On Incorrectness Logic for Quantum Programs

PENG YAN, University of Technology Sydney, Australia
HANRU JIANG*, Yanqi Lake Beijing Institute of Mathematical Sciences and Applications, China
NENGKUN YU†, University of Technology Sydney, Australia

Bug-catching is important for developing quantum programs. Motivated by the incorrectness logic for classical programs, we propose an incorrectness logic towards a logical foundation for static bug-catching in quantum programming. The validity of formulas in this logic is dual to that of quantum Hoare logics. We justify the formulation of validity by an intuitive explanation from a reachability point of view and a comparison against several alternative formulations. Compared with existing works focusing on dynamic analysis, our logic provides sound and complete arguments. We further demonstrate the usefulness of the logic by reasoning several examples, including Grover's search, quantum teleportation, and a repeat-until-success program. We also automate the reasoning procedure by a prototyped static analyzer built on top of the logic rules.

CCS Concepts: • **Theory of computation** → **Programming logic**; • **Computer systems organization** → **Quantum computing**.

Additional Key Words and Phrases: Incorrectness Logic, Quantum Programming Languages, Projective Quantum Predicates

ACM Reference Format:
Peng Yan, Hanru Jiang, and Nengkun Yu. 2022. On Incorrectness Logic for Quantum Programs. *Proc. ACM Program. Lang.* 6, OOPSLA1, Article 72 (April 2022), 28 pages. https://doi.org/10.1145/3527316

1 INTRODUCTION

Quantum computing hardware has made significant progress in the past decade [Arute et al. 2019; Zhong et al. 2020]. Great efforts have been devoted to developing programming languages and software to make it realistic to solve real-world problems using quantum computers. However, difficulty in writing correct quantum programs hinders practical quantum computing. Due to the counter-intuitive nature of quantum mechanics, small quantum programs written and reviewed by professional quantum computing experts are sometimes erroneous. For example, bugs arose in example programs of IBM's OpenQASM project [Cross et al. 2021], Qiskit [Aleksandrowicz et al. 2019], and Rigetti's PyQuil project [Smith et al. 2016] in their official GitHub repositories. Huang and Martonosi [Huang and Martonosi 2019a,b] proposed a taxonomy for bugs based on their debugging experience with Scaffold [Abhari et al. 2012; JavadiAbhari et al. 2015]. Theories and techniques for debugging are in urgent demand.

*The first two authors contributed equally.
†Corresponding author

Authors' addresses: Peng Yan, Centre for Quantum Software and Information, University of Technology Sydney, Sydney, Australia, pengyan.edu@gmail.com; Hanru Jiang, Yanqi Lake Beijing Institute of Mathematical Sciences and Applications, Beijing, China, hanru@bimsa.cn; Nengkun Yu, Centre for Quantum Software and Information, University of Technology Sydney, Sydney, Australia, nengkunyu@gmail.com.

2475-1421/2022/4-ART72
https://doi.org/10.1145/3527316

There are two lines of approaches aiming at debugging quantum programs. One approach is dynamic assertions [Li et al. 2020; Liu et al. 2020], which detect erroneous states via quantum measurements at the cost of additional qubits and quantum operations at run-time. The other is statistical assertions [Huang and Martonosi 2019a,b] that detect errors via statistical tests over sampled simulations. Unfortunately, both approaches suffer from two main limitations:

(1) *Limited support for bug-finding ahead of run-time.* It is desirable to debug a quantum program before submitting it to a quantum device, which might be busy and make the program have to wait in the queue before being executed. The dynamic assertions are designed for run-time debugging instead. Statistical assertions achieve static debugging via repeated simulated measurements, which is inefficient, as argued in [Li et al. 2020].
(2) *Lack of soundness or completeness arguments.* Though these approaches should alarm only true bugs, none of them makes formal arguments about their soundness. Even without complicated control structures like while-loops, none of them guarantees completeness (do not miss bug): both dynamic assertions and statistical assertions capture a bug by chance due to the probabilistic nature of quantum measurement.

To address these limitations, program logics like quantum Hoare logic [Ying 2012] and applied quantum Hoare logic (aQHL) [Zhou et al. 2019] seem to be a good choice, since they facilitate static reasoning and provide soundness and completeness guarantees. However, these logics are not suitable for debugging purposes. They are not known to be decidable, and their proof rules do not prove the existence of a bug: propositions of the form $\neg(\{P\}S\{Q\})$. Negating the postcondition will not help much, because in general,

$$\neg(\{P\}S\{Q\}) \nRightarrow \{P\}S\{\neg Q\},$$

which means true bugs could be missed.

One work that addressed the above issues in the classical world is the incorrectness logic [O'Hearn 2019] (IL), an under-approximate analogy of Hoare logic for reasoning about bugs. Specifications in IL are of the form

$$[\textit{presumption}]\,\textit{code}\,[\textit{result}].$$

It says that the post-assertion *result* is an under-approximation (subset) of the final states obtained by executing the *code* from states in *presumption*. It can be equivalently interpreted as: every state in *result* is reachable from some state in *presumption*. When *result* specifies the erroneous states, such an interpretation matches the principle of debugging tools to avoid false positives (bug suggestions that are not true). Guided by such a principle, static debugging tools [Blackshear et al. 2018; Distefano et al. 2019; Gorogiannis et al. 2019] were developed and proved practical, making it easier for programmers to locate and fix the bugs. This novel idea was also advanced to Incorrectness Separation Logic [Raad et al. 2020], which derived a begin-anywhere, intra-procedural symbolic execution analysis with no false positives.

A similar theory for quantum programming would benefit quantum software development and guide the design and implementation of debugging tools. However, it is unclear how to generalize IL to the quantum settings, where the state model and the predicates are fundamentally different. In particular, it is not clear how to use quantum predicates to characterize errors and what it means by achieving (reaching everything described by) a quantum predicate.

In this paper, we extend the idea of IL by using projection-based quantum predicates from the quantum logic [Birkhoff and Neumann 1936], which has been successfully applied to reasoning about the correctness [Zhou et al. 2019] of quantum programs and designing dynamic assertions

[Li et al. 2020]. The main result is a sound and complete logic system for reasoning about bugs in quantum programs [1] statically. Technical contributions include:

- A novel interpretation of projection-based quantum predicates in the context of bug-catching. The key ingredient is an under-approximation relation, which is the inverted satisfaction relation for projections. We explain why the satisfaction of projections is not suitable for characterizing erroneous quantum states and why our under-approximation relation can capture errors without introducing false positives.
- An incorrectness triple based on the under-approximation relation to incorporate the spirit of reachability analysis proposed by O'Hearn [O'Hearn 2019]. The triple turns out to be an under-approximate dual of the aQHL triple. To better understand and justify our triple, we compare it with several possible alternatives in Sec 8 and find our triple the best in expressiveness and efficiency.
- A sound and complete quantum incorrectness logic (QIL) based on the incorrectness triple. The resulting proof rules in our logic have a similar structure to their classical counterparts. We further prove that bounded loop-unrolling is sufficient to guarantee completeness when the quantum system is finite-dimensional, ensuring that a complete inference is decidable even if the state space is uncountably infinite. This result also shows that aQHL is decidable.
- Three examples for demonstrating the incorrectness reasoning by QIL, namely Grover's algorithm, quantum teleportation, and a repeat-until-success program. In these examples, we introduce and reason about two types of bugs mentioned in Huang et al. [Huang and Martonosi 2019b]. We also developed a prototyped static analyzer built on top of our proof rules to automate the reasoning.

Organization of the paper. Sec. 2 gives the minimal background of quantum computation. Sec. 3 explains the main challenges and develops our key ideas. The language used in this paper is given in Sec. 4. In Sec. 5, we introduce the under-approximation relation, a quantum version of incorrectness triple, and the duality between quantum correctness and incorrectness triples. In Sec. 6, a sound and complete proof system for QIL triples is presented. In Sec. 7, we give three examples for reasoning about the existence of bugs with our proof system. In Sec. 8, we discuss the reasons for our choice by comparing our triple with other alternatives. The related works and conclusion are given in Section 9 and Section 10 respectively. We leave detailed proofs in TR.[Yan et al. 2022]

2 PRELIMINARY

This section presents a minimal background of quantum computation and projection-based quantum predicates to make the paper self-contained. We will walk through the basics of quantum computing with a small program over a 2-qubit system, as shown in Fig. 1a with the corresponding state transitions illustrated in Fig. 1b. The program prepares a Bell state then measures the two qubits, and the two measurement outcomes from the two qubits are expected to be the same.

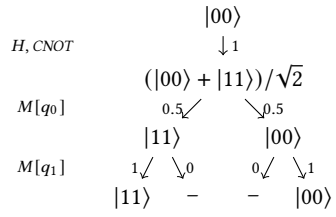
2.1 Quantum States

We consider a quantum system of the form $\bar{q}$ consisting of n *qubits* (quantum bits). The state space of the system is the 2^n-dimensional complex vector space $\mathbb{C}^{2^n}$. We use $\mathcal{H}_{\bar{q}}$ to denote this space and omit $\bar{q}$ when it is evident from context or irrelevant. Unit vectors in $\mathcal{H}$ are denoted by $|\psi\rangle$ following Dirac's notation.

[1] We consider only quantum programs with classical control, and bugs at the software level. We expect software level bug-catching to be important for both near-term and error-corrected quantum computing, because we are not aware of any quantum algorithm robust to logical bugs (instead of noise that arises in hardware).

```
// assume span{|00⟩}
H(q0); CNOT(q0, q1);
if ( M[q0] = true → skip
        □ false → skip ) fi;
if ( M[q1] = true → skip
        □ false → skip ) fi;
// ensures span{|00⟩, |11⟩}
```

(a) the program, where $M = \{|0\rangle\langle 0|, |1\rangle\langle 1|\}$



(b) the transitions of quantum states, labels on arrows are (conditional) probabilities of the transition

Fig. 1. A small example that prepares and measures a Bell state.

A state of the quantum system is generally a probabilistic mixture of unit vectors in $\mathcal{H}$, i.e., a statistical ensemble $\{(p_i, |\psi_i\rangle)\}$ where $p_i \in (0, 1]$ and $\sum_i p_i = 1$. A statistical ensemble is called a *pure state* or *vector state* if it contains only one unit vector (i.e., $\{(1, |\psi\rangle)\}$), otherwise is called a *mixed state*. We denote a pure state $\{(1, |\psi\rangle)\}$ by $|\psi\rangle$ for short.

The quantum states obtained along the example of Fig. 1a are all pure states. In the beginning, both qubits q_0 and q_1 are in the state $|0\rangle = \binom{1}{0}$, the state encoding a classical bit of value 0. The initial state of the entire system is thus a pure state $|0\rangle_{q_0} \otimes |0\rangle_{q_1}$ (written as $|00\rangle$ for short), where $\otimes$ is the tensor product.

A program modifies the quantum state via two kinds of operations, namely unitary operations and measurements. In our example, both H and $CNOT$ are unitary operations, and we also call them *gates*. We skip the details of how these gates change the state for now, but depict the state transition in Fig. 1b by the arrow in the same row with H, $CNOT$. After performing these two gates, we obtain the Bell state $(|00\rangle + |11\rangle)/\sqrt{2}$ with probability 1, where $|1\rangle = \binom{0}{1}$ encodes a classical bit of value 1. Note that the Bell state is pure, a superposition of unit vectors $|00\rangle$ and $|11\rangle$, which encodes the two qubits having the same classical bit-value.

A more general way to represent a quantum state $\{(p_i, |\psi_i\rangle)\}$ is to use a *density matrix* $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$. For example, the density matrix of the pure state $|0\rangle$ is simply $|0\rangle\langle 0| = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$, and the density matrix of the mixed state $\{(\frac{1}{2}, |0\rangle), (\frac{1}{2}, |1\rangle)\}$ is $\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \frac{I}{2}$. One may notice that there might be multiple ensembles that have the same density matrix representation. In this case, we do not distinguish these quantum states because they are not physically distinguishable.

2.2 Quantum Operations

A unitary operation over a quantum system $\bar{q}$ is encoded as a unitary matrix of dimension $2^{|\bar{q}|}$, that is, a matrix U that satisfies $U^\dagger U = UU^\dagger = I$. When applied to a quantum system containing $\bar{q}$, the unitary operation changes a quantum state ρ into $U_{\bar{q}} \rho U_{\bar{q}}^\dagger$, where $U_{\bar{q}}$ is the unitary operation over the entire system which effectively applies U over $\bar{q}$, and leaves qubits other than $\bar{q}$ untouched. For example, if we apply U to the q-th qubit in an n-qubit system, we have

$$U_{\bar{q}} = \otimes_{1 \le i < q} I \otimes U \otimes_{q+1 \le j \le n} I.$$

We often omit the subscript $\bar{q}$ in $U_{\bar{q}}$ when it is clear from the context.

Commonly used single-qubit operators include the *Hadamard* operator H, and the Pauli operators X, Y, and Z. Another commonly used operator is the controlled-NOT operator *CNOT*. The matrices for these operations are

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

Behavior of these operators can also be described by its effect on *computational basis* $\{|0\rangle, |1\rangle\}$. For Pauli operators we have $X|0\rangle = |1\rangle$, $X|1\rangle = |0\rangle$, $Y|0\rangle = i|1\rangle$, $Y|1\rangle = -i|0\rangle$, $Z|0\rangle = |0\rangle$, $Z|1\rangle = -|1\rangle$. For the Hadamard operator, we have $H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and $H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.[2] The $CNOT(q_1, q_2)$ operator takes qubit q_1 as the control qubit and applies X (logic NOT) operator to qubit q_2 if qubit q_1 is in the state $|1\rangle$, that is, $|b_1\rangle_{q_1} \otimes |b_2\rangle_{q_2} \longrightarrow |b_1\rangle_{q_1} \otimes |b_1 \oplus b_2\rangle_{q_2}$, where $\oplus$ is the logical XOR (exclusive or) operation. For example, we have transitions $|00\rangle \longrightarrow |00\rangle$ and $|10\rangle \longrightarrow |11\rangle$ for *CNOT* gates.

