Towards the Correctness of Quantum Program Implementation

Peng Yan

A Thesis presented to the Faculty of Engineering and Information Technology

to satisfy the requirements for the Degree of Doctor of Philosophy

University of Technology Sydney

March 2024

Dissertation Supervisors

Nengkun Yu

Sanjiang Li

Yulei Sui

Certificate of Original Authorship

I, Peng Yan, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature : Production Note:
Signature removed prior to publication.

Date : Sunday 31st March, 2024

For my family,

for my teachers.

Acknowledgements

I want to express my deep appreciation for my principal supervisor, Nengkun Yu, whose unwavering support and patient guidance made completing my dissertation possible. Despite lacking prior experience in quantum computing and program verification, his belief in my potential allowed me to pursue my Ph.D. at the University of Technology Sydney. As his first Ph.D. student, he showed immense patience, teaching me research methods and offering encouragement during challenging times, especially amidst the difficulties caused by the COVID-19 lockdown. I am grateful for his approachable nature; our casual conversations relieved my pressure significantly.

I am also indebted to my primary collaborator, Hanru Jiang, whose expertise in program verification and valuable insights on paper writing proved invaluable. His knowledge and patient guidance enriched our collaboration, and I cherished our working relationship. My heartfelt gratitude goes to Youming Qiao, Sanjiang Li, and Feng Yuan for their timely financial assistance, which sustained me during crucial times. I deeply appreciate their kindness and generosity, including the opportunity to work as a research assistant after my scholarship ended. Special thanks to my co-supervisor, Yulei Sui, for recommending me to Nengkun Yu and offering practical suggestions.

Furthermore, I am thankful for the companionship and joy my friends Gang Tang, Bing Chen, and Xing Hong provided during my time in Sydney. Their presence brightened my daily life and warmed my heart. I also thank my old friend, Hao Zhang, with whom I shared delightful moments playing online computer games. Lastly, I am profoundly grateful and apologetic to my taciturn parents. They are the most lovable person in the world.

Sydney, Australia

December 2023

Abstract

With the rapid advancement of quantum computing hardware, numerous quantum programming languages are continually proposed to facilitate the implementation of quantum algorithms. This research focuses on enhancing the correctness of quantum programs from the perspective of formal verification, given the counter-intuitive nature of quantum mechanics and environmental noise challenges. Two formal verification approaches are explored to validate the correctness of quantum programs.

The first approach extends classical incorrectness logic to the quantum domain, establishing a logical foundation for statically detecting bugs in quantum programming. The proposed quantum incorrectness logic is based on an intuitive explanation derived from a reachability standpoint. Based on under-approximation relations, the formulation of the incorrectness triple is found to be dual to quantum Hoare logic. The corresponding proof system is sound, complete, and decidable.

The second approach introduces approximate quantum coupling as a key tool for studying relational reasoning in quantum programs. This novel proof system generalizes the widely used approximate probabilistic coupling in probabilistic programs, addressing an open question regarding projective predicates in quantum contexts. The application of approximate relational logic aids in assessing the robustness of quantum programs between ideal specifications and imperfect implementations.

The practical utility of these approaches is demonstrated through case studies involving quantum teleportation, Grover's search, the repeat-until-success algorithm, the approximation of unitary gates, and the bit flip code. Notably, the formal verification of the low-depth approximation of the quantum Fourier transform is provided through approximate relational reasoning, showcasing the effectiveness of the proposed methodologies.

Contents

Chapter 1

Introduction

In recent years, quantum computing has earned significant attention from researchers as classical computers approach their physical hardware limits. Owing to some special quantum properties such as superposition, entanglements and no-cloning rule of quantum states, it has been proven that many quantum algorithms can significantly speed up the computational missions. Examples include Shor's factoring algorithm [Sho94], hidden subgroup problem [EHK04], Grover's search [Gro96], and HHL algorithm [HHL09]. The rapid advancement of quantum hardware [Tra17, ZWD+20, AAB+19] suggests that the realization of genuine quantum supremacy is just near the corner. The deployment of a reliable large-scale quantum computer with numerous qubits holds the potential to facilitate the practical implementation of quantum algorithms. Leading by the belief that quantum computers will play a pivotal role in the future, an increasing number of proposals for quantum programming languages and quantum computing platforms [AFD+12, GLR+13, SGT+18, WVMN19, Dev21, HSM+20, Gra05] continue to emerge for more straightforward implementation of quantum algorithms.

Program logic and verification can provide formal reasoning about the correctness of programs. The classical Floyd-Hoare logic, also known as Hoare logic, introduced by Floyd [Flo93] and Hoare [Hoa69] in 1969, constitutes a formula for the verification of program correctness. The Hoare triple, denoted as $\{P\}S\{Q\}$, states that the outputs of the execution of program S should satisfy postcondition Q if the inputs satisfy the precondition P. Typically, Hoare triples are employed to establish a set of axioms and inference rules, forming a proof system for a programming language. In recent years, considerable attention has been directed towards extending classical program verification techniques into the realm of quantum computing. One such extension is the Logic of Quantum Programs (LQP), a quantum dynamic logic proposed by Baltag and Smets [BS06]. LQP, as a direct extension of classical propositional logic, provides a formal syntax and relational semantics for describing various quantum operations, including unitary transformations, measurements, and entanglements between multi-systems. Another approach, presented by Brunet and Jorrand [BJ04], establishes a quantum logic based on the foundational ideas of Garrett Birkhoff and John von Neumann [BN36]. This logic utilizes closed Hilbert spaces to describe system properties and treats logical propositions as operations on state spaces. The extension of the usual propositional languages is achieved by introducing additional unary connectives to represent corresponding quantum unitary transformations and measurements.

As an extension of probabilistic Hoare logic, the quantitative state logic proposed by Chadha et al. [CMS06] was defined as an ensemble of logic enriched by attaching probabilities to pure quantum states. This logic established a sound and complete Hoare proof system under the condition of bounded iterations. In contrast to Chadha's approach, which employed distinct syntax and semantics for pure and mixed states, Selinger [Sel04a] defined the denotational semantics of the QPL language in terms of complete partial orders of super-operators that treat mixed and pure states uniformly. Later, Yoshihiko [Kak09] formulated rules for while programs in his sound but not complete Hoare-style proof system based on Selinger's denotational semantics. D'Hondt and Panangaden [DP06a] introduced the concept of weakest preconditions. Instead of Chadha's representations of probabilistic predicate transformations, they chose the expectation value of a quantum Hermitian operator as a quantum analog to the classical predicate. Subsequently, Ying [Yin12] utilized quantum weakest preconditions to construct a comprehensive Hoare-type logic that provided a sound and complete proof system for both partial and partial correctness for deterministic quantum programs. Zhou et al. [ZYY19] constrained quantum predicates as projections and simplified certain proof rules to facilitate the verification of correctness formulas. [BHY+19, Unr19] proposed the quantum relational Hoare logic based on quantum couplings to verify non-trivial relational properties of quantum programs. A quantum counterpart of separation logic was explored in [ZBH+21] based on the model of the substructural logic of bunched implications, where separable quantum states can be described as the separating conjunction of bunched implications.

This thesis builds upon prior research efforts on quantum Hoare logic and investigates two formal verification methodologies to validate the correctness of quantum programs, considering the perspectives of incorrectness logic and relational logic respectively. The incorrectness logic, introduced by [dVK11, O'H19], represents an emerging bug-detection technique emphasizing the static analysis of bug existence. Our approach takes an initial step towards establishing an incorrectness logic tailored for quantum programs, introducing novel concepts such as underapproximation and reachability analysis. Another approach addresses an unresolved question posed by [BHY+19], devising an approximate relational Hoare logic to robustly reason about the approximate relational properties of two quantum programs. The following sections provide detailed introductions to both approaches.

Incorrectness logic

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 the example programs of IBM's OpenQASM project [CJAA+21], Qiskit [WVMN19], and Rigetti's PyQuil project [SCZ16] in their official GitHub repositories. Huang and Martonosi [HM19a, HM19b] proposed a bug taxonomy based on their debugging experience with Scaffold [AFD+12, JPK+15]. A comprehensive study by [LZM+22] on bug fixing in four popular quantum programming languages (Qiskit, Cirq, Q#, and ProjectQ) revealed that a high proportion (over 80%) of bugs in quantum programs were quantum-specific.[1] Another study [PP22] gathered and inspected a set of 223 real-world bugs from 18 open-source quantum computing platforms and showed that a significant fraction of these bugs (39.9%) are quantum-specific. Theories and techniques for debugging are in urgent demand.

