Approximate Relational Reasoning for Quantum Programs

Peng Yan[1], Hanru Jiang[2], and Nengkun Yu[3(✉)]

[1] University of Technology Sydney, Sydney, Australia
pengyan.edu@gmail.com
[2] Beijing Institute of Mathematical Sciences and Applications, Beijing, China
hanru@bimsa.cn
[3] Stony Brook University: Stony Brook, New York, USA
nengkunyu@gmail.com

Abstract. Quantum computation is inevitably subject to imperfections in its implementation. These imperfections arise from various sources, including environmental noise at the hardware level and the introduction of approximate implementations by quantum algorithm designers, such as lower-depth computations. Given the significant advantage of relational logic in program reasoning and the importance of assessing the robustness of quantum programs between their ideal specifications and imperfect implementations, we design a proof system to verify the approximate relational properties of quantum programs. We demonstrate the effectiveness of our approach by providing the first formal verification of the renowned low-depth approximation of the quantum Fourier transform. Furthermore, we validate the approximate correctness of the repeat-until-success algorithm. From the technical point of view, we develop approximate quantum coupling as a fundamental tool to study approximate relational reasoning for quantum programs, a novel generalization of the widely used approximate probabilistic coupling in probabilistic programs, answering a previously posed open question for projective predicates.

Keywords: Relational Hoare Logic · Approximating Reasoning · Quantum Programming Languages

1 Introduction

Program equivalence [11,18,41] is a central concept in many areas of computer science, including software engineering [31,36,54], translation validation of compilers [38], program optimization [30], and program analysis [6,15,35]. Relational verification aims to prove the relational properties between two programs. A typical Hoare-style relational judgment is of the form $\vdash c_1 \sim c_2 : \Psi \Rightarrow \Phi$ where c_1 and c_2 represent two compared programs, Ψ and Φ are relational assertions in the deterministic scenario [10], where relational Hoare logic (RHL) predicates are

A. Gurfinkel and V. Ganesh (Eds.): CAV 2024, LNCS 14683, pp. 495–519, 2024.
https://doi.org/10.1007/978-3-031-65633-0_22

binary relations over memories. The judgment states that for any initial memories m_1 and m_2 that satisfy the precondition Ψ, the resulting memories m_1' and m_2' should satisfy postcondition Φ. For probabilistic programs [7], probabilistic relational Hoare logic (pRHL) lifted the predicates into relations over probabilistic distributions on memories. Furthermore, [9] introduced extra parameters to allow approximate lifting of relations to distributions. To be specific, the judgments defined in approximate probabilistic relational Hoare logic (apRHL) are of the form $c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$ with parameters α, δ for reasoning about differential privacy.

Since the emergence of quantum programming languages, there have been various works [25,27,32,44,57,59–62] about the formal verification of quantum programs. Among techniques in program analysis, the *exact* relational logic for quantum programs attracts lots of attention [8,33,51]. Relational logic provides a more expressive approach to characterize the relation between two programs. For instance, direct verification of the equivalence between quantum programs S_1 and S_2 defined on register $\bar{q}$ requires checking the equivalence between $[\![S_1]\!](\rho)$ and $[\![S_2]\!](\rho)$ for all ρ in Hilbert space $\mathcal{H}_{\bar{q}}$ that involves enumerations of an infinite set. A quantum relational judgment concerning the quantum equivalence predicate can concisely explain the direct enumerations. However, none of the above works considers approximate reasoning that is universal in practice.

- It is implausible to physically implement quantum gates with perfect accuracy on the hardware level, and the need to consider approximations is likely inevitable. As noted by John Preskill, the noise in quantum gates will limit the size of reliable quantum circuits, and technologies for more accurate quantum gates are of great value in the Noisy Intermediate-Scale Quantum (NISQ) [42] era.
- On the software level, the NISQ nature of hardware signifies the importance of taking noise into account at the level of quantum algorithm design. More specifically, approximate computation can be more efficient and less erroneous than precise one since it can improve the depth of circuits and simplify the calculation. A good example is the approximate quantum Fourier transform [16], which achieves a lower circuit depth approximation of the exact quantum Fourier transform used in Shor's celebrated algorithm [45].

As for approximate reasoning in quantum settings, [66] discussed the robustness of quantum programs by introducing the concept of approximate satisfaction of predicates, [26] proposed a parameterized diamond norm to characterize the distance between an ideal program and a noisy one. Despite the significant advancements in quantum approximate reasoning and the recognition of the importance of relational reasoning, there remains a notable gap in the field — an absence of a robust logical framework for effectively reasoning about the relational properties between quantum programs approximately. In quantum approximate relational reasoning, the main obstacles are:

- There is no mathematical theory for a quantum version of approximate couplings, an open question in [8]. The lack of such a theory significantly affects the applications of exact quantum coupling and relations quantum Hoare

logic. Usually, two quantum programs have different branching probabilities in the presence of noise or approximations. Under these circumstances, their corresponding quantum states have different traces, where *exact* quantum couplings on these states do not exist. The main difficulties in defining an approximate quantum coupling include defining a distance between quantum states, which can be highly nonlinear. Previous knowledge about probabilistic couplings may not directly apply: even for the exact quantum coupling, fundamental properties of probabilistic coupling [24] are no longer true [67].
- Designing an approximate relational quantum Hoare logic system is indeed highly challenging. The system needs to consider several factors, including infinite executions of quantum while loops, approximated quantum couplings, and the applicability of the logic rules. In quantum programming, a while loop can have infinite executions of the loop body because of the probabilistic feature of quantum measurements. Furthermore, when dealing with approximate quantum couplings, the system must handle the inherent uncertainty and approximation errors that arise when coupling with program branches.
- The applicability of logic rules adds another layer of complexity. To strike a balance between the accuracy of the logic rules and simplicity, efficiency, and usability is a crucial consideration when designing a logic system. Ensuring the logic rules are powerful yet easy to use for reasoning relational properties of complicated quantum programs requires careful consideration and analysis.

In this paper, we derive an approximate version of the existing quantum relational Hoare logic, thus making approximate relational reasoning feasible. Our judgment is of the form

$$S_1 \sim_\delta S_2 : A \Rightarrow B$$

where S_1 and S_2 represent compared quantum programs, A and B are projective quantum predicates over the whole system. The validity of our judgment is based on the idea of approximate (quantum) coupling and lifting. A state ρ is a δ-coupling for the state pair $\langle \rho_1, \rho_2 \rangle$ if trace distances $D(\rho_1, \mathrm{Tr}_2(\rho))$ and $D(\rho_2, \mathrm{Tr}_1(\rho))$ are both not bigger than δ. A state σ is a witness of the δ-lifting $\rho_1 \sim_P^\delta \rho_2$ if σ is a δ-coupling for the state pair $\langle \rho_1, \rho_2 \rangle$ and satisfies the predicate P ($P\sigma = \sigma$). Informally, our judgment holds if for any quantum lifting $\rho_1 \sim_A^0 \rho_2$ of the inputs, there exists a witness of the δ-lifting $[\![S_1]\!](\rho_1) \sim_P^\delta [\![S_2]\!](\rho_2)$ of the outputs.[1]

Technical contributions include:

- *Approximate quantum liftings.* We propose a novel notion of approximate quantum liftings concerning projection-based quantum predicates to make approximate reasoning simple and powerful. We do not require two quantum states to have the same trace in approximate quantum lifting. In other words, the exact quantum coupling may not exist. We employ the existing distances, including trace distance and diamond norm, and define a "Hausdorff-like"

[1] See Sect. 2 and Sect. 5 for a detailed definition.

distance between projections incorporated with quantum coupling to be the metric of the approximation of the couplings.
- *Sound aqRHL.* We propose a formal relational judgment to incorporate the spirit of classical apRHL with a new quantum explanation based on the proposed approximate quantum liftings. A sound approximate quantum relational Hoare logic (aqRHL) is built based on our relational judgments. Our choice of quantum δ-lifting allows us to track the relational properties of two programs with different classical branching probabilities. In particular, our methodology allows us to compute proper upper bounds for approximate liftings for quantum equivalence relations, which plays a central role in characterizing the equivalence of quantum programs.
- *Application.* We demonstrate the first formal verification of the low-depth approximate quantum Fourier transform (QFT) with an error bound that is asymptotically equivalent to the one in [16]. Implementing QFT is a significant step in the development of quantum algorithms such as period finding [45], HHL algorithm [21] and quantum principal component analysis [34]. We also apply aqRHL, particularly the loop rule, to reason the repeat until success which is one of the essential loop programs in quantum computation. Other applications covered in the complete edition of this paper include the verification of appropriate decomposition of unitary gates, and the correctness of bit flip code against an arbitrary single-qubit error.

2 Preliminary and Notations

This section offers a brief introduction to quantum computation and necessary notations from [39].

The state space of a quantum system is a Hilbert space $\mathcal{H}$. The Dirac notation $|\psi\rangle$ denotes a unit complex vector (called *pure state* or *vector state*). The most important orthonormal basis of one-qubit system is the *computational basis*, i.e. $\{|0\rangle, |1\rangle\}$. Superposition is a key feature that makes quantum programs different from classical ones, such as a qubit being in the superposition $(|0\rangle \pm |1\rangle)/\sqrt{2}$. An operator acting on an d-dimensional Hilbert space $\mathcal{H}$ is represented as a $d \times d$ matrix. A positive semi-definite, Hermitian operator, ρ acting on $\mathcal{H}$, is called a *partial density operator* if its trace satisfies $\mathrm{Tr}(\rho) \leq 1$. Particularly, ρ is called a *density operator* if $\mathrm{Tr}(\rho) = 1$. The partial density operators can represent both pure and *mixed* quantum states. For a pure state $|\psi\rangle$, its partial density operator is $|\psi\rangle\langle\psi|$, where $\langle\psi|$ is the conjugate transpose of $|\psi\rangle$. For a mixed state which is a classical distribution $\{p_i\}$ over pure states $\{|\psi_i\rangle\}$, its partial density operator is $\sum_i p_i |\psi_i\rangle\langle\psi_i|$. We use $\mathcal{D}(\mathcal{H})$ to denote the set of all partial density operators acting on Hilbert space $\mathcal{H}$.

Let $\bar{q}_1$ and $\bar{q}_2$ be two independent registers in states $\rho_1 \in \mathcal{D}(\mathcal{H}_{\bar{q}_1})$ and $\rho_2 \in \mathcal{D}(\mathcal{H}_{\bar{q}_2})$ respectively, the composite register $\bar{q} = \{\bar{q}_1, \bar{q}_2\}$ is then in the state $\rho_1 \otimes \rho_2 \in \mathcal{H}_{\bar{q}} = \mathcal{H}_{\bar{q}_1} \otimes \mathcal{H}_{\bar{q}_2}$. *Partial trace* is a very useful tool for describing subsystems of a composite quantum system. Formally, the partial trace over $\mathcal{H}_{\bar{q}_1}$ is a mapping $\mathrm{Tr}_1(\cdot)$ from operators in $\mathcal{H}_{\bar{q}_1} \otimes \mathcal{H}_{\bar{q}_2}$ to operators in $\mathcal{H}_{\bar{q}_2}$ defined

by $\mathrm{Tr}_1(|\varphi_1\rangle\langle\psi_1| \otimes |\varphi_2\rangle\langle\psi_2|) = \langle\psi_1|\varphi_1\rangle \cdot |\varphi_2\rangle\langle\psi_2|$ for any $|\psi_1\rangle, |\varphi_1\rangle \in \mathcal{H}_{\bar{q}_1}$ and $|\psi_2\rangle, |\varphi_2\rangle \in \mathcal{H}_{\bar{q}_2}$. The partial trace $\mathrm{Tr}_2(\cdot)$ can be defined symmetrically. If the composite system $\bar{q} = \{\bar{q}_1, \bar{q}_2\}$ is in the state ρ, then subsystems $\bar{q}_1$ and $\bar{q}_2$ are in states $\mathrm{Tr}_2(\rho)$ and $\mathrm{Tr}_1(\rho)$ respectively.