2.3 Quantum Measurements

Programs read information from a quantum system via *quantum measurement*s, which is the source of probabilistic non-determinism during the execution of a quantum program. A measurement is described by a set M of linear operators on $\mathcal{H}$, such that

$$M = \{M_m\} \text{ with } \textstyle\sum_m M_m^\dagger M_m = I_{\mathcal{H}},$$

where $I_{\mathcal{H}}$ is the identity operator on $\mathcal{H}$, and $M_m^\dagger$ is the conjugate transpose of M_m. The subscript m stands for the measurement outcome. Given a pure state $|\psi\rangle$, after applying a measurement $M = \{M_m\}$, the outcome m may be observed with probability $p_m = \langle\psi|M_m M_m^\dagger|\psi\rangle$, the state after the measurement with outcome m collapses into $M_m|\psi\rangle / \sqrt{p_m}$ when $p_m \neq 0$. Here $\langle\psi| = |\psi\rangle^\dagger$ is the conjugate transpose of $|\psi\rangle$.

In the example of Fig. 1, measurement serves as the guard of a branching statement: after the measurement, the quantum program jumps to the branch corresponding to the outcome of the measurement. The two **if**-statements measure the two qubits using the measurement $M = \{|0\rangle\langle 0|, |1\rangle\langle 1|\}$[3], and simply **skip** according to the outcomes. After the first measurement, the program state collapses into $|00\rangle$ or $|11\rangle$ with equal probability. Only one branch is possible for the second measurement, and the program state does not change after the measurement.

An *orthogonal projection* is a linear operator P on $\mathcal{H}$ that satisfies $P^2 = P = P^\dagger$, we call it a *projection* for short. A projective measurement is a special kind of measurement described by a set of projections $\{P_m\}$, that is

$$M = \{P_m\} \text{ with } \textstyle\sum_m P_m = I \text{ and } P_m P_n = \begin{cases} P_m & \text{if } m = n \\ \mathbf{0} & \text{otherwise} \end{cases}$$

In particular, the measurements in Fig. 1 are projective. A critical property of a projective measurement $M = \{P_m\}$ is, when a quantum state is a mixture of unit vectors in P_m for some m, measuring the state with M will return the outcome m for sure, and leave the state unchanged, just like the case of the second measurement in Fig. 1a.

[2] Apart from $\{|0\rangle, |1\rangle\}$, $\{|+\rangle, |-\rangle\}$ is another widely used basis for one-qubit system. These two basis can be transformed to each other by applying the Hadamard operator.

[3] when M is applied on q_0 and leaving q_1 untouched, the corresponding measurement operator over the entire system is $\{|0\rangle\langle 0| \otimes I, |1\rangle\langle 1| \otimes I\}$.

One-to-one correspondence between a projection and a linear subspace. We do not distinguish between a projection and its corresponding subspace. Given the eigen decomposition of a projection $P = \sum_i |p_i\rangle\langle p_i|$, its corresponding subspace is the space spanned by $\{|p_i\rangle\}$. On the contrary, we can construct a projection $P = \sum_i |\psi_i\rangle\langle\psi_i|$ for an arbitrary subspace with its complete orthogonal basis $\{|\psi_i\rangle\}$. For example, give a projection $P = |+\rangle\langle+| = \frac{1}{2}\begin{pmatrix}1&1\\1&1\end{pmatrix}$, it corresponds to one-dimensional subspace spanned by $\{|+\rangle\}$. For the entire subspace of the one-qubit system, we choose the basis $\{|0\rangle, |1\rangle\}$ and then get its corresponding projection $P = |0\rangle\langle 0| + |1\rangle\langle 1| = I$ [4].

Fig. 2 further illustrates how a projection P takes effects on a quantum state ρ. Projection P maps the state ρ into $P\rho P$ that lies in the subspace P; projection $P^\perp$ [5] maps the state ρ into $P^\perp\rho P^\perp$ that lies in the subspace $P^\perp$. The mapping works similarly to the decomposition of Euclidean vectors where we have $\mathrm{Tr}(P\rho P) + \mathrm{Tr}(P^\perp\rho P^\perp) = \mathrm{Tr}(\rho)$. The states in subspace P and $P^\perp$ are orthogonal to each other. For example, projection $P = |0\rangle\langle 0|$ maps state $|+\rangle\langle+|$ into state $|0\rangle\langle 0|/2$, and its complement $P^\perp = |1\rangle\langle 1|$ maps $|+\rangle\langle+|$ into state $|1\rangle\langle 1|/2$.

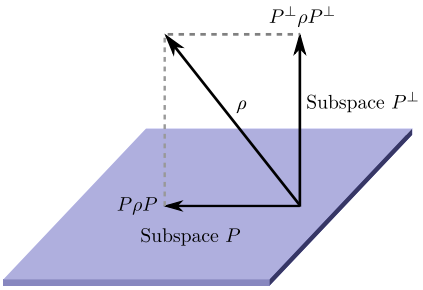


Fig. 2. Projection P on state ρ.

2.4 Projective Quantum Predicates

There are typically two kinds of quantum predicates in the literature. The one we currently do not study is by D'Hondt and Panangaden [D'hondt and Panangaden 2006], where a quantum predicate is defined as a Hermitian operator with some constraints. Such predicates are less intuitive from a classical point of view, because they discuss the expectation (ranging from the interval $[0, 1]$) instead of a yes/no answer about some quantum state satisfying the predicate.

In this paper, we follow the other class of quantum predicates proposed by [Birkhoff and Neumann 1936], that is, projections. Projections can be viewed as a trade-off between expressiveness and practicability. As shown in applied quantum Hoare logic [Zhou et al. 2019] and run-time assertions [Li et al. 2020], projections are sufficient to express important properties, and easy to work with because of its compact formalism of assertions. The key benefits of using projective predicates are:

- Projections are easy to implement using existing quantum devices, thus inserting projections as assertions is logically meaningful and implementable for the dynamic checking.
- Satisfaction of a projection is a boolean function, which coincides with classical predicates, making it easier to incorporate the idea of IL.
- Projection-based run-time assertions (projective measurements) do not have side effects on quantum states satisfying the predicates.

[4] If you choose another basis $\{|+\rangle, |-\rangle\}$, you will still get the same $P = |+\rangle\langle+| + |-\rangle\langle-| = I$

[5] $P^\perp$ is the orthogonal complement of P, i.e. $P^\perp = I - P$.

Although projections can not model all types of bugs, they have a relatively high performance in capturing certain types of bugs that enlarge/shrink the subspaces of correct states, as shown in the example for Grover's search in Sec. 7.

Before discussing about projective quantum predicates formally, we introduce the support of positive semi-definite matrices (including density matrices) in Def. 2.1.

Definition 2.1 (Support). *If $A = \sum_i \lambda_i |\psi_i\rangle\langle\psi_i|$, where $|\psi_i\rangle$s are unit vectors in $\mathcal{H}$ and $\lambda_i > 0$, then the* support *of A is the subspace spanned by $\{|\psi_i\rangle\}$. I.e.,* $\mathrm{supp}(A) = \mathrm{span}\{|\psi_i\rangle\}$.

In particular, the support of a projection P is its corresponding subspace, so we write P as a shorthand of the subspace $\mathrm{supp}(P)$ directly. Besides, we also use the inclusion binary relation $\subseteq$ to denote the partial order on the set of subspaces, and $\in$ to represent the membership of subspaces.

Definition 2.2 (Satisfaction). *A quantum state ρ satisfies a projection P, denoted by $\rho \vDash P$, if* $\mathrm{supp}(\rho) \subseteq P$. *Contrarily, $\rho \nvDash P$ if* $\mathrm{supp}(\rho) \nsubseteq P$.

Formally, when using projections as predicates, a quantum state $\rho = \sum p_i |\psi_i\rangle\langle\psi_i|$ satisfying a projection P means $|\psi_i\rangle \in P$ for all i, i.e. $\mathrm{supp}(\rho) \subseteq P$. For example, to assert that the initial state in Fig. 1a encodes a classical bit array 00, we use projection $|00\rangle\langle 00|$ which corresponds to the space $\mathrm{span}\{|00\rangle\}$; to assert that the final state is a mixture of states having the same classical value in both qubits, we use the projection $P_= ::= |00\rangle\langle 00| + |11\rangle\langle 11|$ which corresponds to the space $\mathrm{span}\{|00\rangle, |11\rangle\}$. The assertion at the end of the program in Fig. 1a holds because the possible final states are exactly $|00\rangle$ and $|11\rangle$.

The largest projection is the identity operator I, corresponding to the entire state space $\mathcal{H}$. The smallest projection is the $\mathbf{0}$-operator, instead of the empty set. Any other projection P on $\mathcal{H}$ has $\mathbf{0} \subseteq P \subseteq I$ when interpreted as subspaces.

Logical operations on projections are different from their classical counterparts. We list the definitions of $\neg$, $\wedge$, and $\vee$ in Def. 2.3, where we use projections to denote both the quantum predicates and their corresponding subspaces.

Definition 2.3. *The logical operations for quantum predicates are defined as follows. For any two quantum predicates P, Q on $\mathcal{H}$,*

$$\neg P ::= P^\perp, \qquad P \wedge Q ::= P \cap Q, \qquad P \vee Q ::= \mathrm{span}(P \cup Q) = \neg(\neg P \wedge \neg Q),$$

where $P^\perp ::= I - P$ is the orthogonal complement of P, and $P^\perp$ is also a subspace.

The main difference from classical predicates lies in the negation: instead of set complement as in classical logic, the negation of a projection P is its orthogonal complement $P^\perp$, which is the projection $I - P$. Here the binary operator $-$ between predicates subtracts linear operators instead of sets. This difference leads to different meanings of disjunction operation: the disjunction of projections P and Q is the subspace spanned by all vectors in P and Q, not merely the union of these two subspaces.[6]

In summary, we list the symbols and notations in Table 1 for reference.

3 CHALLENGES AND OUR KEY IDEAS

We wish to extend IL to the quantum settings. IL triple [*presumption*] *code* [*result*] is interpreted as

every state in the result is reachable from some state in the presumption [O'Hearn 2019]. (1)

To apply this idea to the quantum settings, we need to answer:

- How to characterize an erroneous quantum state using a projection?

[6] The intersection of two subspaces still forms a subspace, but not for the union operation.

Table 1. Symbols and notations.

Symbol	Meanings
$\|\psi\rangle$	Dirac notation of vector state, equal to matrix representation $\|\psi\rangle\langle\psi\|$.
$\rho = \sum p_i \|\psi_i\rangle\langle\psi_i\|$	(partial) density matrix, a statistical ensemble $\{(p_i, \|\psi\rangle)\}$ with $\sum p_i \leq 1$.
$\|\psi\rangle_{q_1} \otimes \|\varphi\rangle_{q_2}$	product state of the composite system consisting of qubits q_1 and q_2.
$\rho \to U\rho U^\dagger$	unitary operation U on ρ.
$\rho \to \frac{M_m \rho M_m^\dagger}{\mathrm{Tr}(M_m^\dagger M_m \rho)}$	quantum measurement $M = \{M_m\}$ on ρ with measurement outcome m.
$\mathrm{supp}(A)$	subspace spanned by the eigenvectors of matrix A. $\mathrm{supp}(P)$ is written as P for short if P is a projection.
$\rho \to P\rho P$	projection P maps the state ρ into $P\rho P$ lying in the subspace $\mathrm{supp}(P)$.
$\|\psi\rangle \in P$	vector state $\|\psi\rangle$ lies in the subspace P.
$\rho \vDash P$	satisfaction relation, subspace $\mathrm{supp}(\rho)$ lies in the subspace P.
$P_1 \subseteq P_2$	subspace P_1 lies in the subspace P_2.
$\neg P$	the orthogonal complement of subspace P, i.e. $P^\perp$.
$P \wedge Q$	the subspace spanned by vectors in both P and Q, i.e. $P \cap Q$.
$P \vee Q$	the subspace spanned by vectors in either P or Q, i.e. $\mathrm{span}(P \cup Q)$.

- What does it mean by reaching everything in (that is, achieving) the *result* predicate in quantum settings?
- With an incorrectness logic built upon the answers to the previous two questions, can we reason about quantum programs automatically?

The following three subsections explain the challenges that arise in answering these three questions correspondingly, and our ideas for addressing them.

3.1 An Obstacle of Characterizing Errors: Satisfaction of a Projection Is Not Precise

Given an assertion P like the one at the bottom of Fig. 1a, we first need to answer how to *precisely* characterize an erroneous state $\rho \nvDash P$. Unfortunately, projections and satisfaction are unable to capture some erroneous states without introducing false positives.

For example, let P_c be the assertion $|0\rangle\langle 0|$, and let $\rho_e = \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|)$ be an erroneous state which is a probabilistic mixture of a good state $|0\rangle\langle 0| \vDash P_c$ and a bad state $|1\rangle\langle 1| \nvDash P_c$. Here we use subscripts c and e to distinguish between correct and erroneous states or the corresponding assertions. If we use satisfaction to specify the erroneous state ρ_e, we need to find a projection Q_e such that $\rho_e \vDash Q_e$. If such Q_e exists, it means $|0\rangle\langle 0| \subseteq \mathrm{supp}(\rho_e) \subseteq Q_e$. That is, a good state $|0\rangle\langle 0| \vDash P_c$ also satisfies Q_e, which is a false positive.

The problem is that the support of an erroneous state $\rho_e \nvDash P$ is not necessarily orthogonal to P, thus characterizing ρ_e using satisfaction and projection may falsely capture good states. Classical IL does not have this problem, because any state $\sigma_e \notin p$ can be precisely characterized by $\{\sigma_e\} \subseteq \neg p$.

Our idea. We replace satisfaction relation with an under-approximation relation for characterizing errors. We say ρ is *under-approximated* by P, denoted by $\rho \dashv P$, if $\mathrm{supp}(\rho) \supseteq P$. The under-approximation relation is the inverted satisfaction. Intuitively, it means that ρ can be a mixture of states that contains $|\psi_i\rangle$ described by $P = \sum |\psi_i\rangle\langle\psi_i|$, and vector states in P are the "100%" errors. With this relation, we can characterize $\rho_e = \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|)$ by $\rho_e \dashv |1\rangle\langle 1|$ without introducing false positives.

3.2 An Obstacle of Interpreting Achieving: Impossibility to Reach Every State Described by a Projection

In IL, the triple requires *result* to be *achieved*, which means every state $\sigma \vDash$ *result* can be obtained after an execution. For quantum programs, directly adopting this idea by replacing $\vDash$ with $\Subset$ is not reasonable, because it is commonly impossible to reach every ρ in the set $\{\rho \mid \rho \Subset P\}$.