There are two lines of approaches aiming at debugging quantum programs. One approach is dynamic assertions [LBZ20, LZY+20], which detect erroneous states via quantum measurements at the cost of additional qubits and quantum operations at run-time. The other is statistical assertions [HM19b, HM19a] that detect errors via statistical tests over sampled simulations. Unfortunately, both approaches suffer from two main limitations:

- *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 [LZY+20].
- *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 logic like quantum Hoare logic [Yin12] and applied quantum Hoare logic (aQHL) [ZYY19] seem to be a good choice since they facilitate static reasoning and provide soundness and completeness guarantees. How-

[1]Generally, fixing quantum-specific bugs requires the domain knowledge of quantum-related concepts, properties, computational formulas, and quantum programming languages.

ever, 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,

$$\neg(\{P\}S\{Q\}) \not\Rightarrow \{P\}S\{\neg Q\},$$

which means true bugs could be missed.

An approach that addressed the aforementioned issues in the classical world is the incorrectness logic [O'H19] (IL), an under-approximate analogy of Hoare logic used for reasoning about bugs. Specifications in IL are of the form

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

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 [BGOS18, GOS19, DFLO19] 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 [RBD+20], 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 how to explain achieving (reaching everything described by) a quantum predicate.

In Chapter 3, we expand on the idea of IL by using projection-based quantum predicates from the quantum logic [BN36]. This approach has proven successful in reasoning about the correctness [ZYY19] of quantum programs and designing dynamic assertions [LZY+20]. The main result is a sound and complete logic system for reasoning about bugs in quantum programs 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'H19]. 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 3.6 and find our triple the best in expressiveness and efficiency.
- A sound and complete $\underline{\text{q}}$uantum $\underline{\text{i}}$ncorrectness $\underline{\text{l}}$ogic (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 quantum teleportation (5.1), Grover's algorithm (5.2), and a repeat-until-success program (5.3). In these examples, we introduce and reason about two types of bugs mentioned in Huang et al. [HM19b]. We also developed a prototyped static analyzer built on top of our proof rules to automate the reasoning.

Relational Hoare logic

Compared with many program verifications that traditionally focus on proving properties of single program execution, relational verification aims to prove the relation between two program executions. A specific instance is program equivalence [CE79, BTT82, Pit97], where two programs can have the same behavior, even though they may be implemented using different algorithms or programming languages. Program equivalence holds significant importance in various areas of computer science, including software engineering [WFF13, MZW+16, LSH15], translation validation of compilers [Nec00], program optimization [KTL09], and program analysis [CPSA19, BALR20, LR15]. In program analysis, program equivalence is used to verify programs' correctness or prove that two programs are functionally equivalent. This verification is typically done by analyzing the program's control flow, data flow, or other characteristics to determine whether the program has the same behavior as another program. To verify the program equivalence, one has to consider the general relationship beyond the equivalence during the program analysis because the implementation of the two programs could be very different.

Relational verification aims to prove the relational properties between two programs. A simple approach is to reason each program and compare the relation between the output. However, this methodology may be resource-consuming. For instance, one must consider the output corresponding to each input to reason about the equivalence between two programs. Expressing the property using Hoare-style relational logic is much more convenient. 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 [Ben04], where relational Hoare logic (RHL) predicates are 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 [BGZB09], probabilistic relational Hoare logic (pRHL) lifted the predicates into relations over probabilistic distributions on memories. It was deployed to provide formal computer-aided verification of cryptographic proofs. Furthermore, [BKOZB13] introduced extra parameters to allow approximate lifting of relations to distributions. To be specific, the authors defined the judgments in approximate probabilistic relational Hoare logic (apRHL) of the form

$$c_1 \sim_{\alpha,\delta} c_2 : \Psi \Rightarrow \Phi$$

with parameters α, δ to reason about differential privacy.

Among techniques in program analysis and verification, the *exact* relational logic for quantum programs attracts lots of attention [BHY+19, Unr19, LU21]. Relational logic provides a more expressive approach to characterize the relation between two programs. For instance, direct verification of the equivalence between two 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 any 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. [2]

However, none of the aforementioned 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) [Pre18] era. This insight sheds light on the importance of modeling hardware with noise.
- 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 [CW00], which achieves a lower circuit depth approximation of the exact quantum Fourier transform used in Shor's celebrated algorithm [Sho94].

[2] Details refers to Definition 4.9 and Example 4.14.

As for approximate reasoning in quantum settings, [ZYY19] discussed the robustness of quantum programs by introducing the concept of approximate satisfaction of predicates, [HHZ+19] 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 approximate relational properties between quantum programs.

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 [BHY+19]. 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 [Hsu17] are no longer true [LBZ20].
- 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.

- Additionally, 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 Chapter 4, 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. And the 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.

The main result is a sound logic system for approximate relational reasoning for quantum programs. 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" 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 (5.6) of the low-depth approximate quantum Fourier transform (QFT) with an error bound that is asymptotically equivalent to the one in [CW00]. Implementing QFT is a significant step in the development of quantum algorithms such as period finding [Sho94], HHL algorithm [HHL09] and quantum principal component analysis [LMR14]. In the example of appropriate decomposition of unitary gates (5.5), 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. We also use aqRHL to verify the correctness of bit flip code against an arbitrary single-qubit error (5.4).

Organization of the thesis

The thesis has a straightforward organizational structure. In Chapter 2, fundamental preliminaries are presented, including key concepts and notations related to quantum computing, the syntax and semantics of Ying's quantum while-program language [Yin12], and quantum predicates. Chapter 3 initiates our exploration of quantum incorrectness logic (QIL). It starts with explaining the motivations for characterizing quantum errors and subsequently introduces modifications to Ying's language for characterizing abnormal terminations. The under-approximation relation and incorrectness triple are introduced to establish a sound and complete proof system. Chapter 4 covers the investigation of approximate quantum relational logic (aqRHL). It introduces the concepts of quantum approximate couplings and lifting, utilizing them to approximate measurement conditions and relational judgments. Consequently, a proof system is developed based on the specification of aqRHL judgments. Chapter 5 showcases six concrete examples, illustrating how our logic can be applied to reason about the correctness of quantum programs. Chapter 6 reviews some related work and outline prospective directions for future studies.

Chapter 2

Preliminaries

2.1 Background of Quantum Computing

This section presents an essential background of quantum computation to make this thesis self-contained. Necessary notations and definitions are also given in this section. If necessary, readers can find more details in the textbook [NC11].

Quantum states

For any isolated physical system, its *state space* is known as a complex vector space with inner product, that is, a Hilbert space $\mathcal{H}$. This thesis only considers finite-dimensional Hilbert space $\mathcal{H}$. Given a quantum system on a register $\bar{q}$ that consists of n qubits, its corresponding Hilbert space, denoted by $\mathcal{H}_{\bar{q}}$, is essentially the complex vector space $\mathbb{C}^{2^n}$. We use a subscript $\bar{q}$ in $\mathcal{H}_{\bar{q}}$ to denote the register $\bar{q}$ of concern and omit it when it is evident from context. The number of qubits in register $\bar{q}$ is denoted by $|\bar{q}|$, which equals the dimension $\mathrm{Dim}(\mathcal{H}_{\bar{q}})$ of the corresponding Hilbert space $\mathcal{H}_{\bar{q}}$.

A quantum system can be completely described by its *state vector*. The Dirac notation $|\cdot\rangle$ is usually used to denote the unit complex vector in the system's Hilbert space $\mathcal{H}$. The simplest quantum system is the *qubit*, a quantum analog to the classical bit. A qubit has a two-dimensional Hilbert space. The most important orthonormal basis of one-qubit system is the *computational basis* $\{|0\rangle, |1\rangle\}$ with

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

, which encode the classical bits 0 and 1, respectively. The vector dual of $|\psi\rangle$, denoted by $\langle\varphi|$, is the Hermitian conjugate of $|\varphi\rangle$. The inner product of two vector states $|\psi\rangle$ and $|\varphi\rangle$ is denoted by $\langle\psi|\varphi\rangle$, and the outer-product of two vector states $|\psi\rangle$ and $|\varphi\rangle$ is denoted by $|\psi\rangle\langle\varphi|$.

