The walk can be described as the following nondeterministic program:

```

QWalk  $\triangleq$ 
   $q_1, q_2 := 0;$ 
  while  $\mathcal{M}[q_1, q_2]$  do
     $(q_1, q_2 \ast= W_1; q_1, q_2 \ast= W_2)$ 
  □  $(q_1, q_2 \ast= W_2; q_1, q_2 \ast= W_1)$ 
  end

```

It has been shown in [12] that *QWalk* does not terminate if the left branch of the nondeterministic choice is always taken in each iteration. This can be easily seen from the fact that $W_2 W_1 |00\rangle = |00\rangle$. In the following, we show a stronger result using our proof system for nondeterministic quantum programs: in fact, *QWalk* does not terminate under any scheduler of the nondeterministic choice! For simplicity, we omit the subscript $\{q_1, q_2\}$ of the assertions and super-operators in the discussion below.

First, note that this non-termination property can be stated as follows:

$$\models_{par} \{I\} \text{ } QWalk \text{ } \{0\}. \quad (15)$$

The reason is, from Definition 4.2(2), Eq. (15) holds iff for any $\rho \in \mathcal{D}(\mathcal{H}_{q_1, q_2})$ and $\sigma \in \llbracket QWalk \rrbracket(\rho)$,

$$\text{tr}(\rho) \leq \text{tr}(\rho) - \text{tr}(\sigma).$$

Thus $\text{tr}(\sigma) = 0$, meaning that starting from any ρ , the probability of *QWalk* reaching any valid quantum state is 0.

To prove Eq. (15), we first compute using rule (Unit) that

$$\left\{ [|00\rangle] + \left[\frac{1}{\sqrt{2}} (|01\rangle + |11\rangle) \right] \right\}$$

$q_1, q_2 \ast= W_1;$

$$\left\{ \left[\frac{1}{\sqrt{3}} (|00\rangle + |01\rangle + |11\rangle) \right] + \left[\frac{1}{\sqrt{6}} (-|01\rangle + 2|10\rangle + |11\rangle) \right] \right\}$$

$$q_1, q_2 \ast= W_2;$$

$$\left\{ [|00\rangle] + \left[\frac{1}{\sqrt{2}} (|01\rangle + |11\rangle) \right] \right\}$$

and

$$\left\{ [|00\rangle] + \left[\frac{1}{\sqrt{2}} (|01\rangle + |11\rangle) \right] \right\}$$

$$\left\{ \left[\frac{1}{3} (-|00\rangle + 2|01\rangle + 2|11\rangle) \right] + \left[\frac{1}{3\sqrt{2}} (4|00\rangle + |01\rangle + |11\rangle) \right] \right\}$$

$$q_1, q_2 \ast= W_2;$$

$$\left\{ \left[\frac{1}{\sqrt{3}} (|00\rangle + |01\rangle - |11\rangle) \right] + \left[\frac{1}{\sqrt{6}} (2|00\rangle - |01\rangle + |11\rangle) \right] \right\}$$

$$q_1, q_2 \ast= W_1;$$

$$\left\{ [|00\rangle] + \left[\frac{1}{\sqrt{2}} (|01\rangle + |11\rangle) \right] \right\}$$

Now let $N \triangleq [|00\rangle] + \left[\frac{1}{\sqrt{2}} (|01\rangle + |11\rangle) \right]$, and S the body of the **while** loop of *QWalk*. Note that $\mathcal{P}^0(0) = 0$ and $\mathcal{P}^1(N) = N$ where for $i = 0, 1$, \mathcal{P}^i is the super-operator with a single Kraus operator P_i . Then by (NDet) we have

$$\vdash_{par} \{N\} S \{ \mathcal{P}^0(0) + \mathcal{P}^1(N) \} \quad (16)$$

Finally, we have

$$\{I\}$$

$$q_1, q_2 := 0;$$

$$\left\{ [|00\rangle] + \left[\frac{1}{\sqrt{2}} (|01\rangle + |11\rangle) \right] \right\} \quad (\text{Init})$$

while $\mathcal{M}[q_1, q_2]$ **do**

$$(q_1, q_2 \ast= W_1; q_1, q_2 \ast= W_2) \square (q_1, q_2 \ast= W_2; q_1, q_2 \ast= W_1)$$

end

$$\{0\}$$

(While), Eq. (16)

With the soundness theorem 4.1, this concludes the proof of Eq. (15).

6 A PROTOTYPE TOOL IMPLEMENTATION

We have illustrated the utility of our proof systems through a couple of examples in the previous section. It can be seen that even for such simple programs, carrying out formal verification is tedious (albeit routine) and involves a large number of matrix manipulations. To ease the burden on human users so that they can focus on more challenging parts such as specifying invariants for while loops, we implement a prototype proof assistant NQPV to automate routine parts of the verification process. Currently, NQPV only supports partial correctness; verification of total correctness is left as future work.

NQPV is a pure Python project that does not rely on existing theorem provers such as Isabelle and Coq. We make this decision taking into account the following factors: (1) Python and its various libraries provide powerful matrix manipulation capabilities that facilitate us to compute the pre- and post-conditions of quantum programs and determine the \sqsubseteq_{inf} relation between quantum assertions easily; (2) The main purpose of NQPV is to illustrate the practicality of our proposed logic system, so the ease of use is our top priority. Python's popularity and rich matrix libraries make it

convenient and natural to describe quantum programs and assertions; (3) Symbolic verification in Python is difficult, which limits the quantum programs that NQPV can express and verify. But in return, most verification steps can be done automatically in NQPV.

NQPV relies on the following Python packages: `ply` for syntax analysis, `numpy` for operator calculation, and `cvxpy` for the SDP solver. NQPV utilises a simple language to define and manage terms of operators and proofs.