For example, consider the program measuring a single qubit q with $M = \{|0\rangle\langle 0|, |1\rangle\langle 1|\}$:

$$\textbf{if } (M[q] = \textbf{true} \to \textbf{skip} \square \textbf{false} \to \textbf{skip}) \textbf{ fi } \textit{// achieve } |0\rangle\langle 0|$$

The program has at most 2 possible output states $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$ for any input state, thus is unable to "achieve" a reasonable projection $|0\rangle\langle 0|$, which under-approximates an infinite set of states, e.g., $\{\lambda\,|0\rangle\langle 0| + (1-\lambda)\,|1\rangle\langle 1| \mid \lambda \in (0,1]\}$. How should we interpret "achieving" a projection in the quantum settings to make the reachability analysis meaningful?

Our idea. We interpret achieving P as: P under-approximates the probabilistic *mixture* of reachable states. This interpretation is reasonable in the sense that, if P is achieved, then any pure state $|\psi\rangle \in P$ can be obtained by measuring the final state of some execution path, using the measurement $\{|\psi\rangle\langle\psi|, I - |\psi\rangle\langle\psi|\}$.

With such an interpretation, we avoid reaching infinitely many states to achieve a projection. In the above example, $|0\rangle\langle 0|$ is the only non-trivial projection that under-approximates the possible output state $|0\rangle$. We achieve the projection $|0\rangle\langle 0|$ whenever it is possible to obtain the state $|0\rangle$ via the **false** branch.

The semantics of a QIL triple $[P]S[Q]$ follows directly from this interpretation of "achieving":

If a state achieves P, the mixture of its reachable states after executing S achieves Q.

It can be equivalently described as $post(S)P \supseteq Q$, which is dual to the correctness triple $post(S)P \subseteq Q$ in aQHL. Here *post* is the largest achievable postcondition defined formally in Sec. 5.3.

We list the key ingredients of IL and QIL discussed above in Table 2 for comparison.

Table 2. Comparison of the key ingredients in IL and QIL. Here σ and p are classical state and predicate, ρ and P are quantum state and projection.

Key Ingredients	IL	QIL
Assertion	p (set of states)	P (linear subspace)
σ or ρ is erroneous	$\sigma \in \neg p$	$\rho \Subset Q$ for some $Q \not\subseteq P$
p or P is achieved	$(\bigcup_{\sigma \text{ reachable}} \{\sigma\}) \supseteq p$	$\left(\sum_{\rho \text{ reachable}} \rho\right) \Subset P$

3.3 An Obstacle of Automation: Bounding Iteration of Loops May Sacrifice Completeness

Based on the QIL triple above, we derive a set of proof rules that is sound and complete. One remaining question is how to automate the reasoning based on the proof rules. The main problem lies in the While-rule below: inferring the backward variant P_n requires infinite loop unrolling.

$$\frac{\forall n. \vdash [\mathrm{supp}(M_1 P_n M_1^\dagger)] S [P_{n+1}]}{\vdash [P_0] \textbf{while } M[\bar{q}] = 1 \textbf{ do } S \textbf{ od} [\mathrm{supp}(M_0 P_N M_0^\dagger)]}$$

IL solves this problem by fixing a bound for loop unrolling at the expense of sacrificing completeness when the set of states is infinite. For QIL, the same sacrifice seems inevitable because the state

space is an infinite set, even for a finite-dimensional quantum system. We are curious whether this solution applies to the quantum settings and what has to be paid for bounding loop unrolling.

Our idea. We observe that for a finite-dimensional state space, the projection $\vee_{i\le N}\,\mathrm{supp}(M_0 P_i M_0^\dagger)$ does not change when N is larger than the dimension of state space $\mathcal{H}$. This enables us to replace the While-rule with a bounded version below, while retaining completeness argument.

$$\frac{\forall n < \dim(\mathcal{H}).\ \vdash [\mathrm{supp}(M_1 P_n M_1^\dagger)]S[P_{n+1}]}{\vdash [P_0]\textbf{while } M[\bar{q}] = 1 \textbf{ do } S \textbf{ od}[\vee_{i\le N}\,\mathrm{supp}(M_0 P_i M_0^\dagger)]}$$

The reason behind this observation is that $\vee_{i\le N}\,\mathrm{supp}(M_0 P_i M_0^\dagger)$ is increasing w.r.t. relation $\subseteq$ as N grows, while its rank is bounded by the dimension of state space $\mathcal{H}$. The bounded version of the While rules enables computing $post(S)P$ within finite steps effectively. Recall that validity of a QIL triple and aQHL triple can be decided by comparing $post(S)P$ with Q. The bounded While rules not only make our logic decidable but also enable automatic verification of an aQHL triple.

4 THE EXTENDED QUANTUM WHILE LANGUAGE

In this section, we introduce our quantum programming language with classical controls, which extends the quantum **while** language [Perdrix 2008a,b; Ying 2012] by adding the **error** statement to capture errors and encode **assert** statements.

4.1 Syntax

The syntax of our program language is defined in Fig. 3. We use q to represent the identity of a qubit and $\bar{q}$ to represent a quantum register, that is, a list of different qubits. The **skip** statement explains itself. Two statements S_1 and S_2 can be sequenced by $S_1; S_2$. Below we explain the other statements.

$$\begin{array}{rrcl} (Stmts) & S & ::= & \textbf{skip} \mid S_1; S_2 \mid q := |0\rangle \mid \bar{q} := U\bar{q} \mid \textbf{if } (\square m \cdot M[\bar{q}] = m \to S_m) \textbf{ fi} \\ & & & \mid \textbf{while } M[\bar{q}] = 1 \textbf{ do } S \textbf{ od} \mid \textbf{error} \\ \\ \textbf{assert}(\bar{q}, P) & & ::= & \textbf{if}(M_P[\bar{q}] = \textbf{true} \to \textbf{skip} \,\square\, \textbf{false} \to \textbf{error})\textbf{fi} \\ & & & \text{where } M_P = \{M_{\textbf{true}}, M_{\textbf{false}}\}, M_{\textbf{true}} = P, M_{\textbf{false}} = I - P. \end{array}$$

Fig. 3. The syntax of the extended quantum while language (above); encoding the assert statement (below).

The statement $q := |0\rangle$ initializes a qubit q to state $|0\rangle$. Formally, initializing a qubit q in a quantum system would change the state ρ into $|0\rangle_q\langle 0|\rho|0\rangle_q\langle 0| + |0\rangle_q\langle 1|\rho|1\rangle_q\langle 0|$, where $|n\rangle_q\langle m|$ is the operator $|n\rangle\langle m|$ on qubit q. The statement $\bar{q} := U\bar{q}$ applies a unitary transformation U on the quantum register $\bar{q}$ and changes the state ρ into $U\rho U^\dagger$. The case statement **if** $(\square m \cdot M[\bar{q}] = m \to S_m)$ **fi** performs a quantum measurement M on the register $\bar{q}$, and executes subprogram S_m according to the measurement outcome m. Loop statement **while** $M[\bar{q}] = 1$ **do** S **od** performs a yes-no measurement with two possible outcomes 0 and 1, then terminates or executes S and reenters the loop correspondingly. Recall that measurement has side effects on quantum states; the branches/loop bodies will start with a collapsed state after measurement.

The newly introduced **error** statement halts the execution and signals an error. One of the main applications of **error** is to encode projection-based assertions [Li et al. 2020] for quantum programs, to test whether a property holds at a particular program point.

An assertion in classical programming languages typically tests whether a predicate (a boolean-valued function over program states) holds at a particular program point. If the test succeeds, the execution continues, otherwise the program terminates abnormally and may throw an exception.

For quantum programs, such tests on quantum states are not generally feasible because i) the only way to observe a quantum state is by measuring the state, and ii) measurement has side effects over quantum states in general.

To achieve a quantum counterpart of an assertion, we restrict the predicate to be a projective measurement, following the approach of [Li et al. 2020]. Concretely, we encode the $\mathbf{assert}(\bar{q}, P)$ statement at the bottom of Fig. 3, where P is a projection over space $\mathcal{H}_{\bar{q}}$, indicating a certain property over $\bar{q}$. Projective measurement is suitable for encoding assertions because for a state ρ,

- if $\mathrm{supp}(\rho) \subseteq P$, the outcome of $M_P[\bar{q}]$ over ρ will be **true** for sure, and ρ keeps unchanged after measurement, thus assertions will not affect future executions,
- otherwise, there is a non-zero probability that the outcome of $M_P[\bar{q}]$ is **false**, and an error would arise.

An abnormal termination caused by $\mathbf{assert}(\bar{q}, P)$ can then be viewed as evidence of a bug. Intuitively, it means some part in ρ lies out of P.

4.2 Semantics

The semantics of the extended quantum **while** programs is standard, except for the treatment of the newly introduced **error** statement. To distinguish abnormal terminations caused by **error** from those normal terminations, we adopt the exit condition ϵ from the incorrectness logic [O'Hearn 2019].

$$(\mathit{ExitCond})\ \epsilon\ ::= \mathit{ok} \mid \mathit{er}$$

The value of an exit condition ϵ can be either *ok* or *er*. Here *ok* is for normal terminations and *er* is for abnormal terminations caused by **error** statements. We call the output of normal/abnormal terminations as normal/abnormal states, respectively.

We assume that any quantum program corresponds to a fixed qubit system, and free variables in the program are within the corresponding state space $\mathcal{H}$ of the system. Program states are defined as *partial density matrices* following Selinger's normalization convention [Selinger 2004]. A partial density matrix ρ guarantees only $\mathrm{Tr}(\rho) \leq 1$ instead of $\mathrm{Tr}(\rho) = 1$, which allows us to absorb the probability of reaching a state as a scalar factor. Formally, the set of partial density matrices over $\mathcal{H}$ (denoted by $\mathcal{D}^-(\mathcal{H})$) is defined below, as we use ρ for elements in $\mathcal{D}^-(\mathcal{H})$. In particular, $\mathbf{0}$ is a partial density matrix representing an impossible state.

$$\mathcal{D}^-(\mathcal{H}) = \{\textstyle\sum_i p_i |\psi_i\rangle\langle\psi_i| \mid p_i \geq 0 \wedge \sum_i p_i \leq 1 \wedge |\psi_i\rangle \in \mathcal{H}\}$$

4.2.1 Operational Semantics. The operational semantics of our language is formalized in Fig. 4. We model the operational semantics by labelled transitions over program *configuration*s of the form $\langle S, \rho\rangle$, where S is the remaining code to be executed, and ρ is the current program state. The transition relation $\rightarrow$ is a ternary relation of type

$$(\mathit{Stmt} \times \mathcal{D}^-(\mathcal{H})) \times \mathit{ExitCond} \times ((\mathit{Stmt} \cup \{\downarrow\}) \times \mathcal{D}^-(\mathcal{H})),$$

where $\downarrow$ is used to denote the termination of a program by convention. A transition is denoted by $\langle S, \rho\rangle \xrightarrow{\epsilon} \langle S', \rho'\rangle$, and the label ϵ ranges from $\{ok, er\}$ to indicate a normal/abnormal transition.

Transitions labelled by *ok* in Fig. 4 are essentially the same as the standard operational semantics in the quantum Hoare logic [Ying 2012]. We explain the two transitions with the *er* label. The transition rule for **error** is intuitive. It simply terminates the execution, raises an *er* label, and returns the quantum state. The transition rule for $S_1; S_2$ with the *er* label says that when the execution of S_1 encounters an **error**, then the execution of the entire program immediately terminates abnormally, discarding the remaining code including S_2. To describe multiple-step executions where only the

$$\langle \mathbf{skip}, \rho\rangle \xrightarrow{ok} \langle\downarrow, \rho\rangle \qquad \langle \mathbf{error}, \rho\rangle \xrightarrow{er} \langle\downarrow, \rho\rangle \qquad \langle \bar{q} := U\bar{q}, \rho\rangle \xrightarrow{ok} \langle\downarrow, U\rho U^\dagger\rangle$$

$$\langle q := |0\rangle, \rho\rangle \xrightarrow{ok} \langle\downarrow, \textstyle\sum_n |0\rangle_q\langle n|\rho|n\rangle_q\langle 0|\rangle \qquad \langle \mathbf{if}\ (\square m \cdot M[\bar{q}] = m \to S_m)\ \mathbf{fi}, \rho\rangle \xrightarrow{ok} \langle S_m, M_m\rho M_m^\dagger\rangle$$

$$\frac{\langle S_1, \rho\rangle \xrightarrow{ok} \langle\downarrow, \rho'\rangle}{\langle S_1; S_2, \rho\rangle \xrightarrow{ok} \langle S_2, \rho'\rangle} \qquad \frac{\langle S_1, \rho\rangle \xrightarrow{ok} \langle S_1', \rho'\rangle}{\langle S_1; S_2, \rho\rangle \xrightarrow{ok} \langle S_1'; S_2, \rho'\rangle} \qquad \frac{\langle S_1, \rho\rangle \xrightarrow{er} \langle\downarrow, \rho'\rangle}{\langle S_1; S_2, \rho\rangle \xrightarrow{er} \langle\downarrow, \rho'\rangle}$$

$$\langle \mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}, \rho\rangle \xrightarrow{ok} \langle\downarrow, M_0\rho M_0^\dagger\rangle$$

$$\langle \mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}, \rho\rangle \xrightarrow{ok} \langle S; \mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}, M_1\rho M_1^\dagger\rangle$$

Fig. 4. Operational semantics.

final execution has an interesting label, we use the notation $\xrightarrow{\epsilon}{}^*$, where ϵ is the exiting condition, and $\epsilon = ok$ if no step is taken.

4.2.2 Denotational Semantics. As mentioned in Sec. 3, "the mixture of all reachable states" is a critical component of our incorrectness triple. The operational semantics characterizes one possible execution path at a time, which is not convenient for formalizing the incorrectness triple. We introduce denotational semantics to collect those reachable states from an input program state. The denotational semantics prove to be equivalent to the standard operational semantics.

By convention, we use $[\![S]\!]_\epsilon$ to denote the semantic function of a program S with exit condition ϵ. The semantic function has the type below. Intuitively, $[\![S]\!]_\epsilon$ maps a program state to the collection of reachable final states with exit condition ϵ.

$$[\![S]\!]_\epsilon : \mathcal{D}^-(\mathcal{H}) \to \mathit{Mset}(\mathcal{D}^-(\mathcal{H}))$$

Here $\mathit{Mset}(A)$ is the type of *multi-sets* (sometimes called *bags*) over the universe A. A multi-set is defined as a function of type $A \to \mathbb{N}$ that maps an element to its multiplicity. We use multi-sets instead of sets because the same final state might be obtained from different execution paths.