The vector state $|\psi\rangle$ is also called as a *pure state*. Moreover, a *mixed state* can be represented by a classical distribution over an ensemble of pure states $(p_i, |\psi_i\rangle)$. That is, the system is in vector state $|\psi_i\rangle$ with probability p_i, where $p_i \in (0, 1]$ and $\sum_i p_i = 1$. The *density operator* or *density matrix* can widely represent the pure or mixed quantum state. A density operator ρ for the ensemble of pure states $(p_i, |\psi_i\rangle)$ is a positive semidefinite operator $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$. One may notice that there might be multiple ensembles that have the same density operator representation. In this case, we do not distinguish these quantum states because they are not physically distinguishable. Particularly, a pure state $|\psi\rangle$ can be represented by the density operator $|\psi\rangle\langle\psi|$. A positive semidefinite operator ρ is said to be a *density operator* if $\mathrm{Tr}(\rho) = 1$, and *partial density operator* if $\mathrm{Tr}(\rho) \leq 1$, where $\mathrm{Tr}(\rho)$ denotes the trace of the density matrix ρ. 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 invalid or impossible state.

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

Let two independent quantum registers $\bar{q}_1$ and $\bar{q}_2$ be in quantum state $\rho_1 \in \mathcal{D}(\mathcal{H}_{\bar{q}_1})$ and $\rho_2 \in \mathcal{D}(\mathcal{H}_{\bar{q}_2})$ respectively, then the composite system $\bar{q} = \{\bar{q}_1, \bar{q}_2\}$ is in the state $\rho_1 \otimes \rho_2 \in \mathcal{D}(\mathcal{H}_{\bar{q}_1} \otimes \mathcal{H}_{\bar{q}_2}) = \mathcal{D}(\mathcal{H}_{\bar{q}})$. We usually write the tensor product state $|\psi\rangle \otimes |\varphi\rangle$ as $|\psi\varphi\rangle$, $|\psi_1\rangle\langle\psi_2| \otimes |\varphi_1\rangle\langle\varphi_2|$ as $|\psi_1\varphi_1\rangle\langle\psi_2\varphi_2|$ for short. On the contrary, the notion of *partial trace* is a handy tool when we want to describe the subsystem of a composite quantum system. Formally, the partial trace over $\mathcal{H}_1$ is a mapping $\mathrm{Tr}_1(\cdot)$ from operators in $\mathcal{H}_1 \otimes \mathcal{H}_2$ to operators in $\mathcal{H}_1$ defined as

$$\mathrm{Tr}_1(|\varphi_1\rangle\langle\psi_1| \otimes |\varphi_2\rangle\langle\psi_2|) = \langle\psi_1|\varphi_1\rangle \cdot |\varphi_2\rangle\langle\psi_2|$$

for any $|\psi_1\rangle, |\varphi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle, |\varphi_2\rangle \in \mathcal{H}_2$. Similarly, the partial trace $\mathrm{Tr}_2(\cdot)$ over $\mathcal{H}_2$ can be defined symmetrically. Suppose the composite system $\bar{q}$ is in the state ρ, then the state of subsystem $\bar{q}_1$ and $\bar{q}_2$ can be described by $\mathrm{Tr}_2(\rho)$ and $\mathrm{Tr}_1(\rho)$ respectively.

If a state of the composite system $\bar{q} = \{\bar{q}_1, \bar{q}_2\}$ can be represented in the form $|\psi\rangle_{\bar{q}_1} \otimes |\varphi\rangle_{\bar{q}_2}$, it is called a *separable state*, or *product state*. If the state is inseparable, it is called an *entangled state*. If a composite system is in an entangled state, quantum measurement on one component system will also have effects on the other one and collapse the entanglement. Let $\bar{q}_1$ and $\bar{q}_2$ both be one-qubit systems. The Bell states are the most famous entangled states of a two-qubit system.

$$\begin{aligned} |\Phi_{00}\rangle &= (|00\rangle + |11\rangle)/\sqrt{2} \qquad & |\Phi_{10}\rangle &= (|00\rangle - |11\rangle)/\sqrt{2} \\ |\Phi_{01}\rangle &= (|01\rangle + |10\rangle)/\sqrt{2} \qquad & |\Phi_{11}\rangle &= (|01\rangle - |10\rangle)/\sqrt{2} \end{aligned}$$

These four Bell states form an orthonormal basis of the two-qubit system. Take Bell state $|\Phi_{00}\rangle$ for example,

$$|\Phi_{00}\rangle\langle\Phi_{00}| = \frac{1}{2}(|0\rangle\langle 0| \otimes |0\rangle\langle 0| + |0\rangle\langle 1| \otimes |0\rangle\langle 1| + |1\rangle\langle 0| \otimes |1\rangle\langle 0| + |1\rangle\langle 1| \otimes |1\rangle\langle 1|)$$

we have $\mathrm{Tr}_1(|\Phi_{00}\rangle\langle\Phi_{00}|) = \mathrm{Tr}_2(|\Phi_{00}\rangle\langle\Phi_{00}|) = \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|) = I/2$, i.e. two subsystems are both in a maximally mixed state $I/2$. If we measure the first qubit in the computational basis, that is, $\mathcal{M} = \{M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|\}$, we have

$$M_0\,|\Phi_{00}\rangle = |00\rangle \qquad M_1\,|\Phi_{00}\rangle = |11\rangle$$

It is clear that both subsystems collapse into either state $|00\rangle$ or $|11\rangle$ simultaneously, and the composite system is not entangled anymore.

Unitary operation

The *evolution* of an *isolated* quantum system can be characterized by *unitary operators*. 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 $U^\dagger U = UU^\dagger = I$. $U^\dagger$ denotes the Hermitian conjugate of matrix U, i.e. the transpose of the complex conjugate of matrix U. When the unitary operation U is applied to the register $\bar{q}$ in a quantum system, it changes the quantum state ρ of the whole system 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 a single-qubit unitary operator U to the q-th qubit in an n-qubit system, we have

$$U_q = \otimes_{1\le i\le q-1} I \otimes U \otimes_{q+1\le j\le n} I.$$

We often write U_q as U for short, omit the subscript q in U_q when it is clear from the context.

We introduce some commonly used unitary operators (also called *gates*) that would arise in further discussion. Their matrix representations are shown as follows,

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

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

The behavior of these operators can also be described by its transformations 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 $H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Apart from $\{|0\rangle, |1\rangle\}$, $\{|+\rangle, |-\rangle\}$ is another widely used orthogonal basis for one-qubit system. These two bases can be transformed to each other by applying the Hadamard operator. The gate $P(\theta)$ is a general phase gate with $P(\theta)|0\rangle = |0\rangle$ and $P(\theta)|1\rangle = e^{i\theta}|1\rangle$. Another important class is muti-qubit controlled gates. A controlled unitary gate takes qubit q_1 as the control qubit and applies unitary operator U to qubit q_2 if qubit q_1 is in the state $|1\rangle$. For example, the CNOT gate performs the transition $|b_1\rangle_{q_1} \otimes |b_2\rangle_{q_2} \mapsto |b_1\rangle_{q_1} \otimes |b_1 \oplus b_2\rangle_{q_2}$, where $b_1 \oplus b_2$ denotes the logical XOR of b_1 and b_2. The *c-$P(\theta)$* gate perform the transition $|b_1\rangle_{q_1} \otimes |b_2\rangle_{q_2} \mapsto (e^{i\theta})^{b_1 \cdot b_2} |b_1\rangle_{q_1} \otimes |b_2\rangle_{q_2}$, where $b_1 \cdot b_2$ denotes binary multiplication of b_1 and b_2.

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 $p_m \neq 0$.

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

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

The corresponding observable of projective measurement $M = \{P_m\}$ is $\sum_m mP_m$, where P_m is the projector on the eigenspace of $\sum_m mP_m$ with eigenvalue m. 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.

There is a known fact that any general quantum measurement $M = \{M_m\}$ can always be implemented by a projective measurement $M' = \{P_m\}$ with the help of introducing a unitary U and ancillary system. Specifically, we need to introduce a unitary U such that $U|\psi\rangle|0\rangle = \sum_m M_m|\psi\rangle|m\rangle$, where $|0\rangle$ is in the ancillary system. Let $P_m = I \otimes |m\rangle\langle m|$, thus we have $P_m U|\psi\rangle|0\rangle = M_m|\psi\rangle|m\rangle$. Therefore, our examples in this thesis would only consider projective measurements for briefness.