The evolution of an isolated quantum system can be characterized by a *unitary operator* U such that $U^\dagger U = UU^\dagger = I$, where $\dagger$ denotes the conjugate and transpose. Here we introduce some commonly used unitary operators, also known as "*gates*", that will be used in later examples:

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

$$P(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \quad CNOT = \begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix} \quad c\text{-}P(\theta) = \begin{pmatrix} I & 0 \\ 0 & P(\theta) \end{pmatrix}$$

The act of extracting information from a quantum system is known as *quantum measurement*. A measurement $\mathcal{M} = \{M_m\}$ is described by a set of linear operators over $\mathcal{H}$ such that $\sum_m M_m^\dagger M_m = I$, where the subscript m refers to the measurement outcome. Applying a quantum measurement $\mathcal{M}$ on $|\psi\rangle$, the probability of observing outcome m is $p_m = \langle\psi|M_m M_m^\dagger|\psi\rangle$, and the state after the measurement collapses into $M_m|\psi\rangle/\sqrt{p_m}$.

A *projection* is a linear operator P on $\mathcal{H}$ that satisfies $P^2 = P = P^\dagger$. This paper adopts the convention from [12] and recent work [51,66] that constrains quantum predicates to be Hilbert spaces or projections. The complete partial order over Hilbert subspaces is equivalent to the inclusion relation $\subseteq$. This choice of predicates enables us to define the assertion about quantum states.

Definition 1 (Support). *If $A = \sum_i \lambda_i |\psi_i\rangle\langle\psi_i|$, where $|\psi_i\rangle$ is an unit vector in $\mathcal{H}$ and $\lambda_i > 0$, then the* support *of A is the space spanned by $\{|\psi_i\rangle\}$. I.e.,* $\mathrm{supp}(A) = \mathrm{span}\{|\psi_i\rangle\}$.

Definition 2 (Satisfaction). *A partial density operator ρ satisfies a predicate P, denoted by $\rho \vDash P$, if* $\mathrm{supp}(\rho) \subseteq P$.

A general quantum operation, described by a superoperator $\mathcal{E}$, can be implemented by combining unitary transformations with quantum measurements by introducing ancilla systems and discarding post-measurement states. A superoperator always maps density matrices to partial density matrices and has the Kraus representation. Readers may refer to the system-environment model in Sect. 8.2 [39] for more details.

3 Quantum Programming Language

In this section, we review the syntax and semantics of the quantum while-language [59]. We use $var(S)$ to represent the set of all variables present in a quantum program S, and $\mathcal{H}_S = \otimes_{q \in var(S)} \mathcal{H}_q$ to denote the Hilbert space of all the quantum variables in program S. The syntax in Definition 3 is the same as [59] except that the conditional statement is replaced by the **if** statement.

Definition 3 (Syntax). *The following syntax defines the quantum programs:*

$$\begin{aligned}(\text{Stmts}) \quad S \ ::= & \ \textbf{skip} \mid q := |0\rangle \mid \bar{q} := U[\bar{q}] \mid S_1; S_2 \\ & \mid \textbf{if } (\square m \cdot \mathcal{M}[\bar{q}] = m \to S_m) \textbf{ fi} \mid \textbf{while } \mathcal{M}[\bar{q}] = 1 \textbf{ do } S \textbf{ od}\end{aligned}$$

The denotational semantics of the quantum while-language are presented in Fig. 1. By convention, we use $[\![S]\!]$ to denote the semantic function of a program S. Statement $q := |0\rangle$ initializes a variable q in state ρ to $|0\rangle\langle 0|$ while leaving other variables unchanged, where $|\psi\rangle_q\langle\varphi|$ denote the outer product of the vector states $|\psi\rangle$ and $|\varphi\rangle$ in $\mathcal{H}_q$. The statement $\bar{q} := U[\bar{q}]$ performs the unitary transition $\rho \mapsto U\rho U^\dagger$ over register $\bar{q}$. Quantum measurements work as guards to set a variable in a mixed state. For the if statement, if the measurement outcome is m, the input state ρ will collapse into $M_m\rho M_m^\dagger/p_m$ with probability $p_m = \mathrm{Tr}(M_m\rho M_m^\dagger)$ and then executes subprogram S_m. Here we absorb the probability p_m into the collapse state, and $M_m\rho M_m^\dagger$ represents the corresponding measurement output. The final output is the summation of the outputs of all branches. For the loop statement, $\mathcal{M}_0 \circ ([\![S]\!] \circ \mathcal{M}_1)^k$ denotes the k-th unrolling of loop statement.

$$[\![\textbf{skip}]\!](\rho) = \rho \qquad [\![q := |0\rangle]\!](\rho) = \textstyle\sum_n |0\rangle_q\langle n|\rho|n\rangle_q\langle 0|$$
$$[\![\bar{q} := U[\bar{q}]]\!](\rho) = U\rho U^\dagger \qquad [\![S_1; S_2]\!] = [\![S_2]\!] \circ [\![S_1]\!]$$
$$[\![\textbf{if } (\square m \cdot \mathcal{M}[\bar{q}] = m \to S_m) \textbf{ fi}]\!](\rho) = \textstyle\sum_m S_m \circ \mathcal{M}_m = \sum_m [\![S_m]\!](M_m\rho M_m^\dagger)$$
$$[\![\textbf{while } \mathcal{M}[\bar{q}] = 1 \textbf{ do } S \textbf{ od}]\!](\rho) = \textstyle\sum_{k=0}^{\infty} \mathcal{M}_0 \circ ([\![S]\!] \circ \mathcal{M}_1)^k(\rho)$$
$$\mathcal{M}_m(\rho) = M_m\rho M_m^\dagger \quad (\mathcal{R}_1 + \mathcal{R}_2)(\rho) = \mathcal{R}_1(\rho) + \mathcal{R}_2(\rho) \quad \mathcal{R}^0(\rho) = \rho \quad \mathcal{R}^n = \mathcal{R}^{n-1} \circ \mathcal{R}$$
$$\mathcal{R}_2 \circ \mathcal{R}_1(\rho) = \{\rho'' \mid \rho' = \mathcal{R}_1(\rho) \wedge \rho'' = \mathcal{R}_2(\rho') \wedge \rho, \rho', \rho'' \in \mathcal{D}(\mathcal{H})\}$$

Fig. 1. Denotational semantics of quantum while-language

Lemma 1 ([59]). *For any quantum while program S defined in Fig. 1, its denotational semantics function $[\![S]\!] : \mathcal{D}(\mathcal{H}) \mapsto \mathcal{D}(\mathcal{H})$ is a superoperator.*

4 Quantum Approximate Coupling and Liftings

4.1 Approximate Quantum Coupling and Lifting

We first review the quantum generalization of the classical trace distance, a commonly used metric for the difference between two (partial) quantum states.

Definition 4. *The trace distance of two partial density operators ρ and σ is $D(\rho, \sigma) \equiv \frac{1}{2}\mathrm{Tr}|\rho - \sigma|$, where $|A| = \sqrt{A^\dagger A}$ for any operator A, i.e., the positive square root of $A^\dagger A$.*

In the classical setting, two discrete distributions μ_1 and μ_2 over sets A_1 and A_2 are coupled by a distribution μ over $A_1 \times A_2$ if and only if the first and second marginals of μ are exactly μ_1 and μ_2, respectively. This notion of coupling for distributions naturally generalizes to an exact quantum coupling [8] for density matrices. Formally, we say ρ is an exact *coupling* for $\langle \rho_1, \rho_2 \rangle$ if $\mathrm{Tr}_1(\rho) = \rho_2$ and $\mathrm{Tr}_2(\rho) = \rho_1$. Commonly, the quantum measurements in two quantum programs produce different probability distributions. In such cases, *exact* quantum coupling does not exist between branches. We propose approximate quantum coupling parameterized by deviation δ to bound the trace distance between the "marginal" density matrices.

Definition 5 (Approximate Quantum Coupling). *Let $\rho_1 \in \mathcal{D}(\mathcal{H}_1)$ and $\rho_2 \in \mathcal{D}(\mathcal{H}_2)$, then $\rho \in \mathcal{D}(\mathcal{H}_1 \otimes \mathcal{H}_2)$ is a* δ-coupling *for $\langle \rho_1, \rho_2 \rangle$ if*

$$D(\rho_1, \mathrm{Tr}_2(\rho)) \leq \delta \qquad D(\rho_2, \mathrm{Tr}_1(\rho)) \leq \delta$$

The approximate quantum coupling degenerates to the exact version if $\delta = 0$. Like the classical case, approximate quantum coupling induces approximate semantics of projective predicates via approximate lifting.

Definition 6 (Approximate Quantum Lifting). *Let $\rho_1 \in \mathcal{D}(\mathcal{H}_1)$ and $\rho_2 \in \mathcal{D}(\mathcal{H}_2)$, let P be a projection onto a closed subspace of $\mathcal{H}_1 \otimes \mathcal{H}_2$, then $\rho \in D(\mathcal{H}_1 \otimes \mathcal{H}_2)$ is called a witness of the δ-lifting $\rho_1 \sim_P^\delta \rho_2$ if,*

1. *ρ is a δ-coupling for $\langle \rho_1, \rho_2 \rangle$;*
2. *$\mathrm{supp}(\rho) \subseteq P$.*

where δ is the deviation *from the exact quantum lifting.*

A valid approximate quantum lifting implies the existence of an approximate quantum coupling that satisfies a quantum predicate. The approximate lifting $\rho_1 \sim_P^\delta \rho_2$ degenerates into the exact lifting $\rho_1 \sim_P \rho_2$ when $\delta = 0$. One of the most important quantum predicates is the equivalence relation between two registers, as defined below [8].

Definition 7 (Equivalence). *Let register $\bar{p}$ and $\bar{q}$ are two disjoint registers of the same size. The quantum equivalence predicate over $(\bar{p}, \bar{q})$, denoted by $\equiv_{(\bar{p},\bar{q})}$, is the projection*

$$(I_{\bar{p}} \otimes I_{\bar{q}} + SWAP)/2$$

over subspace $\mathcal{H}_{\bar{p}} \otimes \mathcal{H}_{\bar{q}}$. SWAP is the swap operator defined on $(\bar{p}, \bar{q})$ such that by $SWAP|\psi\rangle|\varphi\rangle = |\varphi\rangle|\psi\rangle$ for any $|\psi\rangle \in \mathcal{H}_{\bar{p}}$ and $|\varphi\rangle \in \mathcal{H}_{\bar{q}}$.

The quantum equivalence predicate in Definition 7 directly comes from a natural observation. In the probabilistic world, if two probability distributions μ_1 and μ_1 over X are the same, then there exists a coupling μ whose support lives in the identity relation $\{(a, a) \mid a \in X\}$. In quantum settings, this is not true due to superposition. For example, the exact coupling of the state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and itself is $|+\rangle \otimes |+\rangle$, which is not in the space spanned by $|0\rangle \otimes |0\rangle$ and $|1\rangle \otimes |1\rangle$.