Formally, we define the semantic function in Fig. 5, with auxiliary definitions listed at the bottom. Most of the formulations explain themselves. We assign the meaning of impossible executions like $[\![\mathbf{error}]\!]_{ok}\rho$ with the $\mathbf{0}$-state, indicating this is an impossible event. Among these auxiliary definitions, $\mathcal{M}_m$ denotes the semantic function of M_m in the measurement M, and we use $\mathcal{R}$ for a function of type $\mathcal{D}^-(\mathcal{H}) \to \mathit{Mset}(\mathcal{D}^-(\mathcal{H}))$, $\mathcal{R}\rho$ is the multi-set obtained by applying $\mathcal{R}$ to ρ, and $\mathcal{R}\rho\rho'$ is the multiplicity of ρ' in $\mathcal{R}\rho$. The operation $\nu_1 \uplus \nu_2$ is the union operation over multi-sets ν_1 and ν_2, and in $\mathcal{R}_1 \uplus \mathcal{R}_2$, $\uplus$ is the pointwise lifted operation between multi-set valued functions. By our denotational semantics, the probabilistic mixture of all reachable states can be formulated by the sum of a multi-set ν of partial density matrices. The sum converges [Selinger 2004; Ying 2012] and thus is well-defined.

For example, let S_{Bell} be the program in Fig. 1a. Then we have

$$[\![S_{Bell}]\!]_{ok}(|00\rangle\langle 00|) = \{|\ \tfrac{1}{2}\,|11\rangle\langle 11|, \mathbf{0}, \mathbf{0}, \tfrac{1}{2}\,|00\rangle\langle 00|\ |\},$$

where the two $\mathbf{0}$-states correspond to the two impossible branches. The multi-set notation $\{|\ \cdot\ |\}$ wraps the elements and repeats them with their corresponding multiplicities. Since we have

$$[\![\mathbf{error}]\!]_{ok}\rho = \mathbf{0} \qquad [\![\mathbf{error}]\!]_{er}\rho = \rho$$

$$[\![\mathbf{skip}]\!]_{ok}\rho = \rho \qquad [\![\mathbf{skip}]\!]_{er}\rho = \mathbf{0}$$

$$[\![q := |0\rangle]\!]_{ok}\rho = \sum |0\rangle_q\langle n|\rho|n\rangle_q\langle 0| \qquad [\![q := |0\rangle]\!]_{er}\rho = \mathbf{0}$$

$$[\![\bar{q} := U\bar{q}]\!]_{ok}\rho = U\rho U^\dagger \qquad [\![\bar{q} := U\bar{q}]\!]_{er}\rho = \mathbf{0}$$

$$[\![S_1; S_2]\!]_{ok} = [\![S_1]\!]_{ok} \circ [\![S_2]\!]_{ok} \qquad [\![S_1; S_2]\!]_{er} = ([\![S_1]\!]_{ok} \circ [\![S_2]\!]_{er}) \uplus [\![S_1]\!]_{er}$$

$$[\![\mathbf{if}\ (\square m \cdot M[\bar{q}] = m \to S_m)\ \mathbf{fi}]\!]_{ok} = \biguplus_m (\mathcal{M}_m \circ [\![S_m]\!]_{ok})$$

$$[\![\mathbf{if}\ (\square m \cdot M[\bar{q}] = m \to S_m)\ \mathbf{fi}]\!]_{er} = \biguplus_m (\mathcal{M}_m \circ [\![S_m]\!]_{er})$$

$$[\![\mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}]\!]_{ok} = \biguplus_{n\in\mathbb{N}} ((\mathcal{M}_1 \circ [\![S]\!]_{ok})^n \circ \mathcal{M}_0)$$

$$[\![\mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}]\!]_{er} = \biguplus_{n\in\mathbb{N}} ((\mathcal{M}_1 \circ [\![S]\!]_{ok})^n \circ (\mathcal{M}_1 \circ [\![S]\!]_{er}))$$

$$\mathcal{M}_m\rho = M_m\rho M_m^\dagger \qquad \nu_1 \uplus \nu_2 = \{(\rho, \nu_1(\rho) + \nu_2(\rho)) \mid \rho \in \mathcal{D}^-(\mathcal{H})\}$$

$$(\mathcal{R}_1 \uplus \mathcal{R}_2)\rho = \mathcal{R}_1\rho \uplus \mathcal{R}_2\rho \qquad \mathcal{R}^0\rho = \{(\rho, 1)\} \qquad \mathcal{R}^n = \mathcal{R}^{n-1} \circ \mathcal{R}$$

$$(\mathcal{R}_1 \circ \mathcal{R}_2)\rho = \{(\rho'', n_1 n_2) \mid n_1 = \mathcal{R}_1\rho\rho' \wedge n_2 = \mathcal{R}_2\rho'\rho'' \wedge \rho, \rho', \rho'' \in \mathcal{D}^-(\mathcal{H})\}$$

Fig. 5. Denotational semantics.

already absorbed the probability[7] of its corresponding execution into the partial density matrix, the probabilistic mixture of all reachable states is obtained by summing up the multi-set directly:

$$\sum [\![S_{Bell}]\!]_{ok}(|00\rangle\langle 00|) = \tfrac{1}{2}|11\rangle\langle 11| + \tfrac{1}{2}|00\rangle\langle 00|\,.$$

It is straightforward to prove that the denotational semantics is equivalent to the operational semantics, as formulated in Theorem. 4.1. The proof is an induction on the structure of a program.

Theorem 4.1. *For any program S, and $\rho \in \mathcal{D}^-(\mathcal{H})$, the denotational semantics is equivalent to the operational semantics modulo $\mathbf{0}$-states, that is,*

$$([\![S]\!]_\epsilon\rho)\{\mathbf{0} \rightsquigarrow 0\} = (\{(\rho', n) \mid \langle S, \rho\rangle \xrightarrow{\epsilon}{}^*_n \langle\downarrow, \rho'\rangle\})\{\mathbf{0} \rightsquigarrow 0\}.$$

Here $\{\mathbf{0} \rightsquigarrow 0\}$ means discarding $\mathbf{0}$-states from the multi-set. We discard $\mathbf{0}$-states because the denotational semantics would introduce other multiplicities of $\mathbf{0}$-states when encountering impossible executions like **skip** *with er exit condition. The notation $\langle S, \rho\rangle \xrightarrow{\epsilon}{}^*_n \langle\downarrow, \rho'\rangle$ means there are n distinguished execution paths that terminates at ρ' with exit condition ϵ.*

5 SPECIFICATION FORMULA

In this section, we develop the quantum incorrectness triple of the form $[P]S[\epsilon : Q]$. Intuitively, if P under-approximates the initial state, then Q under-approximates the probabilistic mixture of reachable final states with exit condition ϵ. Here P and Q are projection-based quantum predicates treated semantically using their corresponding matrices.

[7] The probability of certain branch is the trace of the corresponding output state (partial density matrix).

5.1 Under-Approximating Quantum States

In the context of bug-catching, the triple $[P]S[\epsilon:Q]$ first needs to characterize erroneous states using a predicate. In the classical settings, characterizing an erroneous state σ w.r.t. a predicate p is straight forward by using satisfaction and negation of the predicate:

$$\sigma \nvDash p \Leftrightarrow \sigma \vDash \neg p. \tag{2}$$

However, satisfaction is not suitable for characterizing *incorrectness* in the quantum settings: given an assertion P_c and an erroneous state $\rho_e \nvDash P_c$, sometimes we cannot find an appropriate Q_e such that $\rho_e \vDash Q_e$ and Q_e excludes correct states, i.e., any $\rho_c \vDash P_c$ does not satisfy Q_e. More concretely, let $\mathbf{0} \subset P_c \subset I$, and let $\rho_e = I/\mathrm{Tr}(I) \nvDash P_c$, then any Q_e that has $\rho_e \vDash Q_e$ would falsely capture any state $\rho_c \vDash P_c$, because $\mathrm{supp}(\rho_c) \subseteq I = \mathrm{supp}(\rho_e) \subseteq Q_e$.

To capture incorrect quantum states, we need a quantum version of equation (2). We achieve this goal by introducing the under-approximation relation.

Definition 5.1 (Under-approximation). *A projection P* under-approximates *a quantum state $\rho \in \mathcal{D}^-(\mathcal{H})$, denoted by $\rho \dashv P$, if* $\mathrm{supp}(\rho) \supseteq P$.

As we can see, under-approximation relation can precisely characterize errors:

$$\rho_e \nvDash P_c \implies \exists Q_e \neq \mathbf{0}.\, (\rho_e \dashv Q_e) \wedge (\forall \rho_e' \dashv Q_e.\, \rho_e' \nvDash P_c).$$

That is, for any erroneous state ρ_e violating P_c, it can be under-approximated by some non-trivial projection Q_e, and this under approximation will not falsely capture correct states. Under-approximation relation is also crucial for interpreting "achieving" a predicate, which we will explain later.

The under-approximation relation is the inverted satisfaction. Logical connections under the under-approximation relation are sometimes counter intuitive compared with those under the satisfaction, for example:

$$\begin{aligned} \rho \vDash P_1 \ \wedge\ \rho \vDash P_2 \ &\Leftrightarrow\ \rho \vDash P_1 \wedge P_2 \\ \rho \dashv P_1 \ \wedge\ \rho \dashv P_2 \ &\Leftrightarrow\ \rho \dashv P_1 \vee P_2. \end{aligned}$$

5.2 Incorrectness Triple for Quantum Programs

Based on the under-approximation relation, we generalize the incorrectness triple by O'Hearn to the quantum settings and obtain the validity defined below. In this definition, "achieving" a projection Q is interpreted as Q under-approximating the mixture of reachable states.

Definition 5.2 (Strong Validity). *A QIL triple is strongly valid (or valid for short), denoted by $\vDash [P]S[\epsilon:Q]$ if for any $\rho \in \mathcal{D}^-(\mathcal{H})$ we have*

$$\rho \dashv P \implies \textstyle\sum [\![S]\!]_\epsilon \rho \dashv Q.$$

We use the term *strong validity* to distinguish this formulation from those alternatives in Sec. 8. This definition says that if a state is under-approximated by P, the mixture of its reachable states with exit condition ϵ is under-approximated by Q. We argue that this interpretation of "achieving" is reasonable from a reachability point of view: given $\vDash [P]S[\epsilon:Q]$, starting from an initial state under-approximated by P, it is possible (with non-zero probability) to obtain any pure state $|\psi\rangle \in Q$ by measuring some reachable state (with exit condition ϵ) using the measurement $M = \{|\psi\rangle\langle\psi|, I - |\psi\rangle\langle\psi|\}$.

Introducing the mixture of reachable states instead of discussing single execution paths is crucial for efficient reasoning. It allows us to have the disjunction rule, without which the number of post-conditions grows exponentially with respect to the number of sequenced branches. Alternative

formulations based on a single execution path (classical and strict validities) can be found in Sec 8, where we discuss in more detail why the disjunction rule does not hold for these formulations and the consequences of not having such a rule.

Although the formulation compares Q with the mixture of the reachable states of *all* execution paths, it is safe to find smaller Q corresponding to *some* executions to construct a valid triple. This coincides with the remark by O'Hearn [O'Hearn 2019]:

> "For correctness reasoning, you get to forget information as you go along a path, but you must remember all the paths. For incorrectness reasoning, you must remember information as you go along a path, but you get to forget some of the paths."

The validity of an incorrectness triple sets the theoretical foundation for static bug-catching with projection-based assertions [Li et al. 2020]. It is straightforward from Def. 5.2 that for the $\mathbf{assert}(\bar{q}, R)$ statement and any presumption P, we have $\models [P]\mathbf{assert}(\bar{q}, R)[er{:}\mathrm{supp}(R^{\perp}PR^{\perp})]$. [8] While the correctness triple $\models \{R\}\mathbf{assert}(\bar{q}, R)\{R\}$ of the applied quantum Hoare logic [Zhou et al. 2019] guarantees that we can safely ignore the **assert** statement in the reasoning when the assertion is satisfied, an incorrectness triple $\models [P]\mathbf{assert}(\bar{q}, R)[er{:}\mathrm{supp}(R^{\perp}PR^{\perp})]$ with $R^{\perp}PR^{\perp} \neq \mathbf{0}$ ensures the assertion would raise an *er* with non-zero probability for some state $\rho \sqsupset P$.

More discussions about the validity of incorrectness triples are given in Sec 8 if readers are interested in why we choose such a kind of formulation.

5.3 Duality between Correctness and Incorrectness Triples

Validity of triples in QIL and the applied quantum Hoare logic [Zhou et al. 2019] are two sides of the same coin when interpreted with predicate transformers.

Definition 5.3. *For any quantum program S defined in Section 4.2 and quantum predicate P, we define the post image of program S with respect to P as follows*

$$post(\llbracket S \rrbracket_{\epsilon})P = \mathrm{supp}(\sum \llbracket S \rrbracket_{\epsilon}(\rho)) \quad \textit{where } \rho = \begin{cases} P/Tr(P) & \textit{if } P \neq \mathbf{0} \\ \mathbf{0} & \textit{otherwise} \end{cases}$$

Note that the choice of ρ is not unique: any ρ that has $\mathrm{supp}(\rho) = P$ would result in an equivalent definition. Based on the operator $post(\llbracket S \rrbracket_{\epsilon})$, we give an equivalent formulation for the validity of incorrectness triple in Lemma 5.1.

Lemma 5.1. *For a quantum program S and a quantum predicate P, we have*

$$\models [P]S[\epsilon{:}Q] \quad \textit{iff} \quad post(\llbracket S \rrbracket_{\epsilon})P \supseteq Q$$

The operator $post(\llbracket S \rrbracket_{\epsilon})$ reveals the connection between the applied quantum Hoare logic [Zhou et al. 2019] and QIL. It is straightforward that when S does not contain the **error** statement, $post(\llbracket S \rrbracket_{ok})P \subseteq Q$ is exactly the partial correctness validity $\models^{\mathrm{a}}_{\mathrm{par}} \{P\}S\{Q\}$ in the applied quantum Hoare logic. The duality is then obvious, as shown below.

$$\begin{array}{lll} \models^{\mathrm{a}}_{\mathrm{par}} \{P\}S\{Q\} & \text{iff} & post(\llbracket S \rrbracket_{ok})P \subseteq Q \\ \models [P]S[\epsilon{:}Q] & \text{iff} & post(\llbracket S \rrbracket_{\epsilon})P \supseteq Q \end{array}$$

Specifically, we prove that the projection $post(\llbracket S \rrbracket_{ok})P$ is the strongest over-approximate post for applied Hoare logic and the weakest under-approximate post for QIL, as shown in Lemma 5.2.