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 $P = |+\rangle\langle+| = \frac{1}{2}\left(\begin{smallmatrix}1 & 1\\ 1 & 1\end{smallmatrix}\right)$, 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 $P = |0\rangle\langle 0| + |1\rangle\langle 1| = I$ [1].

Fig. 2.1 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 $P^\perp$ maps the state ρ into $P^\perp\rho P^\perp$ that lies in the subspace $P^\perp$, where $P^\perp$ is the orthogonal complement of P, i.e. $P^\perp = I - P$. 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 $|1\rangle\langle 1|/2$.

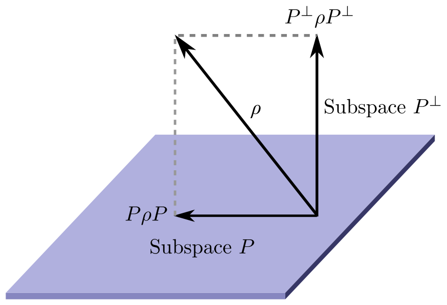

Figure 2.1: Projection P on state ρ.

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

Superoperator

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. Readers may refer to the system-environment model in Sec.8.2 [NC11] for more details. The following Kraus representation theorem for superoperator $\mathcal{E}$ is necessary to denote quantum noise in our further discussions.

Theorem 2.1. *The superoperator $\mathcal{E}$ is a completely-positive map over partial density operators from $\mathcal{D}(\mathcal{H}_1)$ to $\mathcal{D}(\mathcal{H}_2)$ if and only if for any $\rho \in \mathcal{D}(\mathcal{H}_1)$, we have*

$$\mathcal{E}(\rho) = \textstyle\sum_k E_k \rho E_k^\dagger$$

for a set of Kraus operators $\{E_k : \mathcal{H}_1 \to \mathcal{H}_2\}$ with $\sum_k E_k E_k^\dagger \leq I$. The Kraus representation of $\mathcal{E}$ is also written as $\mathcal{E} = \sum_k E_k \cdot E_k^\dagger$ for short.

Apparently, both quantum unitary operations and measurements alone can be seen as special cases of superoperators. It is important to note that the superoperator $\mathcal{E}$ is trace non-increasing, as $\mathrm{Tr}(\mathcal{E}(\rho)) \leq \mathrm{Tr}(\rho)$ holds for any ρ. We say that a superoperator $\mathcal{E}$ is trace-preserving if $\sum_k E_k E_k^\dagger = I$, also written as $\bar{\mathcal{E}}$ explicitly. Additionally, we use $\mathcal{E}_2 \circ \mathcal{E}_1$ to denote the composition of superoperators $\mathcal{E}_1$ and $\mathcal{E}_2$, namely, $\mathcal{E}_2 \circ \mathcal{E}_1(\rho) = \mathcal{E}_2(\mathcal{E}_1(\rho))$. Besides, we use tensor products $\mathcal{E}_1 \otimes \mathcal{E}_2$ to represent the combination of two superoperators $\mathcal{E}_1 \in \mathcal{H}_1$ and $\mathcal{E}_2 \in \mathcal{H}_2$ over the composite subspace $\mathcal{H}_1 \otimes \mathcal{H}_2$. The concept of the *dual* of a superoperator $\mathcal{E}$, denoted by $\mathcal{E}^*$, will also be useful in our further reasoning.

Definition 2.2. The Schrödinger-Heisenberg dual of a superoperator $\mathcal{E} : \mathcal{D}(\mathcal{H}_1) \to \mathcal{D}(\mathcal{H}_2)$, denoted by $\mathcal{E}^*$, is the mapping on any operator A from $\mathcal{H}_2$ to $\mathcal{H}_1$ such that

$$\mathrm{Tr}(\mathcal{E}^*(A)\rho) = \mathrm{Tr}(A\mathcal{E}(\rho))$$

for any $\rho \in \mathcal{D}(\mathcal{H}_1)$, where we have the Kraus representation $\mathcal{E}^* = \sum_k E_k^\dagger \cdot E_k$.

2.2 Quantum Programming Language

In this section, we will review the quantum while-language that we will use our logic to reason. We provide a brief review of the syntax and semantics of this language, and readers who require more information about it can refer to [Yin12]. We define *Var* as a set of quantum variables and use $Var(S)$ to represent the set of all quantum variables present in a quantum program S. Moreover, we use $\mathcal{H}_S = \otimes_{q \in Var(S)} \mathcal{H}_q$ to represent Hilbert spaces of all the quantum variables in program S. The syntax presented in Def. 2.3 is slightly different from [Yin12]. We add the statement $\bar{q} := \mathcal{E}[\bar{q}]$ for general quantum operations, and rewrite the quantum measurement statement in the form of the conditional statement.

Definition 2.3 (Syntax). The following syntax defines the quantum while-programs in [Yin12]:

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

The statement **skip** denotes doing nothing. The statement $q := |0\rangle$ initializes a qubit q to the vector state $|0\rangle$ and keeps other qubits untouched. Notice that there is no analogy for classical initialization expression (i.e., $x := e$) due to the no-cloning theorem of quantum states. The statement $\bar{q} := U[\bar{q}]$ applies a unitary transformation U on the quantum register $\bar{q}$. Similarly, statement $\bar{q} := \mathcal{E}[\bar{q}]$ applies a superopertor $\mathcal{E}$ on the quantum register $\bar{q}$. The conditional statement **if** $(\square m \cdot M[\bar{q}] = m \longrightarrow S_m)$ **fi** performs a quantum measurement M on the register $\bar{q}$, and executes subprogram S_m according to the measurement outcome m. Furthermore, the 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 re-enters the loop correspondingly. It is important to note that measurement has side effects on quantum states; therefore, the branches/loop bodies will start with a collapsed state after measurement.

We model the operational semantics by labeled transitions over program *configurations* 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

$$(Stmt \times \mathcal{D}(\mathcal{H})) \times ((Stmt \cup \{\downarrow\}) \times \mathcal{D}(\mathcal{H}))$$

, where $\downarrow$ denotes the termination of a program by convention. The operational semantics [Yin12] of the quantum while-language are presented in Fig. 2.2a.

$$\begin{array}{ll}
(\textsc{Skip})\ \langle \mathbf{skip}, \rho\rangle \rightarrow \langle \downarrow, \rho\rangle & (\textsc{In})\ \langle q := |0\rangle, \rho\rangle \rightarrow \langle \downarrow, \sum_n |0\rangle_q\langle n|\rho|n\rangle_q\langle 0|\rangle \\
(\textsc{Ut})\ \langle \bar{q} := U\bar{q}, \rho\rangle \rightarrow \langle \downarrow, U\rho U^\dagger\rangle & (\textsc{Qo})\ \langle \bar{q} := \mathcal{E}[\bar{q}], \rho\rangle \rightarrow \langle \downarrow, \mathcal{E}(\rho)\rangle \\
(\textsc{Seq1})\ \dfrac{\langle S_1, \rho\rangle \rightarrow \langle \downarrow, \rho'\rangle}{\langle S_1; S_2, \rho\rangle \rightarrow \langle S_2, \rho'\rangle} & (\textsc{Seq2})\ \dfrac{\langle S_1, \rho\rangle \rightarrow \langle S_1', \rho'\rangle}{\langle S_1; S_2, \rho\rangle \rightarrow \langle S_1'; S_2, \rho'\rangle}
\end{array}$$

$$\begin{array}{l}
(\textsc{If})\ \langle \mathbf{if}\ (\square m \cdot M[\bar{q}] = m \rightarrow S_m)\ \mathbf{fi}, \rho\rangle \rightarrow \langle S_m, M_m\rho M_m^\dagger\rangle \\
(\textsc{Lp1})\ \langle \mathbf{while}\ M[\bar{q}] = 1\ \mathbf{od}\ S\ \mathbf{od}, \rho\rangle \rightarrow \langle \downarrow, M_0\rho M_0^\dagger\rangle \\
(\textsc{Lp2})\ \langle \mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}, \rho\rangle \rightarrow \langle S; \mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}, M_1\rho M_1^\dagger\rangle
\end{array}$$