Instead, we use the projection $(I + SWAP)/2$ to represent the corresponding symmetric space. By doing so, we have $(I + SWAP)(|+\rangle \otimes |+\rangle)/2 = |+\rangle \otimes |+\rangle$. The following lemma shows that approximate lifting concerning $\equiv$[2] effectively encodes the trace distance of two partial density matrices.

Lemma 2. *For any ρ_1, ρ_2, we have $\rho_1 \sim_{\equiv}^{\delta} \rho_2 \Leftrightarrow D(\rho_1, \rho_2) \leq 2\delta$. Particularly, if $\delta = 0$, we have $\rho_1 \sim_{\equiv} \rho_2 \Leftrightarrow \rho_1 = \rho_2$.*

The introduction of approximate couplings/liftings is necessary when the comparison can not match the desired predicate exactly. For example, the implementation of a unitary gate U can be approximated by a proper RUS circuit [13,40]. Given an input ρ, quantum measurement in each iteration of the RUS circuit will generate the desired output $U\rho U^\dagger$ with a probability p, where p is determined by the construction of the RUS circuit. If the iterations of the RUS circuit are unbounded, the desired state can eventually be achieved. The RUS algorithm's function $\mathcal{E}$ converges to U, as expressed as $\mathcal{E}(\rho) = \sum_{k=1}^{\infty} p(1-p)^{k-1} U\rho U^\dagger = U\rho U^\dagger$. In this case, the exact lifting can accurately describe the equivalence $\mathcal{E}(\rho) \sim_{\equiv} U\rho U^\dagger$ with no problem. However, in practical scenarios, there typically exists an upper bound N on the iteration count k, leading to an approximate equivalence denoted as $\mathcal{E}'(\rho) \sim_{\equiv}^{\delta} U\rho U^\dagger$, where $\mathcal{E}'$ represents the function with bounded looping and $\delta = (1-p)^N/2$.

4.2 Upper Bound of Approximation

The approximation usually arises when we use a desired postcondition to approximate an exact postcondition. Formally, let (A, B) be the pair of two projections A and B over Hilbert space $\mathcal{H}_1 \otimes \mathcal{H}_2$, the inference

$$\forall \rho_1, \rho_2, \rho_1 \sim_A \rho_2 \;\Rightarrow\; \rho_1 \sim_B^{\delta} \rho_2 \tag{1}$$

demonstrates a general way of introducing approximate reasoning. That is, given a witness of the exact lifting $\rho_1 \sim_A \rho_2$, does there exist a witness σ of the approximate lifting $\rho_1 \sim_B^{\delta_{\rho_2}} ?$. The optimal deviation δ in Eq. 1 is equivalent to the following quantity,

$$\delta = d(A, B) = \sup_{\rho \vDash A} \inf_{\sigma \vDash B} \max\{D(\mathrm{Tr}_1(\rho), \mathrm{Tr}_1(\sigma)), D(\mathrm{Tr}_2(\rho), \mathrm{Tr}_2(\sigma))\} \tag{2}$$

where $d(A, B)$ can be upper bounded by $\sup\limits_{\rho \vDash A} \inf\limits_{\sigma \vDash B} D(\rho, \sigma)$ introduced in [66].

In the following, we discuss a simple but important instance of Eq. 1 with $A = (U_1 \otimes U_2)B(U_1 \otimes U_2)^\dagger$ and B being the quantum equivalence predicate $\equiv$. Then Eq. 1 can be represented as follows,

$$\forall \rho_1, \rho_2, \rho_1 \sim_{\equiv} \rho_2 \;\Rightarrow\; U_1\rho_1 U_1^\dagger \sim_A U_2\rho_2 U_2^\dagger \;\Rightarrow\; U_1\rho_1 U_1^\dagger \sim_{\equiv}^{\delta} U_2\rho_2 U_2^\dagger$$

where δ can be upper bounded by $\|U_1 \cdot U_1^\dagger - U_2 \cdot U_2^\dagger\|_\diamond$. The diamond norm $\|\cdot\|_\diamond$ proposed by Kitaev [2] can better distinguish between two superoperators with the help of the power of quantum entanglement by introducing auxiliary qubits.

[2] The subscripts can be ignored without confusion.

Definition 8 (Diamond Norm). *Let $\mathcal{A} : \mathcal{L}(\mathcal{H}) \mapsto \mathcal{L}(\mathcal{H})$ with $\mathcal{L}(\mathcal{H})$ denoting the matrix space of $\mathcal{H}$,*

$$\|\mathcal{A}\|_\diamond \equiv \max_{\rho \in \mathcal{D}(\mathcal{H}\otimes\mathcal{H}')} \frac{1}{2}\mathrm{Tr}|(\mathcal{A}\otimes I_{\mathcal{H}'})(\rho)| \tag{3}$$

where $\mathcal{H}'$ denotes a copy of $\mathcal{H}$. The factor $1/2$ is added to keep consistent with trace distance.

It is straightforward to verify that $D(\mathcal{E}_1(\sigma), \mathcal{E}_2(\sigma)) \leq \|\mathcal{E}_1 - \mathcal{E}_2\|_\diamond$ for any $\sigma \in \mathcal{D}(\mathcal{H})$ for superoperators $\mathcal{E}_1, \mathcal{E}_2$ by choosing $\rho = \sigma \otimes I_{\mathcal{H}'}/2^{Dim(\mathcal{H}')}$.[3] The distance between two superoperators can be computed efficiently [53]. Particularly, the distance between U_1 and U_2 is

$$\|U_1 \cdot U_1^\dagger - U_2 \cdot U_2^\dagger\|_\diamond = \begin{cases} \sin\alpha/2 & \alpha < \pi \\ 1 & \alpha \geq \pi \end{cases} \tag{4}$$

where α is the smallest arc containing the spectrum of $U_1^\dagger U_2$ [37].

5 Approximate Relational Logic

5.1 Judgment and Validity

Our logic, called aqRHL, "approximates" the quantum relational Hoare logic described in [8]. The judgments in aqRHL take the following form $S_1 \sim_\delta S_2 : A \Rightarrow B$, where S_1 and S_2 are quantum programs, A and B are projections over subspaces of $\mathcal{H}_{\bar{q}_1} \otimes \mathcal{H}_{\bar{q}_2}$ such that $\bar{q}_i$ contains all free variables of S_i, $\delta \in [0, 1/2]$ is referred to as the *deviation* from the exact quantum lifting, respectively. Registers $\bar{q}_1$ and $\bar{q}_2$ are often omitted since they rarely change along our reasoning and are often clear from the context.

Definition 9 (Validity). *The approximate relational judgement $S_1 \sim_\delta S_2 : A \Rightarrow B$ is valid, written as $\vDash S_1 \sim_\delta S_2 : A \Rightarrow B$, if and only if*

$$\forall \rho_1, \rho_2.\, \rho_1 \sim_A \rho_2 \quad \Rightarrow \quad [\![S_1]\!](\rho_1) \sim_B^\delta [\![S_2]\!](\rho_2)$$

where A and B are projections. If the deviation δ equals zero, it will be omitted for simplicity.

In Definition 9, we choose projective predicates [12] over the joint system of two programs because such predicates are the quantum analog of binary relations, the predicates used in pRHL [9]. Moreover, this definition will become a judgment of [8] if $\delta = 0$. One of the most important applications of relational Hoare logic is to verify the equivalence between programs, as presented in the following lemma. Naturally, the approximate equivalence between programs can also be reasoned by the approximate relational judgment, where δ characterizes the deviation of approximation.

[3] $Dim(\mathcal{H})$ denotes the dimension of $\mathcal{H}$.

Lemma 3 (Program Equivalence). *Program S_1 is equivalent*[4] *to program S_2 if and only if* $\vDash S_1 \sim S_2 : \equiv \Rightarrow \equiv$.

The program equivalence can be expressed concisely with predicates being the equivalence relation, instead of checking whether two quantum programs perform uniformly by enumeration of an infinite number of states in Hilbert space. The following example shows that superposition makes quantum program equivalence more complex than its classical counterpart.

Example 1. Let S_1 and S_2 be two programs defined on a single bit or qubit. For classical programs, the state space for programs S_1 and S_2 is the set $\{|0\rangle, |1\rangle\}$. Let Ψ and Φ be the equivalence relation, the relational judgment $S_1 \sim S_2 : \Psi \Rightarrow \Phi$ holds for classical programs S_1 and S_2 if

$$[\![S_1]\!](|0\rangle\langle 0|) = [\![S_2]\!](|0\rangle\langle 0|) \qquad [\![S_1]\!](|1\rangle\langle 1|) = [\![S_2]\!](|1\rangle\langle 1|) \tag{5}$$

However, this conclusion no longer holds in quantum programs since the input state could be a superposition of $|0\rangle$ and $|1\rangle$. For example, let $S_1 ::= \mathbf{skip}$ and $S_2 ::= q := Z[q]$, it is clear that S_1 and S_2 are not equivalent[5] although Eq. 5 still holds. To check quantum program equivalence, we need to verify the validity of $[\![S_1]\!](\rho) = [\![S_2]\!](\rho)$ for all ρ in the Hilbert space $\mathrm{span}\{|0\rangle, |1\rangle\}$ rather than the set $\{|0\rangle, |1\rangle\}$, which involves enumerations of an infinite set.

5.2 Proof Rules

We are ready to provide some proof rules for our aqRHL judgments. These rules include construct-specific rules (two-sided and one-sided) and structural ones, as is typical in relational Hoare logic. Notice that rules for branching structures are discussed in Sect. 7 later.

Simple Rules. Figure 2 includes the two/one-sided proof rules for basic statements and sequence structure. The basic rules, namely [Skip], [Init], [Ut] are similar to their counterparts in [8] with $\delta = 0$, where they are presented in the forward variant. Here we use $\mathrm{proj}(A)$ to lift non-projection A to its support before assigning it as a predicate. Notice that the rule [Ut] gives the strongest postcondition, which means the reverse $\vdash \bar{q}_1 := U_1^{-1}[\bar{q}_1] \sim \bar{q}_2 := U_2^{-1}[\bar{q}_2] : (U_1 \otimes U_2)A(U_1^\dagger \otimes U_2^\dagger) \Rightarrow A$ still holds. The rule [Seq] demonstrates that the deviation grows linearly with respect to the sequences, which directly comes from the triangle inequality of trace distance. One-sided rules are necessary when two programs do not share the same structure. We have only listed the one-side rules (appended with "-L") for the left side, and similar rules apply to the right side symmetrically.

[4] That is, $[\![S_1]\!](\rho) = [\![S_2]\!](\rho)$ holds for any partial density operator ρ.

[5] $[\![S_1]\!](|\psi\rangle\langle\psi|) \neq [\![S_2]\!](|\psi\rangle\langle\psi|)$ for any superposition $|\psi\rangle = a|0\rangle + b|1\rangle$, $0 < |a|^2 + |b|^2 \leq 1$.

$$
\begin{aligned}
&(\textsc{Skip}) && \vdash \mathbf{skip} \sim \mathbf{skip} : A \Rightarrow A \\
&(\textsc{Init}) && \vdash \bar{q}_1 := |0\rangle \sim \bar{q}_2 := |0\rangle : A \Rightarrow |0\rangle_{\bar{q}_1}\langle 0| \otimes |0\rangle_{\bar{q}_2}\langle 0| \otimes \mathrm{proj}(\mathrm{Tr}_{(\bar{q}_1,\bar{q}_2)}(A)) \\
&(\textsc{Ut}) && \vdash \bar{q}_1 := U_1[\bar{q}_1] \sim \bar{q}_2 := U_2[\bar{q}_2] : A \Rightarrow (U_1 \otimes U_2)A(U_1^\dagger \otimes U_2^\dagger) \\
&(\textsc{Seq}) && \frac{\vdash S_1 \sim_{\delta_1} S_2 : A \Rightarrow R \quad \vdash S_1' \sim_{\delta_2} S_2' : R \Rightarrow B}{\vdash S_1; S_1' \sim_{\delta_1+\delta_2} S_2; S_2' : A \Rightarrow B} \\
&(\textsc{Init-L}) && \vdash \bar{q}_1 := |0\rangle \sim \mathbf{skip} : A \Rightarrow |0\rangle_{\bar{q}_1}\langle 0| \otimes \mathrm{proj}(\mathrm{Tr}_{\mathcal{H}_{\bar{q}_1}}(A)) \\
&(\textsc{Ut-L}) && \vdash \bar{q}_1 := U_1[\bar{q}_1] \sim \mathbf{skip} : A \Rightarrow (U_1 \otimes I_2)A(U_1^\dagger \otimes I_2)
\end{aligned}
$$

Fig. 2. Simple aqRHL rules.