Lemma 5.2. *When S does not contain the* **error** *statement, for any projection P we have*

$$post(\llbracket S \rrbracket_{ok})P = \wedge\{Q \mid \models^{a}_{par} \{P\}S\{Q\}\}$$
$$post(\llbracket S \rrbracket_{\epsilon})P = \vee\{Q \mid \models [P]S[\epsilon{:}Q]\}$$

[8] We write result assertions in red for abnormal termination.

Since the weakest under-approximate post is the disjunction of all quantum post predicates satisfying $[P]S[\epsilon:Q]$, Lemma 5.2 gives a starting point for under-approximating program analysis and guarantees that incorrectness reasoning is sound when shrinking the postcondition.

Why not directly replace all quantum behavior with non-determinism for pure reachability analysis? Here we discuss why we do not choose the set of quantum states as predicates, and apply classical incorrectness logic directly. One is that sets are less compact compared with projections. For example, given a set of the form $\{|\psi\rangle \,||\psi\rangle = \cos(x_i)\,|00\rangle + \sin(x_i)\,|11\rangle\}$, where x_i is the i-th number in the sequence of Collatz conjecture for a random integer n. It is hard to specify the elements in the set neatly (basically a record of the sequence), but it can be easily regulated by a projection $|00\rangle\langle 00| + |11\rangle\langle 11|$.

Another reason is that, when used as loop invariants/variants, sets may converge much slower than projections. Take Grover's algorithm as an example, the state within the loop body keeps rotating in a 2-dimensional subspace, which means the corresponding projection converges within 3 loop unrolling (constant time!). If we use sets instead, since the resulting states after each iteration are very likely to be different from each other (e.g., by choosing $N = 5$ and $M = 1$), we will have to keep unrolling the while-loop until the program terminates (depending on the number of iterations).

6 THE PROOF SYSTEM

In this section, we develop the proof system for QIL based on the strong validity in Def. 5.2. The proof rules of quantum incorrectness logic are shown in Fig. 6. Following O'Hearn [O'Hearn 2019], we use $\vdash [P]S[ok:Q_1][er:Q_2]$ as an abbreviation for $\vdash [P]S[ok:Q_1]$ and $\vdash [P]S[er:Q_2]$. We write result assertions for normal termination in green and abnormal in red.

$$\textsc{Empty}\quad \vdash [P]S[\epsilon:\mathbf{0}] \qquad \textsc{Error}\quad \vdash [P]\mathbf{error}[ok:\mathbf{0}][er:P] \qquad \textsc{Skip}\quad \vdash [P]\mathbf{skip}[ok:P][er:\mathbf{0}]$$

$$\textsc{Unitary}\quad \vdash [P]\bar{q} := U[\bar{q}][ok:UPU^\dagger][er:\mathbf{0}] \qquad \textsc{Init}\quad \vdash [P]q := |0\rangle\,[ok:\mathrm{supp}(\sum_n |0\rangle_q \langle n|P|n\rangle_q \langle 0|)][er:\mathbf{0}]$$

$$\textsc{Seq1}\quad \frac{\vdash [P]S_1[ok:R] \quad \vdash [R]S_2[\epsilon:Q]}{\vdash [P]S_1;S_2[\epsilon:Q]} \qquad \textsc{Seq2}\quad \frac{\vdash [P]S_1[er:Q]}{\vdash [P]S_1;S_2[er:Q]} \qquad \textsc{Order}\quad \frac{P \supseteq P' \quad \vdash [P']S[\epsilon:Q'] \quad Q' \supseteq Q}{\vdash [P]S[\epsilon:Q]}$$

$$\textsc{Disjunction}\quad \frac{\vdash [P_1]S[\epsilon:Q_1] \quad \vdash [P_2]S[\epsilon:Q_2]}{\vdash [P_1 \vee P_2]S[\epsilon:Q_1 \vee Q_2]} \qquad \textsc{If}\quad \frac{\vdash [\mathrm{supp}(M_m P M_m^\dagger)]S_m[\epsilon:Q]}{\vdash [P]\mathbf{if}\ (\square m \cdot M[\bar{q}] = m \ \rightarrow\ S_m)\ \mathbf{fi}[\epsilon:Q]}$$

$$\textsc{While1}\quad \frac{\forall n. \vdash [\mathrm{supp}(M_1 P_n M_1^\dagger)]S[ok:P_{n+1}]}{\vdash [P_0]\mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}[ok:\mathrm{supp}(M_0 P_N M_0^\dagger)]}$$

$$\textsc{While2}\quad \frac{\forall n. \vdash [\mathrm{supp}(M_1 P_n M_1^\dagger)]S[ok:P_{n+1}] \quad \vdash [\mathrm{supp}(M_1 P_N M_1^\dagger)]S[er:Q]}{\vdash [P_0]\mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}[er:Q]}$$

Fig. 6. Proof rules for quantum incorrectness logic.

The first three rules have similar forms as their classical counterparts. The Empty rule is a direct generalization of its classical counterpart, where $\mathbf{0}$ is a trivial valid post predicate that contains no

meaningful state, a quantum extension to the classical *false* assertion (the empty set). The Error and Skip rules are straightforward from their semantics since they do not modify the program state along *er* and *ok* paths, respectively.

The Unitary and Init rules characterize how these two statements alter the support of quantum state. Note that in the Init rule, $\sum_n |0\rangle_q \langle n|P|n\rangle_q \langle 0|$ is not necessarily a projection, we need to lift it to its support before assigning as a postcondition.

The Seq rules are of the same form as in classical settings, where Seq1 is for normal sequencing, and Seq2 is for short-circuiting when S_1 raises an *er*.

The Order rule is the quantum version of the classical consequence rule. By interpreting the subset relation as implication $\Rightarrow$, the rule has the same form of the consequence rule below.

$$\frac{P \Leftarrow P' \quad \vdash [P']S[\epsilon : Q'] \quad Q' \Leftarrow Q}{\vdash [P]S[\epsilon : Q]}$$

Rules for dropping conjunctions/disjunctions can be derived from the Order rule by noticing the fact that $P_i \supseteq P_1 \wedge P_2$ and $Q_1 \vee Q_2 \supseteq Q_i$ for $i \in \{1, 2\}$, as shown below.

$$\frac{\vdash [P_1 \wedge P_2]S[\epsilon : Q]}{\vdash [P_i]S[\epsilon : Q]} \qquad \frac{\vdash [P]S[\epsilon : Q_1 \vee Q_2]}{\vdash [P]S[\epsilon : Q_i]}$$

Note that the ability to shrink the postcondition soundly is a hallmark of under-approximation, which allows us to control the reasoning scale.

The Disjunction rule is also a quantum version of its classical counterpart. It allows us to merge the reasoning for multiple branches, which is crucial to the efficiency of reasoning.

The If rule is the quantum analogy of the Choice rule in IL. The difference lies in the premise of the rule, where we require $\vdash [\mathrm{supp}(M_m P M_m^\dagger)]S[\epsilon : Q]$ instead of $\vdash [P]S[\epsilon : Q]$ because measurement has a side effect on the quantum state.

The While rules can be interpreted as a finite sequential composition of the If rule and Seq rules after unrolling the loop body for finite times, where P_n represents the result predicate for the n-fold sequential composition of measurement and the loop body. Recall that incorrectness logic is for the reasoning about reachability; these rules do not require the termination of all executions but only guarantee some execution paths that reach the result predicate.

Assert

$$\frac{Q_{ok} = \mathrm{supp}(RPR^\dagger) \quad Q_{er} = \mathrm{supp}(R^\perp P R^{\perp\dagger})}{[P]\mathbf{assert}(\bar{q}, R)[ok : Q_{ok}][er : Q_{er}]}$$

Derived If

$$\frac{\vdash [\mathrm{supp}(M_m P M_m^\dagger)]S_m[\epsilon : Q_m] \text{ for all } m}{\vdash [P]\mathbf{if}\ (\square m \cdot M[\bar{q}] = m \ \rightarrow\ S_m)\ \mathbf{fi}[\epsilon : \vee Q_m]}$$

Derived While

$$\frac{\forall n < N. \vdash [\mathrm{supp}(M_1 P_n M_1^\dagger)]S[ok : P_{n+1}]}{\vdash [P_0]\mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}[ok : \vee_{i=0}^{N} \mathrm{supp}(M_0 P_i M_0^\dagger)]}$$

Fig. 7. Useful derived rules.

We list several other derived rules in Fig. 7. The If rule combined with the Skip and Error rules derive the proof rule for the **assert** statement. We also use the Disjunction rule to derive new practical rules for **if** and **while** statements, which merge the reasoning results of multiple branches. Note that in the Derived While rule, we made the bound N for n explicitly for finite loop unrolling. It can be derived from While1 rule by letting $P_n = \mathbf{0}$ for $n \geq N$. Our logic is both sound and complete, as formulated by the following theorem.

Theorem 6.1. [Soundness & Completeness] *For any program S, exit condition ϵ, projections P and Q, we have,*

$$\vdash [P]S[\epsilon:Q] \iff \vDash [P]S[\epsilon:Q]$$

Automating the inference with finite loop unrolling. Although one may use the Derived While rule and a fixed bound N to make the reasoning sound and terminate within finite steps (loop unrolling), such a bound usually means dropping information and making the reasoning incomplete. Stronger reasoning like the While1 rule is needed in general. However, it is unclear how to automatically infer a backward variant $\{P_n\}$ even for finite-dimensional quantum systems because the state space and possible projections are uncountably infinite.

Instead of inferring P_n, we find the post predicate in the Derived While does not change when N is large enough. We prove a stronger completeness result which indicates finite loop unrolling is sufficient for complete reasoning.

Theorem 6.2 (Completeness with bounded While rules). *Replacing the While1 and While2 rule with the following bounded While rules results in another sound and complete proof system.*

Bounded While1

$$\frac{\forall n < \dim(\mathcal{H}). \vdash [\mathrm{supp}(M_1 P_n M_1^\dagger)]S[ok:P_{n+1}] \quad N \le \dim(\mathcal{H})}{\vdash [P_0]\textbf{while } M[\bar{q}] = 1 \textbf{ do } S \textbf{ od}[ok:\mathrm{supp}(M_0 P_N M_0^\dagger)]}$$

Bounded While2

$$\frac{\forall n < \dim(\mathcal{H}). \vdash [\mathrm{supp}(M_1 P_n M_1^\dagger)]S[ok:P_{n+1}] \quad N \le \dim(\mathcal{H}) \quad \vdash [\mathrm{supp}(M_1 P_N M_1^\dagger)]S[er:Q]}{\vdash [P_0]\textbf{while } M[\bar{q}] = 1 \textbf{ do } S \textbf{ od}[er:Q]}$$

Here $\dim(\mathcal{H})$ is the dimension of the state space of the quantum system.

The intuition is that the rank of $\vee_{i<N} P_i$ is bounded by $\dim(\mathcal{H})$, and once $\vee_{i\le N-1} P_i = \vee_{i\le N} P_i$, the sequence stops increasing, thus must converge before N reaches $\dim(\mathcal{H}) + 1$. The theorem indicates that our logic is decidable and has an upper bound for the time complexity of inference. The upper bound is only relevant to the dimension of the quantum system and the number of nested **while**-loops. In practice, the dimension $\dim(\mathcal{H})$ can still be impractically large (e.g., for an n-qubit system, $\dim(\mathcal{H}) = 2^n$), in which case we need to balance between efficiency and completeness by employing While rules with a smaller bound.

Just like the classical incorrectness logic, our logic system avoids false positives by definition, that is, the post-condition is achievable. Theorem 6.2 further shows that our proof system does not miss erroneous states regulated by projections. However, due to the limited expressiveness of projections, our proof system does miss erroneous states violating quantitative properties that cannot be captured by projections, i.e., when correct and erroneous states share the same support. For example, mixed states $0.2\,|0\rangle\langle 0| + 0.8\,|1\rangle\langle 1|$ and $0.5\,|0\rangle\langle 0| + 0.5\,|1\rangle\langle 1|$ are indistinguishable with respect to any projection.

7 REASONING USING THE LOGIC

This section demonstrates how to reason about the quantum program using our logic by three examples. We first show and compare the reasoning of Grover's algorithm with aQHL and QIL. Then we reason about quantum teleportation and a repeat-until-success program where two types of bugs were inserted respectively, according to Huang et al. [Huang and Martonosi 2019b], and show how the **assert** statement combined with our proof system captures the bugs *statically*.

7.1 Grover's Algorithm

Grover's algorithm [Grover 1996] searches and finds the unique input to a black box function that produces a particular output value and provides quadratic speedup over its classical counterparts. The implementation and corresponding proof sketch are listed in Fig. 8, where we put the aQHL and QIL predicates together to show their connection.

$$
\begin{array}{ll}
 & [|0\rangle_{\bar{q}}\langle 0| \otimes |0\rangle_{\bar{r}}\langle 0|] \quad \{|0\rangle_{\bar{q}}\langle 0| \otimes |0\rangle_{\bar{r}}\langle 0|\} \\
1: & \bar{q} := H^{\otimes d}\bar{q}; \\
 & [ok{:}P_0 \otimes |0\rangle_{\bar{r}}\langle 0|] \quad \{P_0 \otimes |0\rangle_{\bar{r}}\langle 0|\} \\
2: & \textbf{while } M[\bar{r}] = 1 \textbf{ do} \\
 & [ok{:}P_n \otimes |n\rangle_{\bar{r}}\langle n|] \quad \{\textstyle\sum_{n=0}^{R-1}(P_n \otimes |n\rangle_{\bar{r}}\langle n|)\} \\
3: & \quad \bar{q} := U_\omega\bar{q}; \\
4: & \quad \bar{q} := H^{\otimes d}\bar{q}; \\
5: & \quad \bar{q} := Ph\bar{q}; \\
6: & \quad \bar{q} := H^{\otimes d}\bar{q}; \\
 & [ok{:}P_{n+1} \otimes |n\rangle_{\bar{r}}\langle n|] \quad \{\textstyle\sum_{n=0}^{R-1}(P_{n+1} \otimes |n\rangle_{\bar{r}}\langle n|)\} \\
7: & \quad \bar{r} := U_+\bar{r}; \\
 & [ok{:}P_{n+1} \otimes |n+1\rangle_{\bar{r}}\langle n+1|] \quad \{\textstyle\sum_{n=0}^{R}(P_n \otimes |n\rangle_{\bar{r}}\langle n|)\} \\
8: & \textbf{od}; \\
 & [ok{:}P_R \otimes |R\rangle_{\bar{r}}\langle R|] \quad \{P_R \otimes |R\rangle_{\bar{r}}\langle R|\} \\
9: & \textbf{if } (\square m \cdot N[\bar{q}] = m \rightarrow \textbf{skip}) \textbf{ fi}; \\
 & [ok{:}\textstyle\sum_i |s_i\rangle_{\bar{q}}\langle s_i| \otimes |R\rangle_{\bar{r}}\langle R|] \quad \{\textstyle\sum_i |s_i\rangle_{\bar{q}}\langle s_i| \otimes |R\rangle_{\bar{r}}\langle R|\} \\
10: & \textbf{assert}(\bar{q}, \textstyle\sum_i |s_i\rangle_{\bar{q}}\langle s_i|); \\
 & [ok{:}\textstyle\sum_i |s_i\rangle_{\bar{q}}\langle s_i| \otimes |R\rangle_{\bar{r}}\langle R|]\,[er{:}\mathbf{0}]
\end{array}
$$

$$
\begin{array}{ll}
P_0 = |\psi\rangle\langle\psi| & |\psi\rangle = \frac{1}{\sqrt{2^d}}\sum_{x\in\{0,1\}^d}|x\rangle \\
P_n = G^n P_0 G^{n\dagger} & G = (2|\psi\rangle\langle\psi| - I)U_\omega \\
M = \{M_0, M_1\} & M_0 = |R\rangle_{\bar{r}}\langle R|,\ M_1 = M_0^\perp \\
N = \{N_m \mid m \in \{0,1\}^d\} & N_m = |m\rangle\langle m|
\end{array}
$$

Fig. 8. Program implementing the Grover's algorithm.