(a) Operational semantics

$$\begin{array}{ll}
(\textsc{Skip})\ [\![\mathbf{skip}]\!](\rho) = \rho & (\textsc{In})\ [\![q := |0\rangle]\!](\rho) = \sum_n |0\rangle_q\langle n|\rho|n\rangle_q\langle 0| \\
(\textsc{Ut})\ [\![\bar{q} := U[\bar{q}]]\!](\rho) = U\rho U^\dagger & (\textsc{Qo})\ [\![\bar{q} := \mathcal{E}[\bar{q}]]\!](\rho) = \mathcal{E}(\rho) \\
(\textsc{Seq})\ [\![S_1; S_2]\!](\rho) = [\![S_2]\!] \circ [\![S_1]\!] &
\end{array}$$

$$\begin{array}{l}
(\textsc{If})\ [\![\mathbf{if}\ (\square m \cdot M[\bar{q}] = m \rightarrow S_m)\ \mathbf{fi}]\!](\rho) = \sum_m \mathcal{M}_m \circ S_m = \sum_m [\![S_m]\!](M_m\rho M_m^\dagger) \\
(\textsc{Lp})\ [\![\mathbf{while}\ M[\bar{q}] = 1\ \mathbf{od}\ S\ \mathbf{od}]\!](\rho) = \sum_{k=0}^{\infty} \mathcal{M}_0 \circ ([\![S]\!] \circ \mathcal{M}_1)^k \\
\mathcal{M}_m(\rho) = M_m\rho M_m^\dagger \quad (\mathcal{R}_1 + \mathcal{R}_2)\rho = \mathcal{R}_1\rho + \mathcal{R}_2\rho \quad \mathcal{R}^0\rho = \rho \quad \mathcal{R}^n = \mathcal{R}^{n-1} \circ \mathcal{R} \\
\mathcal{R}_2 \circ \mathcal{R}_1 = \{\rho'' \mid \rho' = \mathcal{R}_1\rho \wedge \rho'' = \mathcal{R}_2\rho' \wedge \rho, \rho', \rho'' \in \mathcal{D}(\mathcal{H})\}
\end{array}$$

(b) Denotational semantics

Figure 2.2: Semantics of quantum while-language

Here we provide some necessary clarifications regarding statements that differ from classical ones. The In statement initializes a variable q in state ρ to $|0\rangle\langle 0|$ while leaving other variables unchanged. The initialization can be characterized by a superoperator $\mathcal{E} = \sum_n |0\rangle_q\langle n| \cdot (|0\rangle_q\langle n|)^\dagger$, where $|\psi\rangle_q\langle\varphi|$ denote the outer product of the vector states $|\psi\rangle$ and $|\varphi\rangle$ in $\mathcal{H}_q$. In the Ut statement, the unitary operation U over the register $\bar{q}$ performs the transition $\rho \mapsto U\rho U^\dagger$ for any state $\rho \in \mathcal{H}_{\bar{q}}$. If we want to set a variable q to an arbitrary pure state $|\psi\rangle$, we usually initialize q to $|0\rangle$ and then apply a proper unitary U such that $U|0\rangle = |\psi\rangle$. Quantum measurements are necessary to set a variable to a mixed state. In 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. Similar to the If statement, the Lp1 statement indicates the termination of the loop when the measurement outcome is 0, and the Lp2 statement explains the case when the measurement outcome is 1.

The denotational semantics [Yin12] of the quantum while-language are presented in Fig. 2.2b. By convention, we use $[\![S]\!]$ to denote the semantic function of a program S. In the Lp rule, $\mathcal{M}_0 \circ ([\![S]\!] \circ \mathcal{M}_1)^k$ denotes the k-th unrolling of the loop statement **while** $M[\bar{q}] = 1$ **od** S **od**. The following lemma demonstrates the equivalence between denotational semantics and operational semantics in Fig. 2.2. It is straightforward to observe that the denotational semantics of program S with input state ρ equals the statistical sum of all feasible output states.

Lemma 2.4 ([Yin12]). *For any quantum program S defined in Fig.2.2, we have*

$$[\![S]\!](\rho) = \sum\{|\rho' : \langle S, \rho\rangle \to \langle\downarrow, \rho'\rangle|\}$$

where is $\to^*$ the reflexive and transitive closure of $\to$, and $\{|\cdot|\}$ denotes a multi-set.

Lemma 2.5 ([Yin12]). *For any quantum while program S defined in Fig.2.2, its denotational semantics function $[\![S]\!] : \mathcal{D}(\mathcal{H}) \mapsto \mathcal{D}(\mathcal{H})$ is a superoperator.*

2.3 Quantum Predicate

There are typically two kinds of quantum predicates in the literature. The one we currently do not study is by D'Hondt and Panangaden [DP06b], where they proposed to use a positive Hermitian operator whose maximum eigenvalue is bounded by 1 to serve as a quantum predicate. This constraint allows to establish a complete partial order on the poset $(\mathcal{P}(\mathcal{H}), \leq)$, where $\mathcal{P}(\mathcal{H})$ denotes the set of all predicates on Hilbert space $\mathcal{H}$. In contrast to the classical viewpoint, the satisfaction of a quantum predicate P by the quantum state ρ is now represented by an expectation value $\mathrm{Tr}(P\rho)$ that lies within the range $[0, 1]$. From a classical point of view, such predicates are less intuitive since they discuss the expectation instead of a yes/no answer about whether a quantum state satisfies a predicate.

In this thesis, we follow the other class of quantum predicates proposed by [BN36], that is, projections. Projections can be viewed as a trade-off between expressiveness and practicability. As shown in applied quantum Hoare logic [ZYY19], run-time assertions [LZY+20] and quantum relational logic [Unr19], 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 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 affect quantum states satisfying the predicates.

Although projections can not model all types of bugs, they have a relatively high performance in capturing specific bugs that enlarge/shrink the subspaces of correct states.

Before discussing projective quantum predicates formally, we introduce the support of positive semi-definite matrices (including density matrices) in Def. 2.6.

Definition 2.6 (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., $\text{supp}(A) = \text{span}\{|\psi_i\rangle\}$.

In particular, the support of a projection P is its corresponding subspace, so we write P as a shorthand of the subspace $\text{supp}(P)$ 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 2.7 (Satisfaction). A quantum state ρ satisfies a projection P, denoted by $\rho \vDash P$, if $\text{supp}(\rho) \subseteq P$. Contrarily, $\rho \nvDash P$ if $\text{supp}(\rho) \nsubseteq P$.

Formally, when using projections as predicates, a mixed quantum state $\rho = \sum p_i\rho_i$ satisfying the projection P means that every ρ_i satisfies the projection P. On the contrary, for any quantum state ρ_i satisfying the projection P, any kind of ensemble of $\{q_i, \rho_i\}$ also satisfies the projection P. The identity operator, I, is the largest projection, corresponding to the entire state space $\mathcal{H}$. The smallest projection is the $\mathbf{0}$-operator, instead of the empty set. When interpreted as subspaces, any other projection P on $\mathcal{H}$ has $\mathbf{0} \subseteq P \subseteq I$.

Logical operations on projections are different from their classical counterparts. We list the definitions of $\neg$, $\wedge$, and $\vee$ in Def. 2.8, where we use projections to denote both the quantum predicates and their corresponding subspaces.

Definition 2.8. The logical operations for quantum predicates are defined as follows. For any two quantum predicates P, Q on $\mathcal{H}$,

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

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.[2]

2.4 Quantum Program Example

A demonstrating quantum program example is given in this subsection for a better understanding of basic concepts about quantum programs. We will revisit and reason about this example in Chapter 5. We take a simple program implementing a repeat-until-success [PS14, BRS15] (RUS) algorithm as an example. RUS algorithms offer exact, fault-tolerant 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 a while loop.

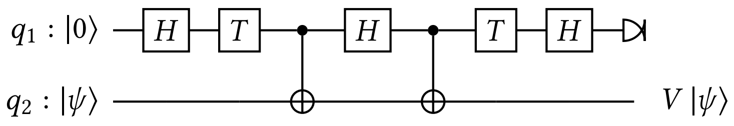

Figure 2.3: An RUS circuit implementing $V = \frac{I+i\sqrt{2}X}{\sqrt{3}}$.