Rules for Equivalence Relation. We address a scenario regarding the rules [Ut], where the precondition and postcondition are equivalence relations defined in Definition 7. We use diamond norm to bound the deviation in rule [Ut-id], where $U \cdot U^\dagger$ denotes the Kraus representation [39] of unitary U. The rule [Comp] permits reasoning equivalence between programs by introducing intermediate programs (Fig. 3).

$$
\begin{aligned}
&(\textsc{Ut-id}) && \vdash \bar{q}_1 := U_1[\bar{q}_1] \sim_{\|U_1 \cdot U_1^\dagger - U_2 \cdot U_2^\dagger\|_\diamond/2} \bar{q}_2 := U_2[\bar{q}_2] : \equiv \Rightarrow \equiv \\
&(\textsc{Comp}) && \frac{\vdash S_1 \sim_{\delta_1} S_2 : \equiv \Rightarrow \equiv \qquad \vdash S_2 \sim_{\delta_2} S_3 : \equiv \Rightarrow \equiv}{\vdash S_1 \sim_{\delta_1+\delta_2} S_3 : \equiv \Rightarrow \equiv}
\end{aligned}
$$

Fig. 3. Rules for Equivalence Relation

5.3 Soundness Theorem

Theorem 1. [Soundness] *For any program S_1, S_2, projections A and B, deviation δ, we have,*

$$\vdash S_1 \sim_\delta S_2 : A \Rightarrow B \quad \Rightarrow \quad \vDash S_1 \sim_\delta S_2 : A \Rightarrow B$$

The soundness of our proof system is proved with respect to the validity of judgments defined in 9, while the completeness remains an open question. For classical deterministic programs, relational Hoare logic has been demonstrated to be relatively complete for terminating programs with the help of providing additional supplementary one-sided rules. However, relative completeness does not extend to probabilistic programs. As highlighted in [5], the probabilistic coupling method lacks the robustness of the conductance method in demonstrating the rapid mixing of Markov chains. Building upon the work laid by [8], the quantum extension of probabilistic couplings and the introduction of approximation in our judgments further complicate this problem.

6 Approximate Quantum Fourier Transform

Objective. As a quantum analog of the classical discrete Fourier transform, *quantum Fourier transform* (QFT) [17] performs a linear transformation on quantum states and extracts the periodicity of the amplitudes of quantum states. Due to the imperfectness of quantum gates, the *approximate quantum Fourier transform* (AQFT) is proposed to improve the circuit depth of QFT for efficiency. Reference [17] proposes a direct AQFT based on ignoring gates related to high-order terms. Cleve and Watrous [16] parallelized the phase estimation procedure to perform AQFT with lower circuit depth. Let S_{QFT} and S_{AQFT} be the corresponding quantum programs for QFT and AQFT, respectively. This section uses our logic to derive the judgment of form $S_{\mathrm{QFT}} \sim_{\delta} S_{\mathrm{AQFT}} : \equiv \Rightarrow \equiv$ to reason about how well AQFT approximates QFT.

Specification. For an n qubit system, QFT on a *computational basis state* $|x\rangle = |x_1 x_2 \ldots x_n\rangle$ is defined as the linear operation U such that

$$U|x\rangle = |\psi_x\rangle = \tfrac{1}{\sqrt{2^n}} \textstyle\sum_{y=0}^{N-1} (e^{2\pi i/N})^{x \cdot y} |y\rangle \tag{6}$$

where $N = 2^n$, $|\psi_x\rangle$ is called a *Fourier basis* state with respect to state $|x\rangle$, $x \cdot y$ denotes the multiplication between the binary representation of x and y. $|\psi_x\rangle$ can be described as $|\psi_x\rangle = |\mu_{0.x_n}\rangle |\mu_{0.x_{n-1}x_n}\rangle \cdots |\mu_{0.x_1 \ldots x_n}\rangle$, where $|\mu_\theta\rangle = (|0\rangle + e^{2\pi i\theta}|1\rangle)/\sqrt{2}$, $0.x_i \ldots x_j$ denotes the binary fraction $x_i/2 + x_{i+1}/4 + \cdots + x_j/2^{j-i+1}$. State $|\mu_\theta\rangle$ can be obtained by applying the phase shift gate $P(2\pi\theta)$ (mentioned in Sect. 2) on state $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. The phase shift gate $P(2\pi\theta)$ can be decomposed as the sequence of gates $R_m = P(2\pi/2^m)$ since $P(\theta_1)P(\theta_2) = P(\theta_1 + \theta_2)$. The controlled R_m gate is denoted by $CR_m[(q_1, q_2)]$, which is the c-$P(\theta)$ gate (mentioned in Sect. 2) with $\theta = 2\pi/2^m$.

QFT can be parallelly implemented [16], as shown in Fig. 4. The unitary V generates the Fourier basis state $|\psi_x\rangle$ without erasing $|x\rangle$. The unitary *Add* introduces auxiliary $(k-1)n$ qubits to create $k-1$ replicas of Fourier basis state $|\psi_x\rangle$. The unitary oracle T introduces auxiliary n qubits to compute the corresponding phase parameter $|x\rangle$ of the Fourier basis state $|\psi_x\rangle$ without erasing $|\psi_x\rangle$. All these auxiliary qubits are not depicted in Fig. 4 since they are reset back to $|0\rangle$ after the computation.

We can perform approximate computations for oracles V and T to achieve a lower circuit depth. Oracle V can be approximated by ignoring CR_m gates of larger m. Oracle T can be approximated by performing quantum measurements followed by classical post-processing on measurement outcomes [28]. Let unitary V' and T' be the approximation of V and T respectively, the corresponding program S_{AQFT} is almost the same as S_{QFT} but with oracles V and T replaced by V' and T' respectively. Next, we use our logic to reason the approximate equivalence between programs S_{QFT} and S_{AQFT}. That is,

$$S_{\mathrm{QFT}} \sim_{\delta_1 + 2\delta_2} S_{\mathrm{AQFT}} : \equiv_{(\bar{q}_0, \bar{q}'_0)} \otimes |0\rangle\langle 0|_{aux} \Rightarrow \equiv_{(\bar{q}_0, \bar{q}'_0)} \otimes |0\rangle\langle 0|_{aux} \tag{7}$$

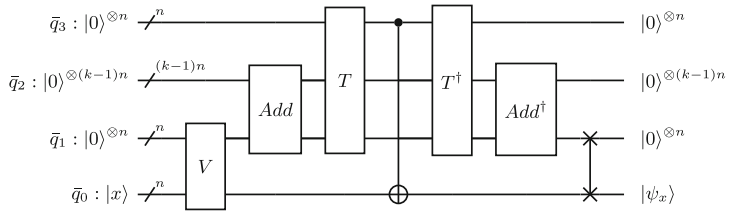

Fig. 4. QFT circuit in [16]. Given a computational basis state $|x\rangle$ and corresponding Fourier basis state $|\psi_x\rangle$, unitary V performs mapping $|x\rangle|0\rangle^{\otimes n} \mapsto |x\rangle|\psi_x\rangle$, unitary Add performs mapping $|\psi_x\rangle|0\rangle^{\otimes n} \cdots |0\rangle^{\otimes n} \mapsto |\psi_x\rangle|\psi_x\rangle \cdots |\psi_x\rangle$, and unitary T performs mapping $|\psi_x\rangle \cdots |\psi_x\rangle|0\rangle^{\otimes n} \mapsto |\psi_x\rangle \cdots |\psi_x\rangle|x\rangle$.

$$\left\{ \equiv_{(\bar{q}_0, \bar{q}'_0)} \otimes |0\rangle\langle 0|_{(\bar{q}_1, \bar{q}_2, \bar{q}_3, \bar{q}'_1, \bar{q}'_2, \bar{q}'_3, \bar{r})} \right\} // P_0$$

$$(\bar{q}_0, \bar{q}_1) := V[(\bar{q}_0, \bar{q}_1)]; \sim_{\delta_1} (\bar{q}'_0, \bar{q}'_1) := V'[(\bar{q}'_0, \bar{q}'_1)];$$

$$\{ (V \otimes V) P_0 (V^\dagger \otimes V^\dagger) \} // P_1$$

$$(\bar{q}_1, \bar{q}_2) := Add[(\bar{q}_1, \bar{q}_2)]; \sim (\bar{q}'_1, \bar{q}'_2) := Add[(\bar{q}'_1, \bar{q}'_2)];$$

$$\{ (Add \otimes Add) P_1 (Add^\dagger \otimes Add^\dagger) \} // P_2$$

$$(\bar{q}_1, \bar{q}_2, \bar{q}_3) = T[(\bar{q}_1, \bar{q}_2, \bar{q}_3)]; \sim_{\delta_2} (\bar{q}'_1, \bar{q}'_2, \bar{q}'_3) = T'[(\bar{q}'_1, \bar{q}'_2, \bar{q}'_3)];$$

$$\{ (T \otimes T) P_2 (T^\dagger \otimes T^\dagger) \} // P_3$$

$$(\bar{q}_3, \bar{q}_0) := CNOT(\bar{q}_3, \bar{q}_0); \sim (\bar{q}_3, \bar{q}_0) := CNOT(\bar{q}'_3, \bar{q}'_0);$$

$$\{ (CNOT \otimes CNOT) P_3 (CNOT \otimes CNOT)^\dagger \} // P_4$$

$$(\bar{q}_1, \bar{q}_2, \bar{q}_3) := T^\dagger[(\bar{q}_1, \bar{q}_2, \bar{q}_3)]; \sim_{\delta_2} (\bar{q}'_1, \bar{q}'_2, \bar{q}'_3, \bar{r}) := T^\dagger[(\bar{q}'_1, \bar{q}'_2, \bar{q}'_3, \bar{r})];$$

$$\{ (T^\dagger \otimes T^\dagger) P_4 (T \otimes T) \} // P_5$$

$$(\bar{q}_1, \bar{q}_2) := Add^\dagger[(\bar{q}_1, \bar{q}_2)]; \sim (\bar{q}'_1, \bar{q}'_2) := Add^\dagger[(\bar{q}'_1, \bar{q}'_2)];$$

$$\{ (Add^\dagger \otimes Add^\dagger) P_5 (Add \otimes Add) \} // P_6$$

$$(\bar{q}_0, \bar{q}_1) := SWAP(\bar{q}_0, \bar{q}_1); \sim (\bar{q}'_0, \bar{q}'_1) := SWAP(\bar{q}'_0, \bar{q}'_1);$$

$$\{ (SWAP \otimes SWAP) P_6 (SWAP^\dagger \otimes SWAP^\dagger) \} // P_7 = P_0$$

Fig. 5. Proof sketch for programs S_{QFT} and S_{AQFT}. To easily refer to predicates, we label each assertion a name $// P_i$ on its right.

where $\delta = n\pi 2^{-k-1} + 2ne^{-k/8}$, $|0\rangle\langle 0|_{aux}$ denotes the tensor product of constant projections $|0\rangle\langle 0|$ over all qubits in other registers except $\bar{q}_0$ and $\bar{q}'_0$. The main proof sketch is shown in Fig. 5.