Suppose we hope to find one of the M solutions $\{s_1, \ldots, s_M\}$ of equation $f(x) = 1$ from the domain of size $N = 2^d$, where $f : \{0,1\}^d \rightarrow \{0,1\}$. The Grover's algorithm prepares a uniform superposition $\frac{1}{\sqrt{2^d}}\sum_{x\in\{0,1\}^d}|x\rangle$ on the d qubits $\bar{q}$ (line 1), performs the "Grover iteration" for R times (line 2-8) before measuring the qubits $\bar{q}$ (line 9), and guarantees the resulting state in $\bar{q}$ encodes a solution for a high probability. The Grover iteration contains an oracle U_ω that acts as $U_\omega|x\rangle = (-1)^{f(x)}|x\rangle$, and a conditional phase shift operator Ph that acts as $Ph|x\rangle = -(-1)^{x=0}|x\rangle$. The quantum register $\bar{r}$ is a counter for the **while**-loop, every time at the end of the loop body it is increased using the operator U_+ that acts as $U_+|n\rangle = |n+1 \mod 2^{|\bar{r}|}\rangle$. Finally, line 10 tests if the output state in $\bar{q}$ is a solution.

To make it easier to understand how to insert assertions in the loop body in Fig. 8, it would be better to give a more intuitive explanation for Grover iteration from a geometric point view. We define two special state $|\alpha\rangle$ and $|\beta\rangle$

$$
|\alpha\rangle = \frac{1}{\sqrt{N-M}}\sum_{x_t}|x_t\rangle \qquad |\beta\rangle = \frac{1}{\sqrt{M}}\sum_{x_s}|x_s\rangle
$$

where $\sum_{x_s}$ ($\sum_{x_t}$) indicates a sum over all states $|x\rangle$ which are (not) solutions to the search problems. Let the initial state be $|0\rangle^{\otimes d}$ and we have the equal superposition state $|\psi\rangle$ before entering the while loop:

$$
|\psi\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle
$$

The whole of line 3-6 performs the Grover operator $G = H^{\otimes d}PhH^{\otimes d}U_\omega = (2\,|\psi\rangle\langle\psi| - I)U_\omega$ on state $|\psi\rangle = \cos(\theta/2)\,|\alpha\rangle + \sin(\theta/2)\,|\beta\rangle$

$$G\,|\psi\rangle = \cos\frac{3\theta}{2}\,|\alpha\rangle + \sin\frac{3\theta}{2}\,|\beta\rangle \qquad (\cos\theta/2 = \sqrt{(N-M)/N})$$

which turns out to be a rotation of θ radians on the vector $|\psi\rangle$ in the 2-dimensional subspace $|\alpha\rangle\langle\alpha| + |\beta\rangle\langle\beta|$ spanned by $|\alpha\rangle$ and $|\beta\rangle$. Generally, we can make the algorithm succeed with a high probability, i.e., to make the rotation end up with a position that is as close to the solution $|\beta\rangle$ as possible. The number of the Grover iterations is upper-bounded by $R = O(\sqrt{N/M})$.

In Fig. 8, we choose M, N, R appropriately such that the program succeeds with probability 1. We insert predicates before or after a line of code and obtain the proof sketches. Predicates in $[-]$ are for QIL, and predicates in $\{-\}$ are for aQHL. It is not surprising to find the proof sketches for QIL and aQHL being mostly identical, since the proof sketches always use the largest/strongest post-conditions, and the two proof systems are connected by $post(\llbracket S \rrbracket_\epsilon)$. The difference lies in the **while**-loop: while QIL concerns about every single execution of the loop body using the loop variant $P_n \otimes |n\rangle_{\bar{r}}\langle n|$, aQHL merges the reasoning of these executions using a loop invariant which is essentially the disjunction of the loop variants.

For general cases where the choice of M and N does not guarantee success with the probability being 1, we may instead insert an assertion $\mathbf{assert}(\bar{q}, |\alpha\rangle\langle\alpha| + |\beta\rangle\langle\beta|)$ at the end of the loop body. Using this assertion, we would miss erroneous implements such as a wrong loop guard R' that has no effect on the subspace $|\alpha\rangle\langle\alpha| + |\beta\rangle\langle\beta|$. [9] But we can still capture any kind of errors that makes state $G^i\,|\psi\rangle$ $(0 \le i \le R)$ end up not lying in the subspace $|\alpha\rangle\langle\alpha| + |\beta\rangle\langle\beta|$, e.g., an erroneous implementation of Ph.

7.2 Quantum Teleportation

Quantum teleportation is a technique that transports quantum states through classical communication. As shown in the Fig. 9, when a Bell state β_{00} is shared between qubits q_1 and q_2, to teleport the state of q_0 to q_2 we may apply H gate on q_0 and measure q_0 and q_1 with $M = \{|0\rangle\langle 0|, |1\rangle\langle 1|\}$, then apply X or Z gate to q_2 if the measurement outcome of q_0 or q_1 is 1, respectively. The output state of q_2 will always be the same as the input state of q_0 for all possible measurement outcomes. Thus, two parties sharing a Bell state may teleport the state of q_0 to q_2 via communicating with measurement outcomes in a classical channel.

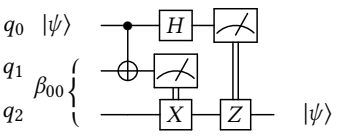


Fig. 9. Circuit for teleportation.

Figure. 10a shows the corresponding program of quantum teleportation circuit and its proof sketch, where we add an assertion at the end to check whether the quantum state of q_0 (described by projection R) is teleported to q_2. If we mismatch the controlled gates on the qubit q_2 in lines 3 and 4, as shown in the Figure. 10b, we introduce a bug of type "incorrect operations" according to [Huang and Martonosi 2019b]. The correct and erroneous programs can be easily reasoned about with the proof sketch and our proof rules, where the non-zero post of the erroneous program provides evidence of a bug. Note that after lines 3 and 4, we merge the result predicates of **if** branches using the Disjunction rule, thus avoid reasoning about exponentially many execution paths.

7.3 The RUS Example

We take a simple program implementing a repeat-until-success [Bocharov et al. 2015; Paetznick and Svore 2014] (RUS) algorithm as another example. RUS algorithms offer exact, fault-tolerant

[9] A bad R would reduce the success rate of Grover's algorithm.

$[R \otimes \beta_{00}\beta_{00}^\dagger]$
1 : $(q_0, q_1) := CNOT(q_0, q_1)$;
2 : $q_0 := Hq_0$;
$[ok: |\varphi\rangle\langle\varphi|]$
3 : **if** $(M[q_1] = 1 \rightarrow q_2 := Xq_2)$ **fi**;
$[ok: Q_1 \vee Q_2]$
4 : **if** $(M[q_0] = 1 \rightarrow q_2 := Zq_2)$ **fi**;
$[ok: I \otimes R]$
5 : **assert**(q_2, R)
$[er: \mathbf{0}]$

(a) Correct program

$[R \otimes \beta_{00}\beta_{00}^\dagger]$
1 : $(q_0, q_1) := CNOT(q_0, q_1)$;
2 : $q_0 := Hq_0$;
$[ok: |\varphi\rangle\langle\varphi|]$
3 : **if** $(M[q_1] = 1 \rightarrow q_2 := Zq_2)$ **fi**;
$[ok: Q_1 \vee Q_3]$
4 : **if** $(M[q_0] = 1 \rightarrow q_2 := Xq_2)$ **fi**;
$[ok: P_= \otimes R \ + P_=^\perp \otimes XZRZX]$
5 : **assert**(q_2, R)
$[er: P_=^\perp \otimes \mathrm{supp}(R^\perp XZRZXR^\perp)]$

(b) Erroneous program

$$M = \{M_0, M_1\} = \{|0\rangle\langle 0|, |1\rangle\langle 1|\} \qquad P_= = |00\rangle\langle 00| + |11\rangle\langle 11|$$
$$|\varphi\rangle = |00\rangle|\psi\rangle + |01\rangle X|\psi\rangle + |10\rangle Z|\psi\rangle + |11\rangle XZ|\psi\rangle$$
$$Q_1 = |00\rangle\langle 00| \otimes R + |00\rangle\langle 10| \otimes RZ + |10\rangle\langle 00| \otimes ZR + |10\rangle\langle 10| \otimes ZRZ$$
$$Q_2 = |01\rangle\langle 01| \otimes R + |01\rangle\langle 11| \otimes RZ + |11\rangle\langle 01| \otimes ZR + |11\rangle\langle 11| \otimes ZRZ$$
$$Q_3 = |01\rangle\langle 01| \otimes XZRZX + |01\rangle\langle 11| \otimes XZRX + |11\rangle\langle 01| \otimes XRZX + |11\rangle\langle 11| \otimes XRX$$

Fig. 10. Reasoning the quantum teleportation program.

implementations of a large set of single-qubit unitary gates that can be used to improve upon the approximate decomposition of single-qubit unitaries significantly. Implementing the algorithm requires wrapping RUS circuits into while-loops, which can be easily erroneous.

Fig. 11 is the smallest circuit for the loop body found in [Paetznick and Svore 2014] that implements non-Clifford single-qubit unitaries. The qubit q_0 is the auxiliary qubit, and q_1 is the target qubit. Besides H and $CNOT$ gates, there is another basic unitary gate $T = \cos(\pi/8)I - i\sin(\pi/8)Z$ used in the circuit. The circuit aims to apply a unitary operator $V = (I + i\sqrt{2}X)/\sqrt{3}$ on the target qubit. The desired operation will have been achieved when the measurement on the auxiliary qubit q_0 returns 0. When that happens, we can exit the program and return the result. Otherwise, we would have to rerun the circuit. To rerun the circuit based on the measurement result, we wrap the circuit into a while-loop. Note that the auxiliary qubit serves as the control qubit of the loop body and the auxiliary qubit for the RUS circuit at the same time. To make sure the program enters the while-loop body, it needs to set the auxiliary qubit to $|1\rangle$ at the beginning using an X gate and restore to $|0\rangle$ by an X gate again before executing the circuit.

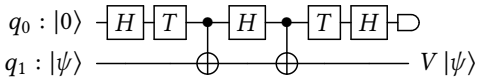


Fig. 11. An RUS circuit implementing $V = \frac{I+i\sqrt{2}X}{\sqrt{3}}$.

Fig. 12a,12b is the correct and erroneous implementations of the RUS procedure corresponding to Fig. 12 along with the proof sketch, respectively. Assume q_1 is of the state $R = |\psi\rangle\langle\psi|$ at the beginning, we add an assertion at line 11 to check whether the program implements the unitary gate V on q_1. The erroneous implementation contains a mistake at line 2: forgetting to restore the auxiliary qubit q_0 before entering the RUS circuit (line 3-9). The seemingly artificial implementation error corresponds to a bug type "incorrect quantum initial values" as reported in [Huang and Martonosi 2019b]: the initial value of the auxiliary qubit q_0 should be $|0\rangle$ instead of $|1\rangle$ before executing line 3-9, the RUS circuit.

We briefly describe how to reason about the erroneous program with our logic. Let the presumption of this erroneous program be $|0\rangle\langle 0| \otimes R$ for projection $R = |\psi\rangle\langle\psi|$, we have $[ok: |1\rangle\langle 1| \otimes R]$ by

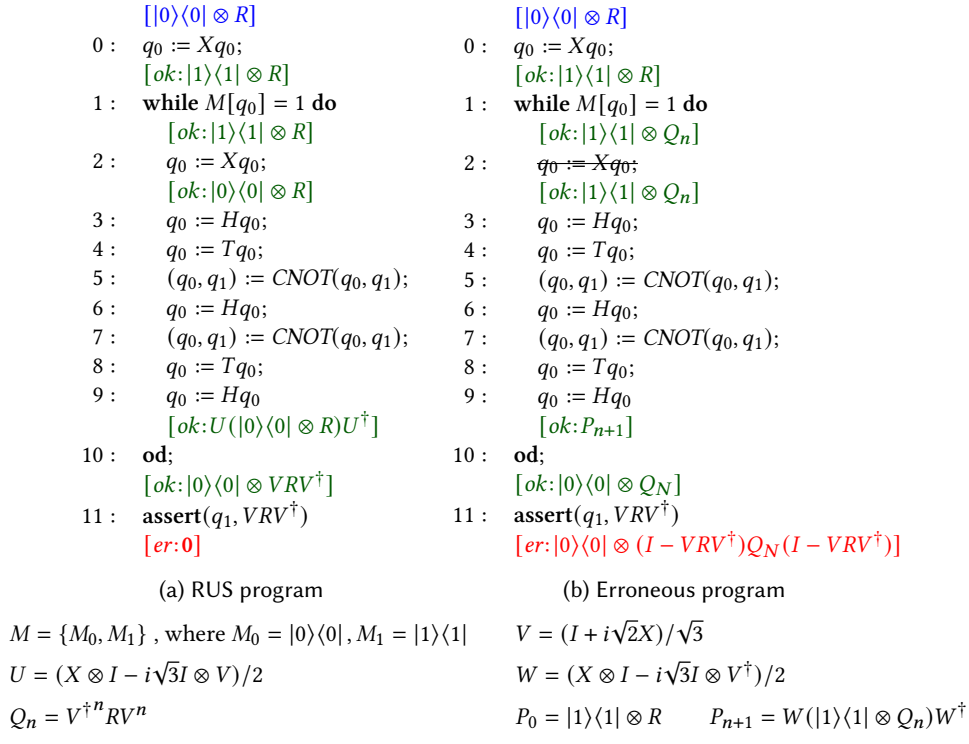


Fig. 12. Reasoning an RUS program.

the Unitary rule after line 0 before the **while**-loop. To apply the While1 rule, it suffices to find a series of P_n such that

$$P_0 = |1\rangle\langle 1| \otimes R \wedge \ \vdash [\mathrm{supp}(M_1 P_n M_1^\dagger)] S_{3-9} [ok{:}P_{n+1}]$$

where S_{3-9} is the erroneous loop body from line 3 to line 9. Since the S_{3-9} is a sequence of unitary statements, by rule Seq1 and Unitary, we have

$$\vdash [|1\rangle\langle 1| \otimes Q] S_{3-9} [ok{:}W(|1\rangle\langle 1| \otimes Q)W^\dagger]$$

for an arbitrary projection Q, where $W = \frac{1}{2}(X \otimes I - i\sqrt{3} I \otimes V^\dagger)$. After a little bit more calculating we have a non-trivial series of P_n and Q_n, as shown at the bottom of Fig. 12, such that $\mathrm{supp}(M_1 P_n M_1^\dagger) = |1\rangle\langle 1| \otimes Q_n$ and $P_{n+1} = W(|1\rangle\langle 1| \otimes Q_n)W^\dagger$. It implies that the premise of the While1 rule holds for P_n. Finally, by the derived **assert** rule, we have the result after line 11. Evidence of bug can be obtained by, e.g., choosing $N = 2$ and $R = |0\rangle\langle 0|$ such that the result predicate is not $\mathbf{0}$.