Fig. 2.3 is the smallest RUS circuit for the loop body found in [PS14] that implements non-Clifford single-qubit unitaries. The qubit q_1 is the auxiliary qubit, and q_2 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_1 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

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

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 simultaneously.

```
1 $q_1 := |0\rangle\,; q_1 := X[q_1];$
2 $q_2 := |0\rangle\,; q_2 := W[q_2];$
3 while $M[q_1] = 1$ do
4     $q_1 := X[q_1]; q_1 := H[q_1]; q_1 := T[q_1]; (q_1, q_2) := CNOT[(q_1, q_2)];$
5     $q_1 := H[q_1]; (q_1, q_2) := CNOT[(q_1, q_2)]; q_1 := T[q_1]; q_1 := H[q_1];$
6 od;
```

Figure 2.4: Example Program S_{RUS} for RUS algorithm

We can encode the whole process into the quantum program S_{RUS} defined in Fig 2.4. To make sure the program enters the while-loop body, The auxiliary qubit q_1 is set to $|1\rangle$ using an X gate at line 1, and it restores to $|0\rangle$ by an X gate again before executing the RUS circuit in the loop body. The target qubit q_2 is set to an initial state $|\psi\rangle$ at line 2 by the unitary gate W such that $W|0\rangle = |\psi\rangle$. Now we verify how the program S_{RUS} simulates the unitary V on qubit q_2. According to the semantics defined in Fig. 2.2, we have the following transitions for the loop statement in program S_{RUS}.

$$\text{(1st iteration)} \quad \langle \textbf{while}, |1\rangle\langle 1| \otimes |\psi\rangle\langle\psi|\rangle \longrightarrow \langle\downarrow, |\varphi\rangle\langle\varphi|\rangle$$

$$\text{(k-th iteration)} \quad \langle \textbf{while}, \frac{1}{4^{k-2}}|\varphi\rangle\langle\varphi|\rangle \rightarrow \begin{cases} \langle\downarrow, \frac{3}{4^{k-1}}|0\rangle\langle 0| \otimes V|\psi\rangle\langle\psi|V^\dagger\rangle \\ \langle \textbf{while}, \frac{1}{4^{k-1}}|\varphi\rangle\langle\varphi|\rangle \end{cases}$$

Here the term **while** denotes the loop statement between lines 3-6 in Fig 2.4 for short. The unitary $U = (X \otimes I - i\sqrt{3}I \otimes V)/2$ characterizes the sequence of unitary computations in Fig. 2.3 and $|\varphi\rangle = U|0\rangle \otimes |\psi\rangle$. It is direct to see that the state $|\varphi\rangle$ is invariant for the loop statement. The denotational semantics can be denoted as

$$[\![S_{RUS}]\!](\rho) = \sum_{k=2}^{\infty} \frac{3}{4^{k-1}}|0\rangle\langle 0| \otimes V|\psi\rangle\langle\psi|V^\dagger = |0\rangle\langle 0| \otimes V|\psi\rangle\langle\psi|V^\dagger$$

for any input ρ on qubits q_1 and q_2. Therefore, the program S_{RUS} outputs the desired state $V|\psi\rangle$ on qubit q_2 as long as the measurement outcome on qubit q_1 is 0.

2.5 Ancillary Lemmas

We end this chapter with some necessay ancillary lemmas that would be used in our further proof. This section can be safely skiped if readers are not interested.

Definition 2.9. For any subspace S of finite-dimensional Hilbert space $\mathcal{H}$, its orthogonal complement is $S^{\perp}$ such that

$$S^{\perp} = \{|\varphi\rangle \mid \langle\psi|\varphi\rangle = 0, \forall\, |\psi\rangle \in S\}$$

Lemma 2.10. *For any subspace S of the finite-dimensional Hilbert $\mathcal{H}$, we have $S = S^{\perp\perp}$. Moreover, we have $S_1 \subseteq S_2 \Leftrightarrow S_1^{\perp} \supseteq S_2^{\perp}$.*

Proof. Choose an orthonormal basis of $\mathcal{H}$, says $\{e_1, \dots, e_n\}$ with $\{e_1, \dots, e_m\}$ being an orthonormal basis of subspace S where $n \geq m$. We have

$$\begin{aligned} S^{\perp} &= \operatorname{span}(\{e_1, \cdots, e_m\})^{\perp} \\ &= \{w \mid \langle e_i|w\rangle = 0, \forall e_i \in S, w \in \mathcal{H}\} \\ &= \{w \mid \textstyle\sum a_j \langle e_i|e_j\rangle = 0, w = \sum a_j e_j, \forall e_i \in S, e_j \in \mathcal{H}\} \\ &= \{w \mid w = \textstyle\sum_{j>m} a_j e_j\} \quad = \operatorname{span}(\{e_{m+1}, \cdots, e_n\}) \end{aligned}$$

And we can also have $\operatorname{span}(\{e_{m+1}, \cdots, e_n\})^{\perp} = S$ in the same way, i.e., $S = S^{\perp\perp}$.

If $S_1 \subseteq S_2$, then we assume an orthonormal basis of S_1 to be $\{e_1, \dots, e_m\}$, and an orthonormal basis of S_2 to be $\{e_1, \dots, e_m, \dots, e_n\}$ $(n \geq m)$. Let $\{e'_1, \dots, e'_k\}$ be a basis of the subspace $S_2^{\perp}$, then we have $\langle e'_i|e_j\rangle = 0$ $(1 \leq i \leq k, 1 \leq j \leq n)$ by definition of orthogonal complement. Then we also have that any $|e'_i\rangle$ must be in the subspace $S_1^{\perp}$ since $\langle e'_i|e_j\rangle = 0$ holds for all elements of the basis of S_1, which means the basis of $S_1^{\perp}$ includes the basis of $S_2^{\perp}$, i.e $S_1^{\perp} \supseteq S_2^{\perp}$. ∎

Lemma 2.11. *For a positive semi-definite matrix A in the finite-dimensional Hilbert space $\mathcal{H}$,*

$$\operatorname{supp}(A) = \{|\psi\rangle \mid \langle\psi|A|\psi\rangle = 0, |\psi\rangle \in \mathcal{H}\}^{\perp}.$$

Proof. Let $A = \sum \lambda_i |\psi_i\rangle\langle\psi_i|$ be the spectral decomposition with $\lambda_i > 0$. Then

$$\mathrm{supp}(A) = \mathrm{span}\{|\psi_i\rangle\}$$

where $\{|\psi_i\rangle\}$ is a basis of $\mathrm{supp}(A)$. Assume $\{|\varphi_i\rangle\}$ is a basis of $\mathrm{supp}(A)^\perp$, then we have $\langle\psi_i|\varphi_j\rangle = 0$ by Def 2.9. Then for any $|\psi\rangle \in \mathcal{H}$, we have $|\psi\rangle = \sum a_i |\psi_i\rangle + \sum b_j |\varphi_j\rangle$.

$$\begin{aligned}
&\{|\psi\rangle \mid \langle\psi|A|\psi\rangle = 0, |\psi\rangle \in \mathcal{H}\}\\
&= \{|\psi\rangle \mid \textstyle\sum a_i \langle\psi|A|\psi_i\rangle = 0\}\\
&= \{|\psi\rangle \mid \textstyle\sum a_i\lambda_i \langle\psi|\psi_i\rangle = 0, \lambda_i > 0\}\\
&= \{|\psi\rangle \mid \textstyle\sum |a_i|^2\lambda_i = 0, \lambda_i > 0\}\\
&= \{|\psi\rangle \mid |\psi\rangle = \textstyle\sum b_j |\varphi_j\rangle\} \quad (a_i = 0)\\
&= \mathrm{supp}(A)^\perp.
\end{aligned}$$

Then Lemma 2.10 implies

$$\mathrm{supp}(A) = \{|\psi\rangle \mid \langle\psi|A|\psi\rangle = 0, |\psi\rangle \in \mathcal{H}\}^\perp.$$

■

Lemma 2.12. *For positive semi-definite matrices A,B in the finite-dimensional Hilbert space $\mathcal{H}$, we have*

$$\mathrm{supp}(A) \supseteq \mathit{supp}(B) \ \ \textit{iff}\ \exists r > 0 \text{ s.t. } A \geq rB$$