6.1 User Inputs

NQPV takes a correctness formula defined in Sec. 4.1 as the input. However, to automate the proof for while programs, loop invariants are also expected whenever a while structure is encountered. For illustration, we reconsider the quantum walk example in Sec. 5.3. The code is shown as follows:

```
def invN := load "invN.npy" end
// more operators imported ...
def pf := proof[q1 q2] :
  { I[q1] };
  [q1 q2] :=0;
  { inv: invN[q1 q2] };
  while MQWalk[q1 q2] do
    ( [q1 q2] *= W1; [q1 q2] *= W2
      # [q1 q2] *= W2; [q1 q2] *= W1 )
  end;
  { Zero[q1] }
end
show pf end
```

Here the unitary operators $W1$, $W2$ and measurement `MQWalk` are defined in Sec. 5.3, and they should be input by the user as numpy matrices. Note that some identifiers such as `I` and `Zero` are reserved for commonly used unitary operators, hermitian operators, and measurements. Furthermore, loop invariants (`invN` in this example) specified by the user for while programs should also be numpy matrices.

In addition to the main program describing behaviours of the quantum walk, the above code also includes two extra lines `{ I[q1] }` and `{ Zero[q1] }` to represent the precondition and postcondition of the program. Hence, the body of proof term `pf` indeed describes the correctness formula presented in Eq. (15). Finally, the keyword `inv` marks a loop invariant, and the `show` command in the last line outputs the proof outline generated by NQPV.

To further simplify usage, NQPV also allows users to omit preconditions and specify only postconditions. In this case, NQPV outputs the weakest precondition it can compute using the given postcondition and user-supplied loop invariants.

6.2 Verification Process

After successfully parsing the input, NQPV inductively constructs proofs according to the logic system in Fig. 3 in an automated way. The strategy is to calculate the weakest preconditions in the backward direction, starting from the postcondition of the whole program. For while loops, NQPV will check if the loop invariant provided by the user is a valid one.

In the end, the assistant compares the verification condition and the precondition proposed by the user (details will be given in the next subsection) and then generates the final result. Again, the following shows the output of NQPV for the quantum walk example:

```
proof [q1 q2] :
  { I[q1] };
  { VAR2[q1 q2] }; // the Veri. Con.
  [q1 q2] :=0;
  { invN[q1 q2] };
  { inv: invN[q1 q2] };
  while MQWalk[q1 q2] do
    { invN[q1 q2] };
    ( { invN[q1 q2] }; [q1 q2] *= W1;
      { VAR0[q1 q2] }; [q1 q2] *= W2
      # { invN[q1 q2] }; [q1 q2] *= W2;
        { VAR1[q1 q2] }; [q1 q2] *= W1 )
  end;
  { Zero[q1] }
```

As is shown, every sub-program statement is annotated with the corresponding pre- and postconditions. Some of them are already defined, such as `invN`. Other operators, such as `VAR0`, are generated by NQPV to represent predicates used in the proof outline. The detailed information of these operators can be shown using the `show` command. For example, `show VAR2 end` for this example will return the two-qubit identity operator, meaning that the verification condition determined by the given postcondition and the loop invariant is `I[q1 q2]` (thus the original correctness formula is valid).

In addition to checking the validity of a correctness formula, NQPV can also verify if a user-supplied operator is indeed a loop invariant. For example, if we change the `invN[q1 q2]` in the code in Sec. 6.1 into `P0[q1]` where $P0 = |0\rangle\langle 0|$, we will get an error message:

```
Error:
Order relation not satisfied:
  { P0[q1] } <= { VAR0[q1 q2] VAR3[q1 q2] }
...
Error: The predicate '{ P0[q1] }' is not
a valid loop invariant.
...
```

6.3 Determining the \sqsubseteq_{inf} Relation

One of the key steps in verifying a correctness formula is to determine if two assertions (sets of quantum predicates) satisfy the \sqsubseteq_{inf} relation. For the sake of implementability, we assume all the quantum assertions allowed in NQPV are finite ones.

LEMMA 6.1. *Let Θ and Ψ are finite set of quantum predicates. Then $\Theta \sqsubseteq_{inf} \Psi$ iff for any $N \in \Psi$,*

$$\forall \rho \in \mathcal{D}(\mathcal{H}_V), \exists M \in \Theta, \text{tr}(M\rho) \leq \text{tr}(N\rho).$$

PROOF. From the finiteness of Θ and Ψ ,

$$\Theta \sqsubseteq_{inf} \Psi$$

$$\begin{aligned}
&\Leftrightarrow \forall \rho \in \mathcal{D}(\mathcal{H}_V), \min_{M \in \Theta} \text{tr}(M\rho) \leq \min_{N \in \Psi} \text{tr}(N\rho) \\
&\Leftrightarrow \forall \rho \in \mathcal{D}(\mathcal{H}_V), \forall N \in \Psi, \exists M \in \Theta, \text{tr}(M\rho) \leq \text{tr}(N\rho) \\
&\Leftrightarrow \forall N \in \Psi, \forall \rho \in \mathcal{D}(\mathcal{H}_V), \exists M \in \Theta, \text{tr}(M\rho) \leq \text{tr}(N\rho). \quad \square
\end{aligned}$$

With Lemma 6.1, we have the following algorithm to determine if $\Theta \sqsubseteq_{\text{inf}} \Psi$ for finite sets Θ and Ψ :

- If $\Theta = \{M\}$ is a singleton, then $\Theta \sqsubseteq_{\text{inf}} \Psi$ iff $M \sqsubseteq N$ for all $N \in \Psi$. This can be done by simply checking if the eigenvalues of $N - M$ are all nonnegative.
- Otherwise, for each $N \in \Psi$ we try to solve the following SDP problem:

$$\begin{array}{ll}
\text{minimize} & 0 \\
\text{subject to} & \forall M \in \Theta, \text{tr}(N\rho) \leq \text{tr}(M\rho) - \epsilon \\
& 0 \sqsubseteq \rho
\end{array}$$

Here ϵ is a user-defined precision that is sufficiently small but positive. This is required by SDP solvers such as MOSEK and CVX. Another reason for the introduction of ϵ is to make the feasible region close and thus the whole problem an SDP one. Obviously, if any of the $|\Psi|$ SDP problems returns a solution, then $\Theta \not\sqsubseteq_{\text{inf}} \Psi$. However, because of the precision parameter, we cannot make sure if $\Theta \sqsubseteq_{\text{inf}} \Psi$ holds even none of the SDP problems returns a solution. Nevertheless, the probability of getting incorrect answers can be negligible if ϵ is taken sufficiently small.