In our experiments using the static analyzer, we found the bound $N = \dim(\mathcal{H})$ are merely reached when applying the bounded **while**-rule. With RUS circuits consisting of more than four qubits, the post predicate of the while loop converges after at most two unrolling steps when random Pauli-operations are inserted into the loop body. One may notice that inserting assertions in the while loop would be more effective in finding bugs. For example, adding **assert**$(q_0, |0\rangle\langle 0|)$ after line 2 would immediately capture the bug in Fig. 12b with postcondition $[er{:}|1\rangle\langle 1| \otimes Q_0]$, indicating an incorrect initial value of the RUS circuit.

8 DISCUSSION ON ALTERNATIVE VALIDITY FORMULATIONS

In this section, we discuss other possible characterization of errors and validity formulations that we have come up with during the development of QIL, from which we found the Def. 5.2 is the best in expressiveness and efficiency. We use different subscripts of $\vDash_{\cdot}$ to distinguish between different definitions of validity. All these triples are only discussed in this section to avoid confusion, readers not interested in alternative validities may safely skip this section.

8.1 A Naive Generalization of IL Validity Formulation

We start with the following naive formulation.

Definition 8.1 (Classical validity). *A triple is classically valid, denoted by* $\vDash_c [P]S[\epsilon:Q]$, *if*

$$\forall \rho' \vDash Q.\, \exists \rho \vDash P.\, \rho' \in [\![S]\!]_\epsilon \rho.$$

This validity formulation is called "classical" because it is a naive generalization of O'Hearn's incorrectness triple (1), by simply replacing classical states, predicates, and satisfaction with their quantum counterparts. It says that every state ρ' satisfying the projection P is reachable from some state ρ in the projection Q.

Requiring every $\rho' \vDash Q$ being reached is rather restrictive and makes the classical validity too strong to have meaningful triples for initialization statements. For example, starting from a Bell state $\beta_{00} = (|00\rangle + |11\rangle)/\sqrt{2}$, initializing one qubit with $q_0 := |0\rangle$ obtains a mixed state $|0\rangle\langle 0| \otimes \frac{1}{2} I$. Let the presumption $P = \beta_{00}\beta_{00}^\dagger$, the only classically valid triple we can have is a trivial $[P]q_0 := |0\rangle\ [\epsilon:\mathbf{0}]$ because any non-trivial postcondition Q can be satisfied by some unreachable pure state.

8.2 A Weak Validity Based on Observable Relation

The second validity formulation is based on the observable relation, a dual of satisfaction relation for characterizing erroneous states.

Definition 8.2 (Observable relation). *Given a quantum state* $\rho \in \mathcal{D}^-(\mathcal{H})$, *a unit vector* $|\psi\rangle$ *is* observable *within* ρ, *denoted by* $\rho \succ |\psi\rangle$, *if* $\langle\psi|\rho|\psi\rangle > 0$. *A quantum predicate* P *is* observable *within* ρ, *denoted by* $\rho \succ P$, *if* $\mathrm{Tr}(P\rho) > 0$.

Equivalently, $\rho \succ P$ if there is some $|\psi\rangle \in P$ such that $\langle\psi|\rho|\psi\rangle > 0$. It means it is possible to observe some $|\psi\rangle \in P$ in the following sense: when measuring the state ρ with the measurement $M = \{|\psi\rangle\langle\psi|, I - |\psi\rangle\langle\psi|\}$, we are able to obtain $|\psi\rangle\langle\psi|$ with a non-zero probability $\langle\psi|\rho|\psi\rangle$. The observable relation is dual to the satisfaction relation:

$$\rho \nvDash P \Leftrightarrow \rho \succ \neg P. \tag{3}$$

The equation (3) is a generalization of its classical counterpart (2) by observing

$$\sigma \vDash \neg p \ \text{ iff } \ \mathrm{Tr}((\neg p)\sigma) > 0,$$

where the set $\neg p$ is interpreted as to its corresponding boolean-valued characterization function, for unit vectors, the observable relation $\rho \succ |\psi\rangle$ degenerates to equality if we replace ρ and $|\psi\rangle$ with classical states.

The relation between observable relation and under-approximation is stated in Lemma 8.1.

Lemma 8.1 (Relation between observable and under-approximation). *For any projection* P *and quantum state* $\rho \in \mathcal{D}^-(\mathcal{H})$, *we have*

$$\rho \dashv P \implies \forall |\psi\rangle \in P.\, \rho \succ |\psi\rangle \qquad \textit{and} \qquad \rho \succ P \implies \exists P'.\, \rho \dashv P' \wedge \mathrm{Tr}(PP') > 0$$

On top of the observable relation, we give another generalization of the classical incorrectness triple that is better than Def. 8.1.

Definition 8.3 (Weak validity). *A QIL triple is weakly valid, denoted by* $\vDash_w [P]S[\epsilon:Q]$*, if*

$$\forall |\psi'\rangle \in Q. \exists |\psi\rangle \in P, \rho' \succ |\psi'\rangle . \langle S, |\psi\rangle\langle\psi|\rangle \xrightarrow{\epsilon}{}^{*} \langle \downarrow, \rho'\rangle.$$

The formula $\vDash_w [P]S[\epsilon:Q]$ means every $|\psi'\rangle \in Q$ is observable from some reachable state ρ' that comes from some pure state $|\psi\rangle \in P$. One nice thing about this validity is that it may degenerate to classical settings. Recall that the classical states correspond to the computational basis of a Hilbert space. When $|\psi'\rangle$ and ρ' are restricted to classical states, the observable relation $\rho' \succ |\psi'\rangle$ degenerates into equality $\rho' = |\psi'\rangle\langle\psi'|$, and this validity degenerates to that of IL triples.

The connection between strong and weak validities is characterized by Lemma 8.2.

Lemma 8.2 (Connection between weak and strong validities). *For any quantum program S, projections P, Q and exit condition ϵ,*

$$\vDash [P]S[\epsilon:Q] \implies \vDash_w [P]S[\epsilon:Q].$$

Conversely, given P and S, we have

$$(\exists Q \neq \mathbf{0}. \vDash_w [P]S[\epsilon:Q]) \implies (\exists Q' \neq \mathbf{0}. \vDash [P]S[\epsilon:Q'] \wedge \mathrm{Tr}(QQ') > 0) .$$

This lemma implies that when we can observe an *er*/*ok* exit condition with the weak validity, we can find a more proper non-trivial post predicate that satisfies the strong validity. It means if we only care about whether an *er* would occur, the weak validity and strong validity are equivalent, so we call Lemma 8.2 a *weak equivalence* between the strong and weak validities.

The main drawback of this validity formulation is that it is too weak to reject useless triples. Recall the program S_{Bell} in Fig. 1a, the weak validity accept $[|00\rangle\langle 00|]S_{\mathrm{Bell}}[ok:\mathrm{span}\{(\alpha|00\rangle + \beta|11\rangle)\}]$ for any $\alpha^2 + \beta^2 = 1$, which is not informative. This validity formulation is weak because the observable relation is strictly weaker than the under-approximation relation, and $\rho \succ P$ is not accurate enough if P has elements out of $\mathrm{supp}(\rho)$, i.e., $P \not\subseteq \mathrm{supp}(\rho)$.

8.3 A Strict Validity Based on Under-Approximation

Based on the under-approximation relation, we obtain another validity formulation weaker than the classical validity and stronger than the weak validity.

Definition 8.4 (Strict validity). *A triple is strictly valid, denoted by* $\vDash_s [P]S[\epsilon:Q]$*, if*

$$\forall \rho \dashv P. \exists \rho'. \rho' \in [\![S]\!]_\epsilon \rho \wedge \rho' \dashv Q$$

The meaning of this validity is straightforward: for any ρ under-approximated by P, there exits an execution path of S starting from ρ and terminating in ρ' with exit condition ϵ such that Q under-approximates ρ'. In other words, it says if a state is under-approximated by P, then there is a reachable state under-approximated by Q.

With the strict validity, we can give meaningful triples for initialization statements that classical validity can not handle and avoid accepting useless, weakly valid triples. For example, we have $\vDash_s [\beta_{00}\beta_{00}^\dagger]q_0 := |0\rangle\ [ok:|0\rangle\langle 0| \otimes I]$ for the initialization on a Bell state β_{00}. For S_{Bell}, we have only two non-trivial meaningful triples $\vDash_s [|00\rangle\langle 00|]S_{\mathrm{Bell}}[ok:|00\rangle\langle 00|]$ and $\vDash_s [|00\rangle\langle 00|]S_{\mathrm{Bell}}[ok:|11\rangle\langle 11|]$ for the program S_{Bell} in Fig. 1a.

The main drawback of the strict validity is too restrictive to preserve the disjunction rule below, thus being not efficiently reasoned about.

$$\frac{[P]S[\epsilon:Q_1] \quad [P]S[\epsilon:Q_2]}{[P]S[\epsilon:Q_1 \vee Q_2]} .$$

A simple counter example would be letting $P = \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)^\dagger$ and S being the program **if** $(M[q]) =$ **true** $\rightarrow$ **skip** $\square$ **false** $\rightarrow$ **skip**) **fi**. We have two strictly valid triples $[P]S[ok:|0\rangle\langle 0|]$

and $[P]S[ok{:}|1\rangle\langle 1|]$ for this example. The disjunction of postconditions $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$ is I, and a state $\rho' \sqsupseteq I$ must be a mixed state, which is impossible to reach via a single execution path starting with the input state P (also a density matrix). For the same season, the classical validity does not have the disjunction either.

The disjunction rule is crucial for efficient incorrectness reasoning. Without the disjunction rule, one will have to remember the post predicates of all paths when reasoning about a program or information about some paths. Since the number of paths grows exponentially with increasing sequenced **if** statements, the disjunction rule is highly desirable to make the reasoning efficient and thorough (covering as many paths as possible). As we have seen in Sec. 6, the strong validity in Def. 5.2 supports the disjunction rule, thus is more efficient compared with other triples.

At the end of this section, we characterize the relationship between these validity formulations in the observation below, where $\vDash [P]S[\epsilon{:}Q]$ and $\vDash_w [P]S[\epsilon{:}Q]$ are strictly weaker than the others, thus the most expressive.

Observation 8.1. *For arbitrary program S, we have*

$$\vDash_c [P]S[\epsilon{:}Q] \implies \vDash_s [P]S[\epsilon{:}Q] \implies \vDash [P]S[\epsilon{:}Q] \approx \vDash_w [P]S[\epsilon{:}Q],$$

where $\approx$ means the weak equivalence described by Lemma 8.2. In particular, if S does not contain initialization statements $q := |0\rangle$, we have

$$\vDash_c [P]S[\epsilon{:}Q] \iff \vDash_s [P]S[\epsilon{:}Q],$$

but these two validities are still strictly stronger than $\vDash [P]S[\epsilon{:}Q]$.

9 RELATED WORKS

With the rapid progress of quantum hardware, significant efforts are devoted to the research of quantum programming languages [Abhari et al. 2012; Aleksandrowicz et al. 2019; Developers 2021; Green et al. 2013; Smith et al. 2016; Svore et al. 2018], quantum program logic [Barthe et al. 2019; Hung et al. 2019; Kakutani 2009; Ying 2012, 2016; Ying et al. 2009; Yu 2019; Yu and Ying 2012], quantum program verification [Paykin et al. 2017; Rand et al. 2018; Yu and Palsberg 2021] and debugging [Huang and Martonosi 2019b; Li et al. 2020; Li and Ying 2014]. We discuss two lines of research that are directly related to our work, namely the projection-based quantum program logic and incorrectness logic.

Projection based quantum program logic. Recently, projections were used as predicates to develop quantum Hoare logic for reasoning about the correctness of quantum programs in [Zhou et al. 2019], and quantum relational Hoare logics (qRHL) for reasoning about equivalence between two quantum programs [Barthe et al. 2019; Unruh 2019]. Compared with other quantum predicates such as observables [D'hondt and Panangaden 2006], subspaces can significantly simplify the verification of quantum programs and are much more convenient when debugging and testing.

Despite the purpose of the logic, one difference between our logic and the previous results can be explained by quoting from [O'Hearn 2019] "Incorrectness logic uses Floyd's forward-running assignment axiom rather than Hoare's backward-running one." The underlying concepts also differ: while correctness logics use satisfaction to rule out bugs, we use a quantum version of under-approximation to capture bugs. Another difference lies in the reasoning of **while**-loops: inference of loop variants can be automated in our logic, but it is not apparent how to infer the loop invariants in quantum correctness logics.