Create the Fourier Basis State. The computation of unitary U in Eq. 6 can be parallelized by individually preparing every $|\mu_\theta\rangle$ by the following unitary

$$Q_{t,i} : |0\rangle^{\otimes t}|x_1 \ldots x_n\rangle \mapsto |\mu_{0.x_i \ldots x_{i+t-1}}\rangle|0\rangle^{\otimes t-1}|x_1 \ldots x_n\rangle$$

in [16], where $i+t-1 \leq n$, qubits $x_1 \ldots x_{i-1}$ and $x_{i+t} \ldots x_n$ in $|x\rangle$ are not used. The unitary $Q_{t,i}$ acting on register $(\bar{q}, \bar{p})$ can be denoted as,

$$U_{GHZ}[\bar{q}]; CR_1[(\bar{p}[i], \bar{q}[1])]; \ldots; CR_t[(\bar{p}[i+t-1], \bar{q}[t])]; U^{\dagger}_{GHZ}[\bar{q}]; H[\bar{q}[1]]$$

where U_{GHZ} denotes the unitary that generates a GHZ state, that is, $U_{GHZ}|0\rangle^{\otimes t} = (|0\rangle^{\otimes t} + |1\rangle^{\otimes t})/\sqrt{2}$. Registers $\bar{q}$ and $\bar{p}$ are of size t and n, respectively. $\bar{q}[i]$ denotes the i-th qubit in register $\bar{q}$. For example, Fig. 6 in [16] represents the circuit of unitary $Q_{4,i}$ on $|x\rangle$. Similar to the approximation in [17], unitary $Q_{t,i}$ can be approximated by ignoring CR_m gates of large m. That is, we could use $Q_{t,i}$ to approximate $Q_{t',i}$ if $1 \leq t < t' \leq n$. The approximation can be modeled by the judgment

$$\begin{aligned} &\vdash (\bar{q}, \bar{p}) := Q_{t,i}[(\bar{q}, \bar{p})] \sim_{\delta(t,t')} (\bar{q}', \bar{p}') := Q_{t',i}[(\bar{q}', \bar{p}')] : \\ &|0\rangle\langle 0|_{(\bar{q},\bar{q}')} \otimes \equiv_{(\bar{p},\bar{p}')} \Rightarrow \equiv_{(\bar{q}[1],\bar{q}'[1])} \otimes |0\rangle\langle 0|_{(\bar{q}[2,n],\bar{q}'[2,n])} \otimes \equiv_{(\bar{p},\bar{p}')} \end{aligned} \tag{8}$$

with $\delta(t,t') = \frac{1}{2}\sin\pi(2^{-t} - 2^{-t'})$. $P_{\bar{q}}$ denotes a projection P over the register $\bar{q}$. Particularly, $|\psi\rangle\langle\psi|_{\bar{q}}$ denotes the tensor product of $|\psi\rangle\langle\psi|$ over all qubits in register $\bar{q}$.

Figure 7 illustrates the circuit of the oracle V over register $(\bar{q}_0, \bar{r}, \bar{q}_1)$. To prepare each $|\mu_\theta\rangle$ in $|\psi_x\rangle$ individually, we need to prepare n copies of state $|x\rangle$ beforehand, which is achieved by the unitary C. The unitary C can be implemented by $CNOT$ gates in a binary tree architecture to achieve a circuit depth of $\log n$. To make it concise, the auxiliary qubits $q[2,n]$ in oracle $Q_{t,i}(\bar{q}, \bar{p})$ that reset back to $|0\rangle$ are ignored in Fig. 7 and the input of $Q_{t,i}$ is set as $|0\rangle|x\rangle$. The circuit for oracle V' is almost the same as Fig. 7 except that $Q_{t,i}$ is approximated by $Q_{k,i}$, where k $(0 < k < t \leq n)$ denotes the number of significant phase shift gates. Specifically, oracles V and V' can be represented as follows,

$$\begin{aligned} V &= C[(\bar{q}_0, \bar{r})]; Q_{1,n}[(\bar{q}_1[1], \bar{q}_0)]; Q_{2,n-1}[(\bar{q}_1[2], \bar{r}_1)]; \ldots; Q_{n,1}[(\bar{q}_1[n], \bar{r}_{n-1})]; C^{\dagger}[(\bar{q}_0, \bar{r})] \\ V' &= C[(\bar{q}'_0, \bar{r}')]; Q_{1,n}[(\bar{q}'_1[1], \bar{q}'_0)]; Q_{2,n-1}[(\bar{q}'_1[2], \bar{r}'_1)]; \ldots; Q_{k,n-k+1}[(\bar{q}'_1[k], \bar{r}'_{k-1})]; \\ &\quad Q_{k,n-k+1}[(\bar{q}'_1[k+1], \bar{r}'_k)]; \ldots; Q_{k,n-k+1}[(\bar{q}'_1[n], \bar{r}'_{n-1})]; C^{\dagger}[(\bar{q}'_0, \bar{r}')] \end{aligned}$$

where register $\bar{r} = \{\bar{r}_1, \ldots, \bar{r}_{n-1}\}$ contains $n-1$ registers $\bar{r}_i$ initialized with $|0\rangle^{\otimes^n}$. Based on judgement 8, we have the following judgment

$$\vdash (\bar{q}_0, \bar{q}_1) := V[(\bar{q}_0, \bar{q}_1)] \sim_{\delta_1} (\bar{q}'_0, \bar{q}'_1) := V'[(\bar{q}'_0, \bar{q}'_1)] : P_0 \;\Rightarrow\; P_1 \tag{9}$$

where $\delta_1 = \sum_{i=k+1}^{n} \delta(k,i) = \frac{1}{2}\sum_{i=k+1}^{n} \sin\pi(2^{-k} - 2^{-i}) \leq n\pi 2^{-k-1}$. Notice that every register $\bar{r}_i$ in Fig. 7 is reset back to $|0\rangle$, thus the predicate $|0\rangle\langle 0|_{(\bar{r},\bar{r}')}$ on register $(\bar{r}, \bar{r}')$ can be ignored.

Replicate & Erase Fourier Basis State. We provide a brief overview of the functionality of the oracle *Add* as described in [16]. We begin with the state $|\psi_x\rangle|0\rangle^{\otimes^n} \cdots |0\rangle^{\otimes^n}$ and apply Hadamard gates $H^{\otimes^n}$ to each $|0\rangle^{\otimes^n}$, resulting in $|\psi_x\rangle|\psi_0\rangle \cdots |\psi_0\rangle$. Then, we apply telescoping subtraction $|x_1\rangle|x_2\rangle \cdots |x_k\rangle \rightarrow$

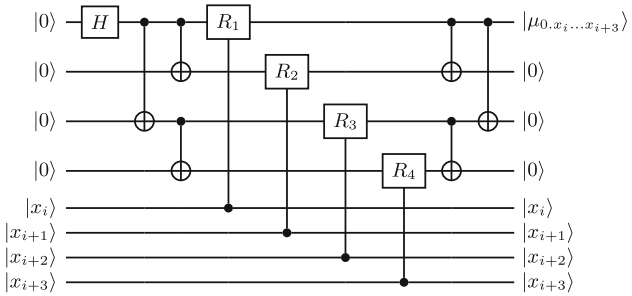

Fig. 6. Circuit for oracle $Q_{4,i}$ on state $|x_1 \ldots x_n\rangle$. Qubits $x_1 \ldots x_{i-1}$ and $x_{i+4} \ldots x_n$ are not used and ignored.

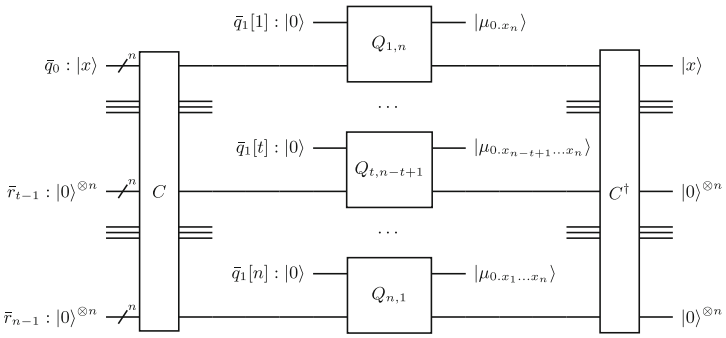

Fig. 7. Circuit for oracle V. Given a computational basis state $|x\rangle = |x_1 \ldots x_n\rangle$, unitary C performs the mapping $|x\rangle|0\rangle^{\otimes^n} \cdots |0\rangle^{\otimes^n} \mapsto |x\rangle^{\otimes^n}$, and unitary $Q_{t,i}$ performs the mapping $|0\rangle|x\rangle \mapsto |\mu_{0.x_i \ldots x_{i+t-1}}\rangle|x\rangle$.

$|x_1\rangle|x_2 - x_1\rangle \cdots |x_k - x_{k-1}\rangle$ to obtain $|\psi_x\rangle|\psi_x\rangle \ldots |\psi_x\rangle$. Reversely, we can use prefix addition $|x_1\rangle|x_2\rangle \cdots |x_k\rangle \to |x_1\rangle|x_1 + x_2\rangle \cdots |x_1 + x_2 + \cdots + x_k\rangle$ to eliminate the duplicates of the Fourier basis state. A $\log(k)$-depth tree of 3-2 adders can be used to generate two encoded numbers, followed by a quantum carry-lookahead adder of $\log(n)$-depth to add the encoded numbers. Since programs S_{AFT} and S_{QAFT} share the same procedure to replicate and erase Fourier basis states, we simplify replicating and erasing procedures by treating them as quantum oracles Add and $Add^\dagger$ respectively. Then we use rule [Ut] to get the following judgment.

$$\vdash (\bar{q}_1, \bar{q}_2) := Add[(\bar{q}_1, \bar{q}_2)] \sim (\bar{q}_1', \bar{q}_2') := Add[(\bar{q}_1', \bar{q}_2')] : P_1 \Rightarrow P_2 \tag{10}$$

$$\vdash (\bar{q}_1, \bar{q}_2) := Add^\dagger[(\bar{q}_1, \bar{q}_2)] \sim (\bar{q}_1', \bar{q}_2') := Add^\dagger[(\bar{q}_1', \bar{q}_2')] : P_5 \Rightarrow P_6 \tag{11}$$

Estimate the Phase of a Fourier State. The key to this step is based on the idea [3] that quantum measurement can be simulated by unitaries with the help of ancillary qubits. As shown in Fig 4, the oracle T generates the phase $|x\rangle$ in register $\bar{q}_3$ of the Fourier state $|\psi_x\rangle$, then the Fourier basis state $|x\rangle$ in register

$\bar{q_0}$ can be erased by the following $CNOT$ gate ($CNOT|x\rangle|x\rangle = |x\rangle|0\rangle$). The gate $T^\dagger$, the reverse of T, is applied subsequently to restore the state to the duplicates of $|\psi_x\rangle$. Given the input $|x\rangle$ in register $\bar{q}_0$, the whole process of erasing $|x\rangle$ works as $|x\rangle|\psi_x\rangle\cdots|\psi_x\rangle|0\rangle^{\otimes^n} \xrightarrow{T} |x\rangle|\psi_x\rangle\cdots|\psi_x\rangle|x\rangle \xrightarrow{CNOT} |0\rangle^{\otimes^n}|\psi_x\rangle\cdots|\psi_x\rangle|x\rangle \xrightarrow{T^\dagger} |0\rangle^{\otimes^n}|\psi_x\rangle\cdots|\psi_x\rangle|0\rangle^{\otimes^n}$ where the auxiliary register $\bar{q}_3$ is initialized with $|0\rangle^{\otimes^n}$ and reset back to $|0\rangle^{\otimes^n}$.