6.4 Related Tools

Program verification can also be done within an interactive theorem prover such as Isabelle and Coq. The first theorem prover for quantum programs is QHLProver [13], which implements the program logic in [28] using Isabelle/HOL. CoqQ [29] implements the same logic in Coq, with a general and abstract representation of linear operators. Qrhl-tool [25] is a verification tool embedded in Isabelle that supports relational verification of quantum programs.

To make a fair comparison, we investigate the difference between NQPV and the existing tools in the following aspects (we only take CoqQ as an example, as QHLProver and Qrhl-tool are built in a similar way to CoqQ):

Reliability. For CoqQ, inference rules are not only defined, but their soundness with respect to the denotational semantics (also formally defined in CoqQ) is proved within Coq. In contrast, the soundness of inference rules is not checked in NQPV. Nevertheless, a rigorous proof for the soundness of our logic system has been given in Theorem 4.1.

Expressiveness. Existing tools only deal with deterministic quantum programs, while our prototype implementation supports nondeterministic ones. On the other hand, symbols can be handled in CoqQ, as in theorem provers; whereas only numerical terms are allowed in NQPV, as in Python. In particular, CoqQ is suitable for verifying general (i.e., with an arbitrary number of qubits) algorithms such as Grover algorithm, since the qubit number appears as a symbol in CoqQ.

Automation. Due to the abstract representation of linear operators in CoqQ, many properties of quantum predicates are not easy to

prove and have to be shown by the user. However, NQPV can take advantage of powerful Python libraries, so many proofs can be automated without user assistance. Therefore, the proof script for NQPV is usually much shorter than the proof written in CoqQ.

Usability. Using the Coq library MathComp, CoqQ develops an abstract representation of linear operators directly based on Hilbert spaces, and a smart way to describe them using labelled Dirac notations. Consequently, users of CoqQ are supposed to have sufficient knowledge of Coq, especially the way linear operators and computer programs are specified in it. In NQPV, operators are given concretely as numpy matrices, and programs are represented in a natural way familiar to ordinary Python programmers. As mentioned at the beginning of this section, we take this easy path to implement the prototype since our main purpose is to illustrate the utility of our logic system, rather than a full-blown verification tool.

Performance. The performance of CoqQ is determined by the proofs provided, while that of NQPV is determined by the calculation backend, which in the worst case is exponential in the number of qubits. In comparison, it takes a few seconds to verify the general Grover algorithm in CoqQ, and 90 seconds for the 13-qubit Grover algorithm in NQPV.

7 CONCLUSION

In this paper, we consider quantum programs where nondeterministic choices are allowed. Denotational semantics of such programs is given by lifting semantics of deterministic quantum programs, and we argue that this is the only natural way to define nondeterminism in the quantum setting. For the purpose of verification, we take sets of hermitian operators on the associated Hilbert space to be the assertions of quantum states, and propose two proof systems, for partial and total correctness respectively. We show that they are both sound and relatively complete with respect to the corresponding correctness notions. Simple quantum algorithms and protocols, such as the three-qubit bit-flip quantum error correction scheme, Deutsch algorithm, and a nondeterministic quantum walk, are analysed to demonstrate the utility of our proof systems. A lightweight prototype of a proof assistant is implemented to aid in the automated reasoning of nondeterministic quantum programs.

For future works, we are going to investigate how to make use of the nondeterministic choice construct and the verification technique proposed in this paper for quantum program refinement. How to incorporate angelic nondeterminism into the picture is also an interesting topic for future study.

ACKNOWLEDGMENTS

The authors thank Profs Mingsheng Ying and Sanjiang Li for inspiring discussions. This work is partially supported by the National Key R&D Program of China under Grant No 2018YFA0306704 and the Australian Research Council under Grant No. DP220102059. YX was also partially funded by the Sydney Quantum Academy.

A WEAKEST (LIBERAL) PRECONDITION SEMANTICS

This section presents an alternative semantics for nondeterministic quantum programs in terms of the weakest (liberal) preconditions. It turns out to be equivalent to the denotational semantics given in Sec. 3.2. More importantly, the weakest (liberal) precondition semantics provides a powerful proof technique for the soundness and completeness of our logic systems, as shown in the next section.

Let $S \in \text{Prog}$. The *weakest precondition semantics* $wp.S$ and *weakest liberal precondition semantics* $wlp.S$ of S are both mappings in $2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$ defined inductively in Fig. 5. To simplify notation, we use xp to denote wp or wlp whenever it is applicable for both of them.

The definitions are similar to those of deterministic quantum programs presented in [28]. Again, because of the possible nondeterministic choices in the loop body S , we have incorporated schedulers $\eta \in \llbracket S \rrbracket^{\mathbb{N}}$ in the weakest (liberal) precondition semantics of **while** $M[\bar{q}]$ **do** S **end**.

The following lemma shows a duality between the denotational and weakest (liberal) precondition semantics of quantum programs.