In particular, for any $p > 0$ and positive semi-definite matrices A,

$$\mathrm{supp}(A) = \mathrm{supp}(pA)$$

Proof. Necessity: If there exists $r > 0$ such that $A \geq rB \geq 0$, we have

$$\begin{aligned}
&A \geq rB \geq 0\\
\Rightarrow\ &\forall |\psi\rangle \in \mathcal{H},\ \langle\psi|A|\psi\rangle \geq r\langle\psi|B|\psi\rangle \geq 0\\
\Rightarrow\ &\forall |\psi\rangle \in \mathcal{H},\ \langle\psi|A|\psi\rangle = 0 \Rightarrow \langle\psi|B|\psi\rangle = 0
\end{aligned}$$

$$\Rightarrow \{|\psi\rangle \mid \langle\psi|A|\psi\rangle = 0, |\psi\rangle \in \mathcal{H}\} \subseteq \{|\psi\rangle \mid \langle\psi|B|\psi\rangle = 0, |\psi\rangle \in \mathcal{H}\}$$

$$\Rightarrow \{|\psi\rangle \mid \langle\psi|A|\psi\rangle = 0, |\psi\rangle \in \mathcal{H}\}^\perp \supseteq \{|\psi\rangle \mid \langle\psi|B|\psi\rangle = 0, |\psi\rangle \in \mathcal{H}\}^\perp \quad \text{(Lemma 2.10)}$$

$$\Rightarrow \operatorname{supp}(A) \supseteq \operatorname{supp}(B) \quad \text{(Lemma 2.11)}$$

Sufficiency: Assume the spectral decomposition of A is $A = \sum_i \lambda_i |\psi_i\rangle\langle\psi_i|, \lambda_i > 0$. Let P_A be the projection on the support of A, and then we have $P_A = \sum_i |\psi_i\rangle\langle\psi_i|$. Then there exits $r_1 > 0$ and $r_2 > 0$ such that $r_1 P_A \geq A \geq r_2 P_A$ if we choose $r_1 = \max\{\lambda_i\}$ and $r_2 = \min\{\lambda_i\}$.

Now assume there exits $r_A > 0$ and $r_B > 0$ such that $A \geq r_A P_A$ and $r_B P_B \geq B$, where P_B denotes the projection on the support of B. Thus we have,

$$\begin{aligned} \operatorname{supp}(A) \supseteq \operatorname{supp}(B) \;&\Rightarrow\; P_A \geq P_B \\ \Rightarrow\; \frac{1}{r_A} A \geq P_A \geq P_B \geq \frac{1}{r_B} B \;&\Rightarrow\; A \geq \frac{r_A}{r_B} B \end{aligned}$$

Particularly, we have $\operatorname{supp}(A) \supseteq \operatorname{supp}(pA)$ with $r = 1/p$ and $\operatorname{supp}(pA) \supseteq \operatorname{supp}(A)$ with $r = p$, thus we have $\operatorname{supp}(A) = \operatorname{supp}(pA)$ for any $p > 0$. ■

Lemma 2.13. *For positive semi-definite matrices A, B, C and D in the finite-dimensional Hilbert space $\mathcal{H}$, we have*

$$\operatorname{supp}(A) \supseteq \operatorname{supp}(B),\ \operatorname{supp}(C) \supseteq \operatorname{supp}(D) \;\Rightarrow\; \operatorname{supp}(A + C) \supseteq \operatorname{supp}(B + D)$$

Particularly, we have

$$\operatorname{supp}(A) = \operatorname{supp}(B),\ \operatorname{supp}(C) = \operatorname{supp}(D) \;\Rightarrow\; \operatorname{supp}(A + C) = \operatorname{supp}(B + D)$$

Proof. By Lemma 2.12, there exits $r_1 > 0$ and $r_2 > 0$ such that $A \geq r_1 B$ and $C \geq r_2 D$. Let $r = \min(r_1, r_2) > 0$, we have $A \geq r_1 B \geq rB$ and $C \geq r_2 D \geq rD$. Thus we have $A + C \geq r(B + D)$, i.e. $\operatorname{supp}(A + C) \supseteq \operatorname{supp}(B + D)$ by Lemma 2.12. The particular case is direct since $\operatorname{supp}(A) = \operatorname{supp}(B) \;\Leftrightarrow\; (\operatorname{supp}(A) \supseteq \operatorname{supp}(B)) \wedge (\operatorname{supp}(A) \subseteq \operatorname{supp}(B))$. ■

Lemma 2.14. *For any super-operator $\mathcal{E}$ and $\rho, \sigma \in \mathcal{D}(\mathcal{H})$, we have*

$$\operatorname{supp}(\rho) \supseteq \operatorname{supp}(\sigma) \Rightarrow \operatorname{supp}(\mathcal{E}(\rho)) \supseteq \operatorname{supp}(\mathcal{E}(\sigma)).$$

Particularly, we have

$$\operatorname{supp}(\rho) = \operatorname{supp}(\sigma) \Rightarrow \operatorname{supp}(\mathcal{E}(\rho)) = \operatorname{supp}(\mathcal{E}(\sigma))$$

Proof.

$$\begin{aligned}
& \operatorname{supp}(\rho) \supseteq \operatorname{supp}(\sigma) \\
\Rightarrow\ & \exists r > 0.\, \rho - r\sigma \geq 0 \quad (\text{Lemma } 2.12) \\
\Rightarrow\ & \exists r > 0.\, E_i(\rho - r\sigma)E_i^\dagger \geq 0 \\
\Rightarrow\ & \exists r > 0.\, \sum E_i(\rho - r\sigma)E_i^\dagger \geq 0 \\
\Rightarrow\ & \exists r > 0.\, \sum E_i\rho E_i^\dagger \geq r \sum E_i\sigma E_i^\dagger \\
\Rightarrow\ & \operatorname{supp}(\sum E_i\rho E_i^\dagger) \supseteq \operatorname{supp}(\sum E_i\sigma E_i^\dagger) \quad (\text{Lemma } 2.12) \\
\Rightarrow\ & \operatorname{supp}(\mathcal{E}(\rho)) \supseteq \operatorname{supp}(\mathcal{E}(\sigma))
\end{aligned}$$

The particular case is direct since $\operatorname{supp}(A) = \operatorname{supp}(B) \Leftrightarrow (\operatorname{supp}(A) \supseteq \operatorname{supp}(B)) \wedge (\operatorname{supp}(A) \subseteq \operatorname{supp}(B))$. ■

Lemma 2.15. *For a projection P and a matrix K, we have*

$$\forall \sigma \vDash \operatorname{supp}(KPK^\dagger) \;\Rightarrow\; \exists \rho \vDash P.\, \sigma = K\rho K^\dagger$$

where $\rho, \sigma \in \mathcal{D}(\mathcal{H})$.

Proof. Assume an orthonormal basis of the support of projection P is $\{|e_i\rangle\}$, then $\{K|e_i\rangle\}$ is also the basis of the support of matrix $KPK^\dagger$. Assume the spectral decomposition of σ is $\sigma = \sum \lambda_i |\psi_i\rangle\langle\psi_i|$, $|\psi_i\rangle\langle\psi_i| \vDash \operatorname{supp}(KPK^\dagger)$, then we have $|\psi_i\rangle = \sum a_{ij}K|e_j\rangle$. Let $\sum a_{ij}|e_j\rangle = \mu_i|\varphi_i\rangle$ with $\langle\varphi_i|\varphi_i\rangle = 1$. Then,

$$\sigma = \sum \lambda_i |\psi_i\rangle\langle\psi_i| = \sum \lambda_i |\mu_i|^2 K|\varphi_i\rangle\langle\varphi_i| K^\dagger.$$

It is clear that $|\varphi_i\rangle\langle\varphi_i| \vDash P$. We can set

$$\rho = \sum \lambda_i |\mu_i|^2 \, |\varphi_i\rangle\langle\varphi_i| \vDash P.$$

■

Lemma 2.16. *A composition of super-operators is also a super-operator.*

Proof. Given any two super-operators $\mathcal{E}$ and $\mathcal{F}$, we have

$$\begin{aligned}(\mathcal{F} \circ \mathcal{E})(\rho) &= \mathcal{F}(\mathcal{E}(\rho)) = \mathcal{F}(\sum E_i \rho E_i^\dagger) \\ &= \sum F_j (\sum E_i \rho E_i^\dagger) F_j^\dagger \\ &= \sum (F_j E_i) \rho (F_j E_i)^\dagger = \sum K_{ij} \rho K_{ij}^\dagger = \mathcal{K}(\rho)\end{aligned}$$

where $K_{ij} = F_j E_i$, $\mathcal{K} = \mathcal{F} \circ \mathcal{E}$ is a super-operator. Thus the composition of super-operators is also a super-operator. ■

Lemma 2.17. *For any two positive semi-definite matrices A and B in the finite-dimensional Hilbert space $\mathcal{H}$, we have*

$$(\mathrm{supp}(A) \vee \mathrm{supp}(B))^\perp = \mathrm{supp}(A)^\perp \wedge \mathrm{supp}(B)^\perp$$