In order to reduce the circuit depth of oracle T, [16] parallelized the phase estimation procedure proposed by [29]. Given k copies of each $|\mu_{x2^{-i}}\rangle$, we perform two single-qubit measurements

$$\mathcal{M}_1 = \{M_1^0 = |\mu_0\rangle\langle\mu_0|, M_1^1 = |\mu_{\frac{1}{2}}\rangle\langle\mu_{\frac{1}{2}}|\} \quad \mathcal{M}_2 = \{M_2^0 = |\mu_{\frac{1}{4}}\rangle\langle\mu_{\frac{1}{4}}|, M_2^1 = |\mu_{\frac{3}{4}}\rangle\langle\mu_{\frac{3}{4}}|\}$$

on $k/2$ of the copies independently, where $\{|\mu_0\rangle, |\mu_{\frac{1}{2}}\rangle\}$ and $\{|\mu_{\frac{1}{4}}\rangle, |\mu_{\frac{3}{4}}\rangle\}$ are the eigenvectors of Pauli operators X and Y respectively. These measurements on copies of $|\psi_x\rangle$ would generate a distribution $\{p_{(x,i)}\}$ over a nk-bit string $|m_{(x,i)}\rangle$ of measurement outcomes. Then a reversible classical processing f is applied to infer x_i' based on measurement outcome $|m_{(x,i)}\rangle$, that is $|m_{(x,i)}\rangle|0\rangle \to |m_{(x,i)}\rangle|x_i'\rangle$, where the probability $p_{(x,i)}$ is close to 1 if $|x_i'\rangle = |x\rangle$, and a properly estimated $|x_i'\rangle$ can be used to erase the phase $|x\rangle$ on register $\bar{q}_0$. The following lemma is proved using Chernoff bound.

Lemma 4. [16] *Given any computational basis $|x\rangle$, measuring observables X and Y randomly generates a distribution $\{p_{(x,i)}\}$ over $\{|m_{(x,i)}\rangle\}$, followed by a classical processing that generates phase $|x_i'\rangle$ from $|m_{(x,i)}\rangle$. We have $Pr(|x_i'\rangle = |x\rangle) = p_{(x,i)} > 1 - 4ne^{-k/8}$.*

We can convert the above whole process into a unitary operation T' without actual measurements that can operate on data in superposition. First, the following unitary $U_M([\bar{q}_1, \bar{q}_2, \bar{r}])$,

$$U_M([\bar{q}_1, \bar{q}_2, \bar{r}]) := \otimes_{i=1}^{n}(\otimes_{j=1}^{k/2} U_X[(r[ik+j], p[ik+j])])\otimes(\otimes_{j=1+k/2}^{k} U_Y[(r[ik+j], p[ik+j])])$$

is applied to simulate measurements on copies of $|\psi_x\rangle$, where register $\bar{p} = \{\bar{q}_1, \bar{q}_2\}$ and auxiliary register $\bar{r}$ is initialized with $|0\rangle$. Unitary gates U_X and U_Y

$$U_X[(q_1, q_2)] := (H[q_1] \otimes I[q_2])CNOT[(q_1, q_2)](H[q_1] \otimes I[q_2]);$$
$$U_Y[(q_1, q_2)] := (H[q_1] \otimes I[q_2])CY[(q_1, q_2)](H[q_1] \otimes I[q_2])$$

introduce auxiliary qubit q_1 initialized with $|0\rangle$ to simulate single-qubit measurements $\mathcal{M}_1$ and $\mathcal{M}_2$ on $|\mu_{x2^{-i}}\rangle$ in qubit q_2.

$$|0\rangle|\mu_{x2^{-i}}\rangle \xrightarrow{U_X} \langle\mu_0|\mu_{x2^{-i}}\rangle \cdot |0\rangle|\mu_0\rangle + \langle\mu_{\frac{1}{2}}|\mu_{x2^{-i}}\rangle \cdot |1\rangle|\mu_{\frac{1}{2}}\rangle$$
$$|0\rangle|\mu_{x2^{-i}}\rangle \xrightarrow{U_Y} \langle\mu_{\frac{1}{4}}|\mu_{x2^{-i}}\rangle \cdot |0\rangle|\mu_{\frac{1}{4}}\rangle + \langle\mu_{\frac{3}{4}}|\mu_{x2^{-i}}\rangle \cdot |1\rangle|\mu_{\frac{3}{4}}\rangle$$

where $CY[(q_1, q_2)]$ denotes the controlled Pauli Y gate. Next, we set the outputs of auxiliary register $\bar{r}$ of U_M to be the input of oracle $O[(\bar{r}, \bar{q_3})]$ such that

$$U_M|\mu_{x2^{-i}}\rangle \otimes |0\rangle \xrightarrow{O} \textstyle\sum_i \sqrt{p_{(x,i)}}|\varphi\rangle \otimes |x_i'\rangle$$

where oracle O denotes the corresponding quantum circuit of the classical processing f on measurement outcomes. Thus, the oracle T' can achieved by $U_M[(\bar{q_1}, \bar{q_2}, \bar{r})]$ and $O[(\bar{r}, \bar{q_3})]$ sequentially. By lemma 4, we would have

$$\begin{aligned} &\vdash (\bar{q}_1, \bar{q}_2, \bar{q}_3) := T[(\bar{q}_1, \bar{q}_2, \bar{q}_3)] \sim_{\delta_2} (\bar{q_1'}, \bar{q_2'}, \bar{q_3'}, \bar{r}) := T'[(\bar{q_1'}, \bar{q_2'}, \bar{q_3'}, \bar{r})] : P_2 \;\Rightarrow\; P_3 \\ &\vdash (\bar{q}_1, \bar{q}_2, \bar{q}_3) := T^{\dagger}[(\bar{q}_1, \bar{q}_2, \bar{q}_3)] \sim_{\delta_2} (\bar{q_1'}, \bar{q_2'}, \bar{q_3'}, \bar{r}) := T'^{\dagger}[(\bar{q_1'}, \bar{q_2'}, \bar{q_3'}, \bar{r})] : P_4 \Rightarrow P_5 \end{aligned} \tag{12}$$

where $\delta_2 = 2ne^{-k/8}$.

Conclusion Finally, we use rule [Seq] to sum up all judgments to get Eq. 7.

7 Measurements Conditions and Additional Proof Rules

7.1 Measurement Conditions

Additional constraints must be imposed on programs to establish feasible relational proof rules for those with complex structures, such as if and loop statements. In the classical pRHL approach in [7], the precondition $m_1 \Psi m_2$ satisfied by the initial memories m_1 and m_2 requires the guards e_1 and e_2 in the if or loop statements must be equal. Things get more complex in quantum programs since quantum mechanics are naturally probabilistic, and it is generally impossible to require two if statements to give the same measurement or with the same probability distributions. In [8], the term "synchronous execution" in quantum programs means that two quantum measurements $\mathcal{M}_1 = \{M_1^m\}$ and $\mathcal{M}_2 = \{M_2^m\}$ should produce the same distribution over branches for input ρ_1 and ρ_2, that is, $\mathrm{Tr}(M_1^m \rho_1 M_1^{m\dagger}) = \mathrm{Tr}(M_2^m \rho_2 M_2^{m\dagger})$. To study more general programs, we propose the approximate measurement conditions, which establish appropriate upper bounds for the deviations in our judgments.

Definition 10 (Approximate Measurement Condition). *Let* $\mathcal{M}_1 = \{M_1^m\}$ *and* $\mathcal{M}_2 = \{M_2^m\}$ *be two measurements in* $\mathcal{H}_1$ *and* $\mathcal{H}_2$ *that share the same set* $\{m\}$ *of measurement outcomes, respectively. The measurement condition*

$$\mathcal{M}_1 \approx_{\{\delta_m\}} \mathcal{M}_2 : A \Rightarrow \{(p_m, B_m)\} \tag{13}$$

means that for every measurement outcome m*, we have*

$$\forall \rho_1,\ \rho_2 .\, \rho_1 \sim_A \rho_2 \;\Rightarrow\; \begin{cases} M_1^m \rho_1 M_1^{m\dagger} \sim_{B_m}^{\delta_m} M_2^m \rho_2 M_2^{m\dagger} \\ \max\{\mathrm{Tr}(M_1^m \rho_1 M_1^{m\dagger}), \mathrm{Tr}(M_2^m \rho_2 M_2^{m\dagger})\} \leq p_m \end{cases}$$

where $p_m \in [0, 1]$*. Deviations* δ_m *in the measurement condition can be ignored if they equal zero. We write predicate* $\{(p_m, B_m)\}$ *as* $\{B_m\}$ *for short if all* $p_m = 1$*.*

7.2 Additional Proof Rules

Now, we introduce the rules for if and loop statements in Fig. 8. The rule [IF] requires the measurement condition in the premises to provide a bound on the whole approximation, where the deviation δ'_m of the branch body is scaled down by p_m. The rule [LP] does not require the synchronous execution of loop guards [8,9], or the speed bound at which loops converge [26]. Instead, the measurement condition only employs an upper bound p_1 on the probabilities of entering loop bodies for the first iteration. Rule [LP] requires $p_1 \in [0, 1)$ and provides better deviation if p_1 is smaller. If p_1 equals one initially, we can unroll loop statements several times to make p_1 less than one.