LEMMA A.1. *Let $S \in \text{Prog}$, $\rho \in \mathcal{D}(\mathcal{H}_{\mathcal{V}})$, M be a quantum predicate, and Θ a quantum assertion. Then*

- (1) $wp.S.M = \{\mathcal{E}^\dagger(M) : \mathcal{E} \in \llbracket S \rrbracket\}$;
- (2) $wlp.S.M = \{\mathcal{E}^\dagger(M) + I - \mathcal{E}^\dagger(I) : \mathcal{E} \in \llbracket S \rrbracket\}$;
- (3) $\text{Exp}(\rho \models wp.S.\Theta) = \inf \{\text{Exp}(\sigma \models \Theta) : \sigma \in \llbracket S \rrbracket(\rho)\}$;
- (4) $\text{Exp}(\rho \models wlp.S.\Theta) = \inf \{\text{Exp}(\sigma \models \Theta) + \text{tr}(\rho) - \text{tr}(\sigma) : \sigma \in \llbracket S \rrbracket(\rho)\}$.

PROOF. We prove clause (1) by induction on the structure of S . The basis cases are easy from the definition. For the induction step, we only show the following two cases as examples.

- Let $S \triangleq \text{if } M[\bar{q}] \text{ then } S_1 \text{ else } S_0 \text{ end}$. Then

$$\begin{aligned} wp.S.M &= \mathcal{P}_q^1(wp.S_1.M) + \mathcal{P}_q^0(wp.S_0.M) \\ &= \left\{ \mathcal{P}_q^1 \circ \mathcal{F}^\dagger(M) : \mathcal{F} \in \llbracket S_1 \rrbracket \right\} + \left\{ \mathcal{P}_q^0 \circ \mathcal{E}^\dagger(M) : \mathcal{E} \in \llbracket S_0 \rrbracket \right\} \\ &= \left\{ (\mathcal{F} \circ \mathcal{P}_q^1 + \mathcal{E} \circ \mathcal{P}_q^0)^\dagger(M) : \mathcal{E} \in \llbracket S_0 \rrbracket, \mathcal{F} \in \llbracket S_1 \rrbracket \right\} \\ &= \left\{ \mathcal{G}^\dagger(M) : \mathcal{G} \in \llbracket S \rrbracket \right\}. \end{aligned}$$

- Let $S \triangleq \text{while } M[\bar{q}] \text{ do } S' \text{ end}$. For any $\eta \in \llbracket S' \rrbracket^{\mathbb{N}}$ and $i \geq 0$, let M_i^η be defined as in Fig. 5 for $wp.(\text{while } M[\bar{q}] \text{ do } S' \text{ end}).M$, and \mathcal{F}_i^η be defined as in Figure 2 for $\llbracket \text{while } M[\bar{q}] \text{ do } S' \text{ end} \rrbracket$. First, from Lemma 3.2 it is easy to show by induction on i that

$$\forall i \geq 0, \forall \eta \in \llbracket S' \rrbracket^{\mathbb{N}}, M_i^\eta = (\mathcal{F}_i^\eta)^\dagger(M). \quad (17)$$

Thus we have

$$\bigvee_{i \geq 0} M_i^\eta = \bigvee_{i \geq 0} (\mathcal{F}_i^\eta)^\dagger(M) = \left(\bigvee_{i \geq 0} \mathcal{F}_i^\eta \right)^\dagger(M),$$

and so

$$wp.S.M = \left\{ \bigvee_{i \geq 0} M_i^\eta : \eta \in \llbracket S' \rrbracket^{\mathbb{N}} \right\} = \left\{ \mathcal{E}^\dagger(M) : \mathcal{E} \in \llbracket S \rrbracket \right\}$$

from the definition of $\llbracket S \rrbracket$.

The proof for clause (2) is similar; the only difference in proving the case for **while** $M[\bar{q}]$ **do** S' **end** is that instead of Eq. (17) we have to prove

$$\forall i \geq 0, \forall \eta \in \llbracket S' \rrbracket^{\mathbb{N}}, I - M_i^\eta = (\mathcal{F}_i^\eta)^\dagger(I - M)$$

where M_i^η 's are defined in a similar way as in Fig. 5 but for $wlp.(\text{while } M[\bar{q}] \text{ do } S' \text{ end}).M$. However, this is also easy by induction on i .

For clause (3), we calculate

$$\begin{aligned} \text{Exp}(\rho \models wp.S.\Theta) &= \inf \{\text{tr}(M\rho) : N \in \Theta, M \in wp.S.N\} \\ &= \inf \left\{ \text{tr}(\mathcal{E}^\dagger(N)\rho) : N \in \Theta, \mathcal{E} \in \llbracket S \rrbracket \right\} \\ &= \inf \left\{ \text{tr}(N\mathcal{E}(\rho)) : N \in \Theta, \mathcal{E} \in \llbracket S \rrbracket \right\} \\ &= \inf \left\{ \text{tr}(N\sigma) : N \in \Theta, \sigma \in \llbracket S \rrbracket(\rho) \right\} \\ &= \inf \left\{ \text{Exp}(\sigma \models \Theta) : \sigma \in \llbracket S \rrbracket(\rho) \right\} \end{aligned}$$

where the second equality is from clause (1). Finally, clause (4) follows from (2) with a similar argument. \square

The next lemma shows that the weakest (liberal) precondition of a while program is a fixed point of some functor on \mathcal{A} .

LEMMA A.2. *For any quantum predicate M , let $\Theta = xp.(\text{while } M[\bar{q}] \text{ do } S \text{ end}).M$. Then*

$$\Theta = \mathcal{P}_q^0(M) + \mathcal{P}_q^1(xp.S.\Theta).$$

PROOF. Let $\text{while} \triangleq \text{while } M[\bar{q}] \text{ do } S \text{ end}$. Then

$$\begin{aligned} wp.\text{while}.M &= \left\{ \mathcal{E}^\dagger(M) : \mathcal{E} \in \llbracket \text{while} \rrbracket \right\} \\ &= \left\{ \mathcal{P}_q^0(M) + \mathcal{P}_q^1 \circ \mathcal{F}^\dagger \circ \mathcal{G}^\dagger(M) : \mathcal{F} \in \llbracket S \rrbracket, \mathcal{G} \in \llbracket \text{while} \rrbracket \right\} \\ &= \mathcal{P}_q^0(M) + \mathcal{P}_q^1 \left(wp.S. \left\{ \mathcal{G}^\dagger(M) : \mathcal{G} \in \llbracket \text{while} \rrbracket \right\} \right) \\ &= \mathcal{P}_q^0(M) + \mathcal{P}_q^1(wp.S.(wp.\text{while}.M)) \end{aligned}$$

where the second equality is from Eq. (3) while the other ones follow from Lemma A.1(1). The case for wlp is similar. \square

To conclude this section, we prove a lemma which is useful in our later discussion.