Incorrectness logic and debugging quantum programs. The incorrectness logic [O'Hearn 2019] for classical programs mainly inspires our paper. We integrate the spirit of classical incorrectness logic of [O'Hearn 2019] and generalize to the quantum settings. This result is partly inspired by the use

of projections as assertions for testing and debugging quantum programs [Li et al. 2020]. There are other practical debugging tools for quantum programs, such as the one employing assertions based on statistical tests on classical observations [Huang and Martonosi 2019b]. These works are designed for debugging at run-time, while our logic enables static analysis. In addition, our logic is sound and complete, but few if any of the earlier works on debugging quantum programs are accompanied by sound arguments, let alone completeness.

10 CONCLUSION AND FUTURE WORK

In this paper, we take an initial step towards an incorrectness logic for quantum programs. We develop the incorrectness triple for quantum programs by introducing new notions of under-approximation and reachability analysis. These notions are fundamentally different from their classical counterparts. Built on the incorrectness triple, we propose an incorrectness logic that proves to be sound, complete, and decidable. We also demonstrate how to use our logic for reasoning about quantum programs by three examples–Grover search, quantum teleportation, and repeat until success. We have already implemented a static analyzer based on our incorrectness logic and successfully applied it to analyzing the program examples in Sec. 7. We hope our work will enrich the future development of static analysis for quantum programs.

Assertion language for quantum predicates. Currently, we treat the predicates in our logic semantically, i.e., writing matrices explicitly. It is possible to introduce an assertion language for predicates to capture the properties of a practical subset of quantum applications to help simplify the representation of predicates. A syntactical predicate may also expose more mathematical structures, which may help automate the inference procedure using logic.

Supporting local reasoning. Readers may notice that the Unitary rule effectively computes the strongest postcondition, which requires multiplicating matrices of size exponential in the number of qubits. One possible way to avoid such full-blown quantum simulation is local reasoning. A promising idea is to combine the recently developed quantum separation logic [Zhou et al. 2021] and incorrectness separation logic [Raad et al. 2020] for classical programs, and determine the extent to which local reasoning is feasible for quantum programs from the incorrect point of view. Compared with classical separation logics, the main challenge is how to deal with the entanglement between subsystems, which is a unique phenomenon in quantum programs. Under certain conditions, one possibility is a frame rule below.

$$\frac{\vdash [P]S[\epsilon : Q]}{\vdash [P \otimes R]S[\epsilon : Q \otimes R]}$$

When the state is not a product state, it is not obvious if a similar frame rule exists.

Supporting quantum noise and quantum control. We do not mention the quantum noise in this paper. It would be more practical to incorporate noise estimation such as Gleipnir [Tao et al. 2021] and [Hung et al. 2019] into the incorrectness logic. Another limitation is that we consider quantum programs with classical controls rather than quantum ones. The semantics for quantum controls would be more complex and fuzzy, and we need to build a new explanation of semantics and quantum predicates to establish any practical proof rules.

ACKNOWLEDGMENTS

We thank the anonymous referees for their suggestions and comments on earlier versions of this paper. This work is supported by ARC Discovery Program (#DP210102449) and ARC DECRA (#DE180100156).

REFERENCES

Ali J Abhari, Arvin Faruque, Mohammad J Dousti, Lukas Svec, Oana Catu, Amlan Chakrabati, Chen-Fu Chiang, Seth Vanderwilt, John Black, and Fred Chong. 2012. *Scaffold: Quantum programming language.* Technical Report. Princeton Univ NJ Dept of Computer Science.

Gadi Aleksandrowicz et al. 2019. Qiskit: An Open-source Framework for Quantum Computing. https://doi.org/10.5281/zenodo.2562111

Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510. https://doi.org/10.1038/s41586-019-1666-5

Gilles Barthe, Justin Hsu, Mingsheng Ying, Nengkun Yu, and Li Zhou. 2019. Relational Proofs for Quantum Programs. *Proc. ACM Program. Lang.* 4, POPL, Article 21 (dec 2019), 29 pages. https://doi.org/10.1145/3371089

Garrett Birkhoff and John Von Neumann. 1936. The Logic of Quantum Mechanics. *Annals of Mathematics* 37, 4 (1936), 823–843. http://www.jstor.org/stable/1968621

Sam Blackshear, Nikos Gorogiannis, Peter W. O'Hearn, and Ilya Sergey. 2018. RacerD: Compositional Static Race Detection. *Proc. ACM Program. Lang.* 2, OOPSLA, Article 144 (oct 2018), 28 pages. https://doi.org/10.1145/3276514

Alex Bocharov, Martin Roetteler, and Krysta M. Svore. 2015. Efficient Synthesis of Universal Repeat-Until-Success Quantum Circuits. *Phys. Rev. Lett.* 114 (Feb 2015), 080502. Issue 8. https://doi.org/10.1103/PhysRevLett.114.080502

Andrew Cross, Ali Javadi-Abhari, Thomas Alexander, Niel de Beaudrap, Lev S. Bishop, Steven Heidel, Colm A. Ryan, Prasahnt Sivarajah, John Smolin, Jay M. Gambetta, and Blake R. Johnson. 2021. OpenQASM 3: A Broader and Deeper Quantum Assembly Language. *ACM Transactions on Quantum Computing* (dec 2021). https://doi.org/10.1145/3505636 Just Accepted.

Cirq Developers. 2021. *Cirq.* https://doi.org/10.5281/zenodo.5182845 See full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors.

Ellie D'hondt and Prakash Panangaden. 2006. Quantum Weakest Preconditions. *Mathematical. Structures in Comp. Sci.* 16, 3 (jun 2006), 429–451. https://doi.org/10.1017/S0960129506005251

Dino Distefano, Manuel Fähndrich, Francesco Logozzo, and Peter W. O'Hearn. 2019. Scaling Static Analyses at Facebook. *Commun. ACM* 62, 8 (jul 2019), 62–70. https://doi.org/10.1145/3338112

Nikos Gorogiannis, Peter W. O'Hearn, and Ilya Sergey. 2019. A True Positives Theorem for a Static Race Detector. *Proc. ACM Program. Lang.* 3, POPL, Article 57 (jan 2019), 29 pages. https://doi.org/10.1145/3290370

Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. 2013. Quipper: A Scalable Quantum Programming Language. *SIGPLAN Not.* 48, 6 (jun 2013), 333–342. https://doi.org/10.1145/2499370.2462177

Lov K. Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (Philadelphia, Pennsylvania, USA) *(STOC '96)*. Association for Computing Machinery, New York, NY, USA, 212–219. https://doi.org/10.1145/237814.237866

Yipeng Huang and Margaret Martonosi. 2019a. QDB: From Quantum Algorithms Towards Correct Quantum Programs. In *9th Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU 2018) (OpenAccess Series in Informatics (OASIcs), Vol. 67)*, Titus Barik, Joshua Sunshine, and Sarah Chasins (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 4:1–4:14. https://doi.org/10.4230/OASIcs.PLATEAU.2018.4

Yipeng Huang and Margaret Martonosi. 2019b. Statistical Assertions for Validating Patterns and Finding Bugs in Quantum Programs. In *Proceedings of the 46th International Symposium on Computer Architecture* (Phoenix, Arizona) *(ISCA '19)*. Association for Computing Machinery, New York, NY, USA, 541–553. https://doi.org/10.1145/3307650.3322213

Shih-Han Hung, Kesha Hietala, Shaopeng Zhu, Mingsheng Ying, Michael Hicks, and Xiaodi Wu. 2019. Quantitative robustness analysis of quantum programs. *Proc. ACM Program. Lang.* 3, POPL (2019), 31:1–31:29. https://doi.org/10.1145/3290344

Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T. Chong, and Margaret Martonosi. 2015. ScaffCC: Scalable compilation and analysis of quantum programs. *Parallel Comput.* 45, C (jun 2015), 2–17. https://doi.org/10.1016/j.parco.2014.12.001

Yoshihiko Kakutani. 2009. A Logic for Formal Verification of Quantum Programs. In *Proceedings of the 13th Asian Conference on Advances in Computer Science: Information Security and Privacy* (Seoul, Korea) *(ASIAN'09)*. Springer-Verlag, Berlin, Heidelberg, 79–93. https://doi.org/10.1007/978-3-642-10622-4_7

Gushu Li, Li Zhou, Nengkun Yu, Yufei Ding, Mingsheng Ying, and Yuan Xie. 2020. Projection-Based Runtime Assertions for Testing and Debugging Quantum Programs. *Proc. ACM Program. Lang.* 4, OOPSLA, Article 150 (nov 2020), 29 pages. https://doi.org/10.1145/3428218

Yangjia Li and Mingsheng Ying. 2014. Debugging quantum processes using monitoring measurements. *Phys. Rev. A* 89 (Apr 2014), 042338. Issue 4. https://doi.org/10.1103/PhysRevA.89.042338

Ji Liu, Gregory T. Byrd, and Huiyang Zhou. 2020. *Quantum Circuits for Dynamic Runtime Assertions in Quantum Computation.* Association for Computing Machinery, New York, NY, USA, 1017–1030. https://doi.org/10.1145/3373376.3378488

Peter W. O'Hearn. 2019. Incorrectness Logic. *Proc. ACM Program. Lang.* 4, POPL, Article 10 (dec 2019), 32 pages. https://doi.org/10.1145/3371078

Adam Paetznick and Krysta M. Svore. 2014. Repeat-until-Success: Non-Deterministic Decomposition of Single-Qubit Unitaries. *Quantum Info. Comput.* 14, 15–16 (nov 2014), 1277–1301. https://doi.org/10.26421/QIC14.15-16-2

Jennifer Paykin, Robert Rand, and Steve Zdancewic. 2017. QWIRE: A Core Language for Quantum Circuits. *SIGPLAN Not.* 52, 1 (Jan 2017), 846–858. https://doi.org/10.1145/3093333.3009894

Simon Perdrix. 2008a. A Hierarchy of Quantum Semantics. *Electron. Notes Theor. Comput. Sci.* 192, 3 (2008), 71–83. https://doi.org/10.1016/j.entcs.2008.10.028

Simon Perdrix. 2008b. Quantum Entanglement Analysis Based on Abstract Interpretation. In *Proceedings of the 15th International Symposium on Static Analysis* (Valencia, Spain) *(SAS '08, Vol. 5079)*. Springer-Verlag, Berlin, Heidelberg, 270–282. https://doi.org/10.1007/978-3-540-69166-2_18

Azalea Raad, Josh Berdine, Hoang-Hai Dang, Derek Dreyer, Peter O'Hearn, and Jules Villard. 2020. Local Reasoning About the Presence of Bugs: Incorrectness Separation Logic. In *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part II* (Los Angeles, CA, USA). Springer-Verlag, Berlin, Heidelberg, 225–252. https://doi.org/10.1007/978-3-030-53291-8_14

Robert Rand, Jennifer Paykin, and Steve Zdancewic. 2018. QWIRE Practice: Formal Verification of Quantum Circuits in Coq. *Electronic Proceedings in Theoretical Computer Science* 266 (Feb 2018), 119–132. https://doi.org/10.4204/eptcs.266.8

Peter Selinger. 2004. Towards a Quantum Programming Language. *Mathematical. Structures in Comp. Sci.* 14, 4 (aug 2004), 527–586. https://doi.org/10.1017/S0960129504004256

Robert S. Smith, Michael J. Curtis, and William J. Zeng. 2016. A Practical Quantum Instruction Set Architecture. arXiv:1608.03355 [quant-ph]

Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. 2018. Q#: Enabling Scalable Quantum Computing and Development with a High-Level DSL. In *Proceedings of the Real World Domain Specific Languages Workshop 2018* (Vienna, Austria) *(RWDSL2018)*. Association for Computing Machinery, New York, NY, USA, Article 7, 10 pages. https://doi.org/10.1145/3183895.3183901

Runzhou Tao, Yunong Shi, Jianan Yao, John Hui, Frederic T. Chong, and Ronghui Gu. 2021. Gleipnir: Toward Practical Error Analysis for Quantum Programs. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation* (Virtual, Canada) *(PLDI 2021)*. Association for Computing Machinery, New York, NY, USA, 48–64. https://doi.org/10.1145/3453483.3454029

Dominique Unruh. 2019. Quantum Relational Hoare Logic. *Proc. ACM Program. Lang.* 3, POPL, Article 33 (jan 2019), 31 pages. https://doi.org/10.1145/3290346

Peng Yan, Hanru Jiang, and Nengkun Yu. 2022. On Incorrectness Logic for Quantum Programs (Technical Report). https://hrjiang.github.io/ilq/

Mingsheng Ying. 2012. Floyd–Hoare Logic for Quantum Programs. *ACM Trans. Program. Lang. Syst.* 33, 6, Article 19 (Jan. 2012), 49 pages. https://doi.org/10.1145/2049706.2049708

Mingsheng Ying. 2016. *Foundations of Quantum Programming* (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. https://doi.org/10.1016/C2014-0-02660-3

Mingsheng Ying, Runyao Duan, Yuan Feng, and Zhengfeng Ji. 2009. *Predicate Transformer Semantics of Quantum Programs*. Cambridge University Press, Cambridge, 311–360. https://doi.org/10.1017/CBO9781139193313.009

Nengkun Yu. 2019. Quantum Temporal Logic. https://doi.org/10.48550/arXiv.1908.00158

Nengkun Yu and Jens Palsberg. 2021. Quantum Abstract Interpretation. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation* (Virtual, Canada) *(PLDI 2021)*. Association for Computing Machinery, New York, NY, USA, 542–558. https://doi.org/10.1145/3453483.3454061

Nengkun Yu and Mingsheng Ying. 2012. Reachability and Termination Analysis of Concurrent Quantum Programs. In *Proceedings of the 23rd International Conference on Concurrency Theory* (Newcastle upon Tyne, UK) *(CONCUR'12)*. Springer-Verlag, Berlin, Heidelberg, 69–83. https://doi.org/10.1007/978-3-642-32940-1_7

Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, et al. 2020. Quantum computational advantage using photons. *Science* 370, 6523 (2020), 1460–1463. https://doi.org/10.1126/science.abe8770

Li Zhou, Gilles Barthe, Justin Hsu, Mingsheng Ying, and Nengkun Yu. 2021. A Quantum Interpretation of Bunched Logic & Quantum Separation Logic. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '21)*. IEEE Computer Society, Los Alamitos, CA, USA, 1–14. https://doi.org/10.1109/LICS52264.2021.9470673

Li Zhou, Nengkun Yu, and Mingsheng Ying. 2019. An Applied Quantum Hoare Logic. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Phoenix, AZ, USA) *(PLDI 2019)*. Association for Computing Machinery, New York, NY, USA, 1149–1162. https://doi.org/10.1145/3314221.3314584