We derive rule [LP*] as an alternative to rule [LP] by incorporating more specific measurement conditions for the iterations of loops when we can not find a good A for rule [LP]. When doing approximate reasoning about loops, it is typical to set an upper bound N on the number of iterations. Notice that the factor λ_n is not an upper bound on the probability of entering $(n+1)$-th iteration except for $n = 0$. Overall, rule [LP*] is a direct application of rule [Seq] on a finite number of iterations, where measurement conditions are used to scale the deviations. Since we can always make the **skip** statement share the same probability distribution with any **if** and **while** statements, the measurement conditions for one-side rules [IF-L], [LP-L], and [LP*-L] are more straightforward.

$$\text{(IF)}\quad \frac{\mathcal{M}_1 \approx_{\{\delta_m\}} \mathcal{M}_2 : A \Rightarrow \{(p_m, B_m)\} \quad \vdash S_{1m} \sim_{\delta'_m} S_{2m} : B_m \Rightarrow C}{\vdash \textbf{if } (\square m \cdot \mathcal{M}_1[\bar{q}] = m \to S_{1m}) \textbf{ fi} \sim_{\sum_m \delta_m + p_m \cdot \delta'_m} \textbf{if } (\square m \cdot \mathcal{M}_2[\bar{q}] = m \to S_{2m}) \textbf{ fi} : A \Rightarrow C}$$

$$\text{(LP)}\quad \frac{\mathcal{M}_1 \approx_{\{\delta_0,\delta_1\}} \mathcal{M}_2 : A \Rightarrow \{(p_0, B_0), (p_1, B_1)\} \quad \vdash S_1 \sim_\delta S_2 : B_1 \Rightarrow A}{\vdash \textbf{while } \mathcal{M}_1[\bar{q}] = 1 \textbf{ do } S_1 \textbf{ od} \sim_{\frac{\delta_0+\delta_1+p_1\delta}{1-p_1}} \textbf{while } \mathcal{M}_2[\bar{q}] = 1 \textbf{ do } S_2 \textbf{ od} : A \Rightarrow B_0}$$

$$\text{(LP*)}\quad \frac{\begin{array}{c}\forall 0 \le k < N. \mathcal{M}_1 \approx_{\{\alpha_k,\beta_k\}} \mathcal{M}_2 : A_k \Rightarrow \{(q_k, C), (p_k, B_k)\} \\ \mathcal{M}_1 \approx_{\{\alpha_N,0\}} \mathcal{M}_2 : A_N \Rightarrow \{(1, C), (0, I)\} \quad \vdash S_1 \sim_{\delta_k} S_2 : B_k \Rightarrow A_{k+1}\end{array}}{\vdash \textbf{while } \mathcal{M}_1[\bar{q}] = 1 \textbf{ do } S_1 \textbf{ od} \sim_{f(\alpha_k,\beta_k,p_k)} \textbf{while } \mathcal{M}_2[\bar{q}] = 1 \textbf{ do } S_2 \textbf{ od} : A_0 \Rightarrow C}$$

$$\text{(IF-L)}\quad \frac{\mathcal{M}_1 \approx I_2 : A \Rightarrow \{(p_m, B_m)\} \quad \vdash S_{1m} \sim_{\delta_m} \textbf{skip} : B_m \Rightarrow C}{\vdash \textbf{if } (\square m \cdot \mathcal{M}_1[\bar{q}] = m \to S_{1m}) \textbf{ fi} \sim_{\sum_m p_m \delta_m} \textbf{skip} : A \Rightarrow C}$$

$$\text{(LP-L)}\quad \frac{\mathcal{M}_1 \approx I_2 : A \Rightarrow \{(p_0, B_0), (p_1, B_1)\} \quad \vdash S_1 \sim_\delta \textbf{skip} : B_1 \Rightarrow A}{\vdash \textbf{while } \mathcal{M}_1[\bar{q}] = 1 \textbf{ do } S_1 \textbf{ od} \sim_{p_1\delta/(1-p_1)} \textbf{skip} : A \Rightarrow B_0}$$

$$\text{(LP*-L)}\quad \frac{\begin{array}{c}\forall 0 \le k < N. \mathcal{M}_1 \approx I_2 : A_k \Rightarrow \{(q_k, C), (p_k, B_k)\} \quad \lambda_n = \prod_{k=0}^{n} p_k \\ \mathcal{M}_1 \approx_{\{\delta_N,0\}} I_2 : A_N \Rightarrow \{(1, C), (0, I)\} \quad \vdash S_1 \sim_{\delta_k} \textbf{skip} : B_k \Rightarrow A_{k+1}\end{array}}{\vdash \textbf{while } \mathcal{M}_1[\bar{q}] = 1 \textbf{ do } S_1 \textbf{ od} \sim_{\lambda_{N-1}\delta_N + \sum_{n=0}^{N-1}(N-n)\lambda_n\delta_n)} \textbf{skip} : A_0 \Rightarrow C}$$

Fig. 8. Rules for branching structure in aqRHL. The deviation of rule (LP*) is given by $f(\alpha_k, \beta_k, p_k) = (\alpha_0 + \sum_{n=0}^{N-1} \lambda_n \alpha_{n+1}) + (N\beta_0 + \sum_{n=0}^{N-2}(N-n-1)\lambda_n\beta_{n+1}) + (\sum_{n=0}^{N-1}(N-n)\lambda_n\delta_n)$ with $\lambda_n = \prod_{k=0}^{n} p_k$.

Structural Rules. Unlike classical programs, the potential quantum entanglement between subsystems brings a unique challenge in constructing a general frame

rule for quantum programs [8,51,64]. We derive a simple frame rule [Frame] to specify a specific instance that the predicate C on additional independent system $(\bar{r}_1, \bar{r}_2)$ is one-dimensional. Subscripts related to registers are displayed explicitly for clarity. Rule [Order] adds an order relation $\leq$ over deviations. In addition, an additional condition is introduced in rule [Approx] to allow switching postconditions at the cost of bringing approximation (Fig. 9).

$$\text{(Frame)}\quad \frac{\vdash_{(\bar{q}_1,\bar{q}_2)} S_1 \sim_\delta S_2 : A \Rightarrow B \quad (\bar{r}_1,\bar{r}_2) \cap var(S_1,S_2) = 0 \quad Dim(C_{(\bar{r}_1,\bar{r}_2)}) = 1}{\vdash_{(\bar{q}_1,\bar{r}_1),(\bar{q}_2,\bar{r}_2)} S_1 \sim_\delta S_2 : A \otimes C_{(\bar{r}_1,\bar{r}_2)} \Rightarrow B \otimes C_{(\bar{r}_1,\bar{r}_2)}}$$

$$\text{(Order)}\quad \frac{\vdash S_1 \sim_{\delta'} S_2 : A' \Rightarrow B' \quad A \subseteq A' \quad B' \subseteq B \quad \delta' \leq \delta}{\vdash S_1 \sim_\delta S_2 : A \Rightarrow B}$$

$$\text{(Approx)}\quad \frac{\vdash S_1 \sim S_2 : A \Rightarrow B \qquad \forall \rho_1, \rho_2.\ \rho_1 \sim_B \rho_2 \Rightarrow \rho_1 \sim_C^\delta \rho_2}{\vdash S_1 \sim_\delta S_2 :\ A \Rightarrow C}$$

Fig. 9. Structural aqRHL rules.

8 Related and Future Works

With the fast development of quantum hardware [50], various quantum programming languages [1,4,19,20,22,47,48] have been proposed for more straightforward implementation of quantum algorithms. Very recently, significant efforts have been devoted to the research of quantum logic and quantum program analysis [14,23,43,49,55,56,63,65] for these emerging quantum programs.

Comparison with Quantitative Robustness Reasoning. [26] develops semantics for erroneous quantum while programs and logic to prove robustness between an ideal program and a noisy one. [66] derives applied quantum Hoare logic by employing projection as predicates and reasons about the robustness of quantum programs, i.e., error bounds of outputs. These two works focus on single-program executions, while our work studies relational reasoning. In particular, the major differences are as follows. a). Different formula: In the logic formula of [26,66], the predicate lives in the space of the principle program. The predicate of our logic lives in the joint space of the two programs. b). Different scope of applications: The proof systems developed by [26,66] focus on studying the robustness of quantum programs, i.e., equivalence or closeness. Our choice of relational Hoare logic can reason about general relations beyond equivalence or closeness. We can reason relational properties between programs with different numbers of qubits. c). Different proof rules: The proof rules of [26,66] discuss programs with the same syntax statement, while our one-side rules can track relational properties for different statements.

Comparison with Relational Quantum Hoare Logics. Our work is primarily inspired by the quantum relational Hoare logics recently proposed by [8,33,51].

In particular, [8] suggests that casting approximate reasoning into the general framework of relational quantum Hoare logic remains open. Generally, two quantum while programs do not share the same probabilities for taking different paths or outcomes during their execution. Under such circumstances, exact quantum couplings cannot be found, as they only exist for partial density operators with identical trace. This mathematical condition significantly restricts the flexibility of the exact quantum relational Hoare logic. Our work provides a promising solution to this open question. In particular, by introducing approximate quantum coupling, our logic system offers a more general scope of applications. Our logic, aqRHL, is a quantum counterpart to apRHL [9], even from a technical point of view: aqRHL employs projective predicates [12] over the joint systems of the programs, a natural quantum counterpart of binary relations, the predicates used in apRHL.

Future Work. There are several promising directions for future work. Firstly, we would like to extend our theory to the hybrid system, i.e., programs with quantum and classical variables. Hybrid quantum-classical systems allow for the exploitation of quantum advantages while leveraging the existing classical computing infrastructure. A unified language incorporating both quantum and classical effects may offer advantages in analyzing hybrid programs [52]. Secondly, we will investigate the potential applications of the newly developed approximate relational quantum Hoare logic. Particularly, we are interested in applying it to the construction and verification of quantum cryptographic proofs and ensuring the correctness of optimized quantum compilers specifically designed for NISQ (Noisy Intermediate-Scale Quantum) devices. Lastly, it is interesting to incorporate recently developed tools such as quantum abstract interpretation [62] and quantum separation logic [64] to design over-approximation techniques [58]. Another interesting technique is Context-Free-Language Ordered Binary Decision Diagrams [46], which may serve as a backend representation and manipulation technique in studying quantum Hoare logics.

9 Conclusion

We resolve the open question of [8] by designing an approximate relational Hoare logic for robustly reasoning the relational properties of two programs. We showcase the success of our methodology by formally verifying the well-known low-depth approximation of the quantum Fourier transform, and the correctness of the repeat-until-success algorithm and bit flip code.

Acknowledgement. We thank our anonymous referees for their comments and suggestions on earlier versions of this paper. Hanru Jiang acknowledges the support from the National Natural Science Foundation of China under Grant No. 62202265, and the Beijing Natural Science Foundation under Grant No. Z220002.

References

1. Abhari, A.J., et al.: Scaffold: quantum programming language. Princeton University NJ Department of Computer Science, Tech. rep. (2012)
2. Aharonov, D., Kitaev, A., Nisan, N.: Quantum circuits with mixed states (1998). https://doi.org/10.48550/ARXIV.QUANT-PH/9806029, https://arxiv.org/abs/quant-ph/9806029
3. Aharonov, D., Kitaev, A., Nisan, N.: Quantum circuits with mixed states. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 20-30. STOC '98, Association for Computing Machinery, New York, NY, USA (1998). https://doi.org/10.1145/276698.276708
4. Aleksandrowicz, G., et al.: Qiskit: an open-source framework for quantum computing (2019). https://doi.org/10.5281/zenodo.2562111
5. Anil Kumar, V., Ramesh, H.: Coupling vs. conductance for the jerrum-sinclair chain*. Random Struct. Algorithms **18**(1), 1–17 (2001). https://doi.org/10.1002/1098-2418(200101)18:11::AID-RSA13.0.CO;2-7
6. Badihi, S., Akinotcho, F., Li, Y., Rubin, J.: ARDiff: scaling program equivalence checking via iterative abstraction and refinement of common code. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 13–24. ESEC/FSE 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3368089.3409757
7. Barthe, G., Grégoire, B., Zanella Béguelin, S.: Formal certification of code-based cryptographic proofs. In: Proceedings of the 36th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 90–101. POPL '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1480881.1480894
8. Barthe, G., Hsu, J., Ying, M., Yu, N., Zhou, L.: Relational proofs for quantum programs. Proc. ACM Program. Lang. **4**(POPL) (2019). https://doi.org/10.1145/3371089
9. Barthe, G., Köpf, B., Olmedo, F., Zanella-Béguelin, S.: Probabilistic relational reasoning for differential privacy. ACM Trans. Program. Lang. Syst. **35**(3) (2013). https://doi.org/10.1145/2492061
10. Benton, N.: Simple relational correctness proofs for static analyses and program transformations. In: Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 14–25. POPL '04, Association for Computing Machinery, New York, NY, USA (2004). https://doi.org/10.1145/964001.964003
11. Bergstra, J., Tiuryn, J., Tucker, J.: Floyd's principle, correctness theories and program equivalence. Theor. Comput. Sci. **17**(2), 113–149 (1982). https://doi.org/10.1016/0304-3975(82)90001-9, https://www.sciencedirect.com/science/article/pii/0304397582900019
12. Birkhoff, G., Neumann, J.V.: The logic of quantum mechanics. Ann. Math. **37**(4), 823–843 (1936). http://www.jstor.org/stable/1968621
13. Bocharov, A., Roetteler, M., Svore, K.M.: Efficient synthesis of universal repeat-until-success quantum circuits. Phys. Rev. Lett. **114**, 080502 (2015). https://doi.org/10.1103/PhysRevLett.114.080502
14. Chen, Y.F., Chung, K.M., Lengál, O., Lin, J.A., Tsai, W.L., Yen, D.D.: An automata-based framework for verification and bug hunting in quantum circuits. Proc. ACM Program. Lang. **7**(PLDI) (2023). https://doi.org/10.1145/3591270