LEMMA A.3. *For any correctness formula $\{\Theta\} S \{\Psi\}$,*

- (1) $\models_{tot} \{\Theta\} S \{\Psi\}$ if and only if $\Theta \sqsubseteq_{inf} wp.S.\Psi$;
- (2) $\models_{par} \{\Theta\} S \{\Psi\}$ if and only if $\Theta \sqsubseteq_{inf} wlp.S.\Psi$.

PROOF. For clause (1), we compute

$$\begin{aligned} \models_{tot} \{\Theta\} S \{\Psi\} &\Leftrightarrow \forall \rho, \text{Exp}(\rho \models \Theta) \leq \inf \{\text{Exp}(\sigma \models \Psi) : \sigma \in \llbracket S \rrbracket(\rho)\} \\ &\Leftrightarrow \forall \rho, \text{Exp}(\rho \models \Theta) \leq \text{Exp}(\rho \models wlp.S.\Psi) \\ &\Leftrightarrow \forall \rho, \inf \{\text{tr}(M\rho) : M \in \Theta\} \leq \inf \{\text{tr}(N\rho) : N \in wlp.S.\Psi\} \\ &\Leftrightarrow \Theta \sqsubseteq_{inf} wp.S.\Psi \end{aligned}$$

where the second equivalence is from Lemma A.1(3) while the third one from Eq (8). The case for wlp in clause (2) is similar. \square

$$\begin{aligned}
xp.\text{skip}.M &= M & xp.(\bar{q} * = U).M &= U_{\bar{q}}^{\dagger} M U_{\bar{q}} \\
xp.(\bar{q} := 0).M &= \sum_{i=0}^{2^n-1} |i\rangle_{\bar{q}} \langle 0|M|0\rangle_{\bar{q}} \langle i| & xp.S.\Theta &= \bigcup_{M \in \Theta} xp.S.M \\
xp.(S_0; S_1).M &= xp.S_0.(xp.S_1.M) & xp.(S_0 \square S_1).M &= xp.S_0.M \cup xp.S_1.M \\
wlp.\text{abort}.M &= \{I\} & wp.\text{abort}.M &= \{0\} \\
xp.(\text{if } M[\bar{q}] \text{ then } S_1 \text{ else } S_0 \text{ end}).M &= \mathcal{P}_{\bar{q}}^1(xp.S_1.M) + \mathcal{P}_{\bar{q}}^0(xp.S_0.M) \\
wlp.(\text{while } M[\bar{q}] \text{ do } S \text{ end}).M &= \left\{ \bigwedge_{i \geq 0} M_i^\eta : \eta \in \llbracket S \rrbracket^{\mathbb{N}} \right\} \text{ where for each } \eta, M_0^\eta \triangleq I, \text{ and for any } i \geq 0, \\
& M_{i+1}^\eta = \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1 \left(\eta_1^\dagger \left(M_i^{\eta \rightarrow} \right) \right) + I - \eta_1^\dagger(I) \\
wp.(\text{while } M[\bar{q}] \text{ do } S \text{ end}).M &= \left\{ \bigvee_{i \geq 0} M_i^\eta : \eta \in \llbracket S \rrbracket^{\mathbb{N}} \right\} \text{ where for each } \eta, M_0^\eta \triangleq 0, \text{ and for any } i \geq 0, \\
& M_{i+1}^\eta = \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1 \left(\eta_1^\dagger \left(M_i^{\eta \rightarrow} \right) \right)
\end{aligned}$$

Figure 5: Weakest (liberal) precondition semantics for quantum programs, where $xp \in \{wp, wlp\}$.

B PROOF DETAILS

B.1 Proof of Theorem 4.1

Soundness: We need only to show that each rule in Fig. 3 is valid in the sense of partial correctness. Take the rule (While) as an example; the others are simpler. From (Union), it suffices to consider the special case when $\Psi = \{M\}$ for some quantum predicate M . Let

$$\models_{par} \{\Theta\} S \left\{ \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta) \right\}.$$

Then $\Theta \sqsubseteq_{inf} wlp.S. \left(\mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta) \right)$. We now prove by induction on i that

$$\forall i \geq 0, \forall \eta \in \llbracket S \rrbracket^{\mathbb{N}}, \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta) \sqsubseteq_{inf} M_i^\eta$$

where M_i^η is defined as in Fig. 5 for the wlp semantics of **while** \triangleq **while** $M[\bar{q}]$ **do** S **end**. The case when $i = 0$ is trivial. Then we calculate

$$\begin{aligned}
& \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta) \\
& \sqsubseteq_{inf} \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1 \left(wlp.S. \left(\mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta) \right) \right) \\
& \sqsubseteq_{inf} \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1 \left(wlp.S.M_i^{\eta \rightarrow} \right) \\
& \sqsubseteq_{inf} \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1 \left(\eta_1^\dagger \left(M_i^{\eta \rightarrow} \right) \right) + I - \eta_1^\dagger(I) \\
& = M_{i+1}^\eta,
\end{aligned}$$

where the first inequality follows from Lemma 4.2(1), the second one from the induction hypothesis and that $wlp.S$ is monotonic with respect to \sqsubseteq_{inf} , and the third one from Lemma A.1 and that $\eta_1 \in \llbracket S \rrbracket$. Thus

$$\mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta) \sqsubseteq_{inf} wlp.\text{while}.M,$$

and so

$$\models_{par} \left\{ \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta) \right\} \text{while } \{M\}$$

from Lemma A.3.

Completeness: By Lemma A.3 and the (Imp) rule, it suffices to show that for any Θ and S' ,

$$\vdash_{par} \{wlp.S'.\Theta\} S' \{\Theta\}.$$

Again, we take the case for loops as an example. For any $M \in \Theta$, let $\Psi \triangleq wlp.\text{while}.M$. By induction, we have $\vdash_{par} \{wlp.S.\Psi\} S \{\Psi\}$. Note from Lemma A.2 that $\Psi = \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(wlp.S.\Psi)$. Then we have $\vdash_{par} \{\Psi\} \text{while } \{M\}$ by rule (While). Finally, the desired result

$$\vdash_{par} \{wlp.\text{while}.\Theta\} \text{while } \{\Theta\}$$

follows from (Union) and Lemma 4.2(2).

B.2 Proof of Theorem 4.2

Soundness: We need only to show that each rule of the proof system is valid in the sense of total correctness. Take rule (WhileT) as an example. Again, from (Union), it suffices to consider the special case when $\Psi = \{M\}$ for some quantum predicate M . Let

$$\models_{tot} \{\Theta\} S \left\{ \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta) \right\},$$

and $\{R_i^\eta : i \geq 0, \eta \in \llbracket S \rrbracket^{\mathbb{N}}\}$ be a $\mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta)$ -ranking assertion for **while** $M[\bar{q}]$ **do** S **end**. Then $\Theta \sqsubseteq_{inf} wp.S. \left(\mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta) \right)$. We now prove by induction on i that

$$\forall i \geq 0, \forall \eta \in \llbracket S \rrbracket^{\mathbb{N}}, \mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta) \sqsubseteq_{inf} M_i^\eta + R_i^\eta$$

where M_i^η is defined as in Fig. 5 for the wp semantics of **while** $M[\bar{q}]$ **do** S **end**. The case when $i = 0$ is from the assumption that $\mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta) \sqsubseteq_{inf} R_0^\eta$. Then we calculate

$$\mathcal{P}_{\bar{q}}^0(M) + \mathcal{P}_{\bar{q}}^1(\Theta)$$

$$\begin{aligned}
& \sqsubseteq_{\text{inf}} \mathcal{P}_q^0(M) + \mathcal{P}_q^1 \left(wp.S. \left(\mathcal{P}_q^0(M) + \mathcal{P}_q^1(\Theta) \right) \right) \\
& \sqsubseteq_{\text{inf}} \mathcal{P}_q^0(M) + \mathcal{P}_q^1 \left(wp.S. \left(M_i^{\eta \rightarrow} + R_i^{\eta \rightarrow} \right) \right) \\
& \sqsubseteq_{\text{inf}} \mathcal{P}_q^0(M) + \mathcal{P}_q^1 \left(\eta_1^\dagger \left(M_i^{\eta \rightarrow} \right) \right) + \mathcal{P}_q^1 \left(\eta_1^\dagger \left(R_i^{\eta \rightarrow} \right) \right) \\
& \sqsubseteq_{\text{inf}} M_{i+1}^\eta + R_{i+1}^\eta,
\end{aligned}$$

where the first inequality follows from Lemma 4.2(1), the second one from the induction hypothesis and that $wp.S$ is monotonic with respect to \sqsubseteq_{inf} , the third one from Lemma A.1 and that $\eta_1 \in \llbracket S \rrbracket$, and the last one from Eq. (11). Thus

$$\mathcal{P}_q^0(M) + \mathcal{P}_q^1(\Theta) \sqsubseteq_{\text{inf}} wp.(\text{while } M[\bar{q}] \text{ do } S \text{ end}).M$$

by noting that $\bigwedge_i R_i^\eta = 0$ for any η , and so

$$\models_{\text{tot}} \left\{ \mathcal{P}_q^0(M) + \mathcal{P}_q^1(\Theta) \right\} \text{ while } M[\bar{q}] \text{ do } S \text{ end } \{M\}$$

from Lemma A.3.

Completeness: By Lemma A.3 and the (Imp) rule, it suffices to show that for any Θ and S' ,

$$\vdash_{\text{tot}} \{wp.S'.\Theta\} S' \{\Theta\}.$$

Again, we take the case for while loops as an example. Let $\text{while } \bar{q} \text{ do } S \text{ end}$ and $M \in \Theta$. By induction, we have $\vdash_{\text{tot}} \{wp.S.\Psi\} S \{\Psi\}$ where $\Psi \triangleq wp.\text{while}.M$. Note also from Lemma A.2 that $\Psi = \mathcal{P}_q^0(M) + \mathcal{P}_q^1(wp.S.\Psi)$. To finish the proof, we have to construct a Ψ -ranking assertion for **while**.