15. Churchill, B., Padon, O., Sharma, R., Aiken, A.: Semantic program alignment for equivalence checking. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 1027–1040. PLDI 2019, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3314221.3314596
16. Cleve, R., Watrous, J.: Fast parallel circuits for the quantum fourier transform. In: Proceedings 41st Annual Symposium on Foundations of Computer Science, pp. 526–536. IEEE Computer Society, Redondo Beach, CA, USA (2000). https://doi.org/10.1109/SFCS.2000.892140
17. Coppersmith, D.: An approximate fourier transform useful in quantum factoring (2002). https://doi.org/10.48550/arxiv.quant-ph/0201067
18. Cousineau, G., Enjalbert, P.: Program equivalence and provability. In: Bečvář, J. (ed.) Mathematical Foundations of Computer Science, pp. 237–245. Springer, Berlin, Heidelberg (1979). https://doi.org/10.1007/3-540-09526-8_20
19. Developers, C.: Cirq (2021). https://doi.org/10.5281/zenodo.5182845, See full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors
20. Green, A.S., Lumsdaine, P.L., Ross, N.J., Selinger, P., Valiron, B.: Quipper: a scalable quantum programming language. SIGPLAN Not. **48**(6), 333–342 (2013). https://doi.org/10.1145/2499370.2462177
21. Harrow, A.W., Hassidim, A., Lloyd, S.: Quantum algorithm for linear systems of equations. Phys. Rev. Lett. **103**, 150502 (2009). https://doi.org/10.1103/PhysRevLett.103.150502
22. Heim, B., et al.: Quantum programming languages. Nat. Rev. Phys. **2**, 709–722 (2020). https://doi.org/10.1038/s42254-020-00245-7
23. Hietala, K., Rand, R., Hung, S.H., Wu, X., Hicks, M.: A verified optimizer for quantum circuits. Proc. ACM Program. Lang. **5**(POPL) (2021). https://doi.org/10.1145/3434318
24. Hsu, J.: Probabilistic couplings for probabilistic reasoning (2017)
25. Huang, Y., Martonosi, M.: Statistical assertions for validating patterns and finding bugs in quantum programs. In: Proceedings of the 46th International Symposium on Computer Architecture, pp. 541–553. ISCA '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3307650.3322213
26. Hung, S., Hietala, K., Zhu, S., Ying, M., Hicks, M., Wu, X.: Quantitative robustness analysis of quantum programs. Proc. ACM Program. Lang. **3**(POPL), 31:1–31:29 (2019). https://doi.org/10.1145/3290344
27. Kakutani, Y.: A logic for formal verification of quantum programs. In: Proceedings of the 13th Asian Conference on Advances in Computer Science: Information Security and Privacy, pp. 79–93. ASIAN'09, Springer-Verlag, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10622-4_7
28. Kitaev, A.Y., Shen, A.H., Vyalyi, M.N.: Classical and Quantum Computation. American Mathematical Society, USA (2002). https://doi.org/10.1090/gsm/047
29. Kitaev, A.Y.: Quantum measurements and the abelian stabilizer problem. Electron. Colloquium Comput. Complex. **TR96-003** (1996). https://eccc.weizmann.ac.il/eccc-reports/1996/TR96-003/index.html
30. Kundu, S., Tatlock, Z., Lerner, S.: Proving optimizations correct using parameterized program equivalence. SIGPLAN Not. **44**(6), 327–337 (2009). https://doi.org/10.1145/1543135.1542513
31. Lahiri, S.K., Sinha, R., Hawblitzel, C.: Automatic root causing for program equivalence failures in binaries. In: Kroening, D., Păsăreanu, C.S. (eds.) Computer Aided Verification, pp. 362–379. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-21690-4_21

32. Li, G., Zhou, L., Yu, N., Ding, Y., Ying, M., Xie, Y.: Projection-based runtime assertions for testing and debugging quantum programs. Proc. ACM Program. Lang. **4**(OOPSLA) (2020). https://doi.org/10.1145/3428218
33. Li, Y., Unruh, D.: Quantum relational hoare logic with expectations. In: Bansal, N., Merelli, E., Worrell, J. (eds.) 48th International Colloquium on Automata, Languages, and Programming (ICALP 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 198, pp. 136:1–136:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). https://doi.org/10.4230/LIPIcs.ICALP.2021.136, https://drops.dagstuhl.de/opus/volltexte/2021/14205
34. Lloyd, S., Mohseni, M., Rebentrost, P.: Quantum principal component analysis. Nat. Phys. **10**(9), 631–633 (2014). https://doi.org/10.1038/nphys3029
35. Lucanu, D., Rusu, V.: Program equivalence by circular reasoning. Form. Asp. Comput. **27**(4), 701–726 (2015). https://doi.org/10.1007/s00165-014-0319-6
36. Ming, J., Zhang, F., Wu, D., Liu, P., Zhu, S.: Deviation-based obfuscation-resilient program equivalence checking with application to software plagiarism detection. IEEE Trans Reliab. **65**(4), 1647–1664 (2016). https://doi.org/10.1109/TR.2016.2570554
37. Nechita, I., Puchała, Z., Pawela, L., Życzkowski, K.: Almost all quantum channels are equidistant. J. Math. Phys. **59**(5), 052201 (2018). https://doi.org/10.1063/1.5019322
38. Necula, G.C.: Translation validation for an optimizing compiler. SIGPLAN Not. **35**(5), 83–94 (2000). https://doi.org/10.1145/358438.349314
39. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, New York, 10th edn. (2011). https://doi.org/10.1017/CBO9780511976667
40. Paetznick, A., Svore, K.M.: Repeat-until-success: non-deterministic decomposition of single-qubit unitaries. Quantum Info. Comput. **14**(15-16), 1277–1301 (2014). https://doi.org/10.26421/QIC14.15-16-2
41. Pitts, A.: Operationally-Based Theories of Program Equivalence, pp. 241–298. Publications of the Newton Institute, Cambridge University Press, Cambridge (1997). https://doi.org/10.1017/CBO9780511526619.007
42. Preskill, J.: Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018). https://doi.org/10.22331/q-2018-08-06-79
43. Rand, R.: Verification logics for quantum programs (2019)
44. Rand, R., Paykin, J., Zdancewic, S.: QWIRE Practice: formal verification of quantum circuits in Coq. Electron. Proc. Theor. Comput. Sci. **266**, 119–132 (2018). https://doi.org/10.4204/eptcs.266.8
45. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 124–134. SFCS '94, IEEE Computer Society, USA (1994). https://doi.org/10.1109/SFCS.1994.365700
46. Sistla, M., Chaudhuri, S., Reps, T.: CFLOBDDs: context-free-language ordered binary decision diagrams (2023)
47. Smith, R.S., Curtis, M.J., Zeng, W.J.: A practical quantum instruction set architecture (2016). https://arxiv.org/abs/1608.03355
48. Svore, K., et al.: Q#: enabling scalable quantum computing and development with a high-level DSL. In: Proceedings of the Real World Domain Specific Languages Workshop 2018. RWDSL2018, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3183895.3183901

49. Tao, R., et al.: Giallar: push-button verification for the Qiskit quantum compiler. In: Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation, pp. 641–656. PLDI 2022, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3519939.3523431
50. Trabesinger, A.: Quantum computing: towards reality. Nature **543**, S1 (2017). https://doi.org/10.1038/543S1a
51. Unruh, D.: Quantum relational Hoare logic. Proc. ACM Program. Lang. **3**(POPL) (2019). https://doi.org/10.1145/3290346
52. Voichick, F., Li, L., Rand, R., Hicks, M.: Qunity: a unified language for quantum and classical computing. Proc. ACM Program. Lang. **7**(POPL) (2023). https://doi.org/10.1145/3571225
53. Watrous, J.: Simpler semidefinite programs for completely bounded norms. Chicago J. Theor. Comput. Sci. **2013**, 8 (2013). http://cjtcs.cs.uchicago.edu/articles/2013/8/contents.html
54. Weimer, W., Fry, Z.P., Forrest, S.: Leveraging program equivalence for adaptive program repair: models and first results. In: 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 356–366. ASE '13, IEEE Press, Silicon Valley, CA, USA (2013). https://doi.org/10.1109/ASE.2013.6693094
55. Xu, A., Molavi, A., Pick, L., Tannu, S., Albarghouthi, A.: Synthesizing quantum-circuit optimizers. Proc. ACM Program. Lang. **7**(PLDI) (2023). https://doi.org/10.1145/3591254
56. Xu, M., et al.: Quartz: superoptimization of quantum circuits. In: Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation, pp. 625–640. PLDI 2022, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3519939.3523433
57. Yan, P., Jiang, H., Yu, N.: On incorrectness logic for quantum programs. Proc. ACM Program. Lang. **6**(OOPSLA1) (2022). https://doi.org/10.1145/3527316
58. Yang, H.: Relational separation logic. Theor. Comput. Sci. **375**(1), 308–334 (2007). https://doi.org/10.1016/j.tcs.2006.12.036, https://www.sciencedirect.com/science/article/pii/S0304397506009261, festschrift for John C. Reynolds's 70th birthday
59. Ying, M.: Floyd–Hoare logic for quantum programs. ACM Trans. Program. Lang. Syst. **33**(6) (2012). https://doi.org/10.1145/2049706.2049708
60. Ying, M.: Foundations of Quantum Programming. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edn. (2016). https://doi.org/10.1016/C2014-0-02660-3
61. Ying, M., Duan, R., Feng, Y., Ji, Z.: Predicate Transformer Semantics of Quantum Programs, pp. 311–360. Cambridge University Press, Cambridge (2009). https://doi.org/10.1017/CBO9781139193313.009
62. Yu, N., Palsberg, J.: Quantum abstract interpretation. In: Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, pp. 542–558. PLDI 2021, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3453483.3454061
63. Yuan, C., McNally, C., Carbin, M.: Twist: sound reasoning for purity and entanglement in quantum programs. Proc. ACM Program. Lang. **6**(POPL) (2022). https://doi.org/10.1145/3498691

64. Zhou, L., Barthe, G., Hsu, J., Ying, M., Yu, N.: A quantum interpretation of bunched logic & quantum separation logic. In: Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science. LICS '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1109/LICS52264.2021.9470673
65. Zhou, L., Barthe, G., Strub, P.Y., Liu, J., Ying, M.: CoqQ: foundational verification of quantum programs. Proc. ACM Program. Lang. **7**(POPL) (2023). https://doi.org/10.1145/3571222
66. Zhou, L., Yu, N., Ying, M.: An applied quantum Hoare logic. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 1149–1162. PLDI 2019, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3314221.3314584
67. Zhou, L., Yu, N., Ying, S., Ying, M.: Quantum earth mover's distance, a no-go quantum Kantorovich-Rubinstein theorem, and quantum marginal problem. J. Math. Phys. **63**(10), 102201 (2022). https://doi.org/10.1063/5.0068344