For any $\eta \in \llbracket S \rrbracket^{\mathbb{N}}$ and $k \geq 0$, let

$$R_k^\eta = \sum_{i=k}^{\infty} \mathcal{P}_q^1 \circ \eta_1^\dagger \circ \dots \circ \mathcal{P}_q^1 \circ \eta_1^\dagger \circ \mathcal{P}_q^0(I). \quad (18)$$

We would like to show that the set of R_k^η satisfy the conditions in Definition 4.3. First, we prove by induction on i that

$$\forall i \geq 0, \forall \eta \in \llbracket S \rrbracket^{\mathbb{N}}, M_i^\eta \sqsubseteq R_0^\eta \quad (19)$$

where M_i^η is defined as in Fig. 5 for $wp.\text{while}.M$, which then implies that $\Psi \sqsubseteq_{\text{inf}} R_0^\eta$. The base case of Eq. (19) where $i = 0$ is trivial since $M_0^\eta = 0$. We further calculate that

$$\begin{aligned}
M_{i+1}^\eta &= \mathcal{P}_q^0(M) + \mathcal{P}_q^1 \left(\eta_1^\dagger \left(M_i^{\eta \rightarrow} \right) \right) \\
&\sqsubseteq \mathcal{P}_q^0(I) + \mathcal{P}_q^1 \left(\eta_1^\dagger \left(R_0^{\eta \rightarrow} \right) \right) \\
&= R_0^\eta.
\end{aligned}$$

For the second condition of Definition 4.3, we note that from Eq. (18), $R_{k+1}^\eta \sqsubseteq R_k^\eta$ and $\bigwedge_k R_k^\eta = 0$. Furthermore, it is easy to see that $R_{k+1}^\eta = \mathcal{P}_q^1 \left(\eta_1^\dagger \left(R_k^{\eta \rightarrow} \right) \right)$.

Now using rule (WhileT), we have $\vdash_{\text{tot}} \{\Psi\} \text{ while } \{M\}$. Finally, the desired result

$$\vdash_{\text{tot}} \{wp.\text{while}.\Theta\} \text{ while } \{\Theta\}$$

follows from (Union) and Lemma 4.2(2).

C ARTIFACT APPENDIX

C.1 Abstract

NQPV is a verification assistant prototype of nondeterministic quantum programs. It implements the verification logic of partial correctness in the numerical form, with soundness guaranteed by the theory and experiments. It is a Python project, depending on numpy, cvxpy and ply packages. The three examples in our paper: the three-qubit bit-flip quantum error correction scheme, Deutsch algorithm and quantum walk, are encoded and verified numerically. The evaluation is integrated into a step-by-step Jupyter notebook. It shows that the tool can check the program syntax, automatically calculate the verification condition, check whether it is satisfied by the precondition, and return a proof outline of the correctness formula if so. Compared to other verification tools based on theorem provers like Coq or Isabelle, NQPV is weaker in expressiveness but stronger in automation. The whole evaluation can be conducted using a computer with 32 GB memory.

C.2 Artifact check-list (meta-information)

- **Algorithm:** Verification of nondeterministic quantum programs.
- **Run-time environment:** Python 3. The project is developed with Python 3.9.
- **Metrics:** Expressiveness of the programs and quantum assertions; Automation of the verification; Time and memory consumption for the verification of larger programs.
- **Output:** Numerical verification conditions saved in a computer file.
- **Experiments:** Integrated in a Jupyter notebook.
- **How much disk space required (approximately)?:** Only the performance test is resource consuming. 8 GB is sufficient for the experiments here.
- **How much time is needed to prepare workflow (approximately)?:** 5 minutes.
- **How much time is needed to complete experiments (approximately)?:** 15 minutes to go through the Jupyter notebook.
- **Publicly available:** Yes, on GitHub, Zenodo and PyPI.
- **Code licenses (if publicly available)?:** Apache-2.0
- **Archived (provide DOI)?:** <https://doi.org/10.5281/zenodo.7564087>

C.3 Description

C.3.1 How to access. This is a Python project on GitHub: <https://github.com/Lucifer1008/NQPV>. For evaluation purpose, please clone the "Article Release". The project itself is about 1 MB in size.

This project is also uploaded on PyPI as a package: <https://pypi.org/project/NQPV/>

C.3.2 Hardware dependencies. Only the performance test is resource consuming. A laptop computer with 32 GB memory and 8 GB disk storage is sufficient for the evaluation.

C.3.3 Software dependencies. Python packages: numpy for matrix calculation, cvxpy for the SDP solver, and ply for syntax parsing.

C.4 Installation

Experiments with the source code:

- (1) Prepare a Python 3 environment (This project was developed with Python 3.9).
- (2) Install package dependencies with this command:
pip install cvxpy ply
- (3) Clone the NQPV project or download the "Article Release".

- (4) Open a console at the root folder of the project and run the test script with


```
python test_install.py
```

 If no error is reported, then the installation is successfully completed.

NQPV as a Python package: run the command `pip install NQPV`

C.5 Experiment workflow

Find the Jupyter notebook `evaluate.ipynb` in the root folder of source. This notebook contains a brief introduction to the tool and integrates the verification experiments of all examples in our paper.

Every individual experiment is done with a `.nqpv` file containing the NQPV program code, and a `.py` python script to prepare the operators and invoke the verification.

C.6 Evaluation and expected results

The three examples in the paper: the three-qubit bit-flip quantum error correction scheme, Deutsch algorithm and quantum walk are encoded as program codes in the Jupyter notebook. The notebook also contains some examples to demonstrate the ability of NQPV to reject false correctness claims. The required operators (special unitary operators or measurements) are prepared and saved as `numpy` matrices.

It is expected that the verification is automatically completed and a proof outline is printed. We can check every intermediate condition in the proof outline using the `show` command. The verification condition of each example is compared with the given precondition.

Finally, a performance test of NQPV on larger qubit numbers is presented. For the Grover algorithm, NQPV takes several minutes and 32 GB memory to finish the verification of the 13 qubit case.

C.7 Experiment customisation

The examples in the notebook demonstrate the syntax of describing a verification task in NQPV.

To conduct a customised experiment, we can:

- (1) Encode the verification task in the `example.nqpv` file. It contains the program to be verified, the pre- and post-conditions, and the loop invariants.
- (2) Prepare a python script to produce the operators used in `example.nqpv` as binary `numpy` matrix files. Then invoke the verification using the command


```
nqpv.verify("example.nqpv")
```
- (3) Run the python script to conduct the verification.
- (4) After obtaining the proof outline, modify `example.nqpv` and repeat to print the conditions during verification.

To avoid path errors, the above experiments should be conducted in the root folder of NQPV source, or use NQPV installed from PyPI.

C.8 Notes

MacOS users may encounter an installation error due to the package `cmake`. In this case, try to run `pip install cmake` first, then add `cmake` to `PATH` manually, and finally install NQPV.

We recommend using `conda` to create a new experimental environment. The deployment was also tested on a Linux cloud server.

REFERENCES

- [1] Ralph-Johan Back and Joakim Wright. 2012. *Refinement calculus: a systematic introduction*. Springer Science & Business Media.
- [2] Yuxin Deng, Yuan Feng, and Ugo Dal Lago. 2015. On Coinduction and Quantum Lambda Calculi. In *26th International Conference on Concurrency Theory*. 427–440.
- [3] David Deutsch. 1985. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400, 1818 (1985), 97–117.
- [4] Ellie D’Hondt and Prakash Panangaden. 2006. Quantum weakest preconditions. *Mathematical Structures in Computer Science* 16, 3 (2006), 429–451.
- [5] Edsger Wybe Dijkstra, Edsger Wybe Dijkstra, Edsger Wybe Dijkstra, Etats-Unis Informaticien, and Edsger Wybe Dijkstra. 1976. *A discipline of programming*. Prentice-Hall Englewood Cliffs.
- [6] Yuan Feng, Runyao Duan, Zhengfeng Ji, and Mingsheng Ying. 2007. Proof rules for the correctness of quantum programs. *Theoretical Computer Science* 386, 1–2 (2007), 151–166.
- [7] Yuan Feng and Mingsheng Ying. 2021. Quantum Hoare logic with classical variables. *ACM Transactions on Quantum Computing* 2, 4 (2021), 1–43.
- [8] Jifeng He, Karen Seidel, and Annabelle McIver. 1997. Probabilistic models for the guarded command language. *Science of Computer Programming* 28, 2–3 (1997), 171–192.
- [9] CAR Hoare, He Jifeng, and Augusto Sampaio. 1993. Normal form approach to compiler design. *Acta informatica* 30, 8 (1993), 701–739.
- [10] Karl Kraus, Arno Böhm, John D Dollard, and WH Wootters. 1983. States, effects, and operations: fundamental notions of quantum theory. *Lecture notes in physics* 190 (1983).
- [11] Yangjia Li and Mingsheng Ying. 2017. Algorithmic analysis of termination problems for quantum programs. In *Proceedings of the ACM on Programming Languages*. ACM New York, NY, USA, 1–29.
- [12] Yangjia Li, Nengkun Yu, and Mingsheng Ying. 2014. Termination of nondeterministic quantum programs. *Acta informatica* 51, 1 (2014), 1–24.
- [13] Junyi Liu, Bohua Zhan, Shuling Wang, Shenggang Ying, Tao Liu, Yangjia Li, Mingsheng Ying, and Naijun Zhan. 2019. Formal Verification of Quantum Algorithms Using Quantum Hoare Logic. Springer, Cham, 187–207. https://doi.org/10.1007/978-3-030-25543-5_12
- [14] Annabelle McIver, Carroll Morgan, and Charles Carroll Morgan. 2005. *Abstraction, refinement and proof for probabilistic systems*. Springer Science & Business Media.
- [15] A K McIver and Carroll Morgan. 2001. Partial correctness for probabilistic demonic programs. *Theoretical Computer Science* 266, 1–2 (09 2001), 513 – 541. [https://doi.org/10.1016/s0304-3975\(00\)00208-5](https://doi.org/10.1016/s0304-3975(00)00208-5)
- [16] Graeme Mitchison and Richard Jozsa. 2001. Counterfactual computation. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 457, 2009 (2001), 1175–1193.
- [17] Carroll Morgan. 1993. The refinement calculus. In *Program Design Calculi*. Springer, 3–52.
- [18] Carroll Morgan, Annabelle McIver, and Karen Seidel. 1996. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 18, 3 (1996), 325–353.
- [19] Joseph M Morris. 1987. A theoretical basis for stepwise refinement and the programming calculus. *Science of Computer programming* 9, 3 (1987), 287–306.
- [20] Michael A Nielsen and Isaac Chuang. 2002. *Quantum computation and quantum information*. Cambridge University Press.
- [21] Michele Pagani, Peter Selinger, and Benoît Valiron. 2014. Applying quantitative semantics to higher-order quantum computing. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. 647–658.
- [22] Jeff W Sanders and Paolo Zuliani. 2000. Quantum programming. In *International Conference on Mathematics of Program Construction*. Springer, 80–99.
- [23] Peter Selinger. 2004. Towards a quantum programming language. *Mathematical Structures in Computer Science* 14, 4 (2004), 527–586.
- [24] Peter Selinger and Benoît Valiron. 2006. A lambda calculus for quantum computation with classical control. *Mathematical Structures in Computer Science* 16, 3 (2006), 527–552.
- [25] Dominique Unruh. 2019. Quantum relational Hoare logic. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 1–31. <https://doi.org/10.1145/3290346>
- [26] André Van Tonder. 2004. A lambda calculus for quantum computation. *SIAM J. Comput.* 33, 5 (2004), 1109–1135.
- [27] John Von Neumann. 1955. *Mathematical Foundations of Quantum Mechanics*. Princeton University Press, Princeton, NJ.
- [28] Mingsheng Ying. 2012. Floyd–Hoare logic for quantum programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 33, 6 (2012), 1–49.

[29] Li Zhou, Gilles Barthe, Pierre-Yves Strub, Junyi Liu, and Mingsheng Ying. 2022. *CoqQ: Foundational Verification of Quantum Programs*. arXiv. <https://doi.org/10.48550/arXiv.2207.11350>

[30] Paolo Zuliani. 2004. Non-deterministic quantum programming. In *Proceedings of the 2nd International Workshop on Quantum Programming Languages*. 179–195.

Received 2022-10-20; accepted 2023-01-19