Proof. Let $\{|e_i\rangle\}$ be the orthonormal basis of the Hilbert space $\mathcal{H}$, then we can find two orthonormal bases $\{|a_i\rangle\}$ and $\{|b_i\rangle\}$ of the support of matrices A and B respectively, where $\{|a_i\rangle\} \subseteq \{|e_i\rangle\}$ and $\{|b_i\rangle\} \subseteq \{|e_i\rangle\}$. By the definition of the operation $\wedge$ and $\vee$ on the subspace, we have

$$\begin{aligned}\mathrm{supp}(A) \vee \mathrm{supp}(B) &= \mathrm{span}(\{|a_i\rangle\} \cup \{|b_i\rangle\}) \\ \mathrm{supp}(A) \wedge \mathrm{supp}(B) &= \mathrm{span}(\{|a_i\rangle\} \cap \{|b_i\rangle\})\end{aligned}$$

Then we have

$$\begin{aligned}(\mathrm{supp}(A) \vee \mathrm{supp}(B))^\perp &= \mathrm{span}(\{|a_i\rangle\} \cup \{|b_i\rangle\})^\perp \\ &= \mathrm{span}(\{|e_i\rangle\} \backslash (\{|a_i\rangle\} \cup \{|b_i\rangle\}))\end{aligned}$$

$$
\begin{aligned}
&= \operatorname{span}((\{|e_i\rangle\}\backslash\{|a_i\rangle\}) \cap (\{|e_i\rangle\}\backslash\{|b_i\rangle\})) \\
&= \operatorname{span}(\{|e_i\rangle\}\backslash\{|a_i\rangle\}) \wedge \operatorname{span}(\{|e_i\rangle\}\backslash\{|b_i\rangle\}) \\
&= \operatorname{supp}(A)^\perp \wedge \operatorname{supp}(B)^\perp
\end{aligned}
$$

where we use the fact $\operatorname{supp}(A)^\perp = \operatorname{span}(\{|e_i\rangle\}\backslash\{|a_i\rangle\})$ from the definition of support and operation $\perp$ on the subspace. ■

Lemma 2.18. *For any two positive semi-definite matrices A and B in the finite-dimensional Hilbert space $\mathcal{H}$, we have*

$$\operatorname{supp}(A) \vee \operatorname{supp}(B) = \operatorname{supp}(A + B)$$

Particularly, we have $\operatorname{supp}(A) \vee \operatorname{supp}(B) = \operatorname{supp}(r_1 A + r_2 B)$ *for any* $r_1 > 0$ *and* $r_2 > 0$.

Proof.

$$
\begin{aligned}
&(\operatorname{supp}(A) \vee \operatorname{supp}(B))^\perp = \operatorname{supp}(A)^\perp \wedge \operatorname{supp}(B)^\perp \qquad \text{(Lemma 2.17)} \\
&= \{|\psi\rangle \mid \langle\psi|A|\psi\rangle = 0, |\psi\rangle \in \mathcal{H}\} \wedge \{|\psi\rangle \mid \langle\psi|B|\psi\rangle = 0, |\psi\rangle \in \mathcal{H}\} \\
&= \{|\psi\rangle \mid \langle\psi|A|\psi\rangle = 0 \wedge \langle\psi|B|\psi\rangle = 0, |\psi\rangle \in \mathcal{H}\} \\
&= \{|\psi\rangle \mid \langle\psi|A + B|\psi\rangle = 0, |\psi\rangle \in \mathcal{H}\} \quad = \operatorname{supp}(A + B)^\perp
\end{aligned}
$$

where we use the fact that $\langle\psi|A|\psi\rangle = 0 \wedge \langle\psi|B|\psi\rangle = 0 \iff \langle\psi|A + B|\psi\rangle = 0$ since $\langle\psi|A|\psi\rangle \geq 0$ and $\langle\psi|B|\psi\rangle \geq 0$ for any $|\psi\rangle \in \mathcal{H}$. Thus we have $\operatorname{supp}(A) \vee \operatorname{supp}(B) = \operatorname{supp}(A + B)$ by Lemma 2.10. The particular case is direct since $\operatorname{supp}(A) = \operatorname{supp}(pA)$ for any $p > 0$ by Lemma 2.12. ■

Lemma 2.19. *For any density operator ρ_i and projection P_i in the finite-dimensional Hilbert space $\mathcal{H}$, if $\rho_i \dashv P_i$ holds for every $i \in \mathbb{Z}^+$, then we have $\sum \rho_i \dashv \vee P_i$.*

Proof. By Lemma 2.18, we have $\vee P_i = \operatorname{supp}(\sum_i P_i)$. And given it $\rho_i \dashv P_i$ for every i, it is direct to have $\operatorname{supp}(\sum_i \rho_i) \supseteq \operatorname{supp}(\sum_i P_i)$ by Lemma 2.13, i.e. $\sum \rho_i \dashv \vee P_i$. ■

Lemma 2.20. *For any program S defined in Fig. 3.2, there are super-operators $\mathcal{E}_i$ such that for any $\rho \in \mathcal{D}(\mathcal{H}_S)$,*

$$[\![S]\!]_\epsilon(\rho) = \{(\mathcal{E}_i(\rho), \lambda_i) \mid i \in \mathbb{Z}^+\}.$$

where each $\mathcal{E}_i$ corresponds to any path of S described by the transition $\langle S, \rho\rangle \xrightarrow{\epsilon}{}^{*} \langle\downarrow, \mathcal{E}_i(\rho)\rangle$, and λ_i is the counting number of such paths.

Proof. We proceed by induction on the structure of S and case study on ϵ.

(1) Case $S = \textbf{error}$. We have $\mathcal{E}(\rho) = I\rho I^\dagger$ for $[\![S]\!]_{er} = \{|(\rho,\rho)|\}$, and $\mathcal{E}(\rho) = \mathbf{0}\rho\mathbf{0}^\dagger$ for $[\![S]\!]_{ok} = \{|(\rho,\mathbf{0})|\}$.

(2) Case $S = \textbf{skip}$. We have $\mathcal{E}(\rho) = I\rho I^\dagger$ for $[\![S]\!]_{ok} = \{|(\rho,\rho)|\}$, and $\mathcal{E}(\rho) = \mathbf{0}\rho\mathbf{0}^\dagger$ for $[\![S]\!]_{er} = \{|(\rho,\mathbf{0})|\}$.

(3) Case $S = q := |0\rangle$. We have $\mathcal{E}(\rho) = \sum E_i\rho E_i^\dagger$ for $[\![S]\!]_{ok} = \{|(\rho, \sum |0\rangle_q\langle n|\rho|n\rangle_q\langle 0|)|\}$ with $E_i = |0\rangle_q\langle i|$, and $\mathcal{E}(\rho) = \mathbf{0}\rho\mathbf{0}^\dagger$ for $[\![S]\!]_{er} = \{|(\rho,\mathbf{0})|\}$.

(4) Case $S = \bar{q} := U\bar{q}$. We have $\mathcal{E}(\rho) = U\rho U^\dagger$ for $[\![S]\!]_{ok} = \{|(\rho, U\rho U^\dagger)|\}$, and $\mathcal{E}(\rho) = \mathbf{0}\rho\mathbf{0}^\dagger$ for $[\![S]\!]_{er} = \{|(\rho,\mathbf{0})|\}$.

(5) Case $S = S_1; S_2$. By the induction hypothesis on S_1 and S_2, we have

$$[\![S_1]\!]_\epsilon(\rho) = \{|(\mathcal{E}_i(\rho), n_i)|\} \qquad [\![S_2]\!]_\epsilon(\rho) = \{|(\mathcal{F}_j(\rho), m_j)|\}$$

For the $[\![S_1]\!]_{ok}$ case, by Fig. 2.2b we have

$$\begin{aligned}
([\![S_1]\!]_{ok} \circ [\![S_2]\!]_\epsilon)(\rho) &= \{(\rho'', n * m) \mid ((\rho,\rho'), n) \in [\![S_1]\!]_{ok} \text{ and } ((\rho',\rho''), m) \\
&\quad \in [\![S_2]\!]_\epsilon\} \\
&= \{((\mathcal{F}_j \circ \mathcal{E}_i)(\rho), n_i * m_j) \mid \forall i.\ ((\rho, \mathcal{E}_i(\rho)), n_i) \in [\![S_1]\!]_{ok} \text{ and } \forall j.\ ((\mathcal{E}_i(\rho), \\
&\quad (\mathcal{F}_j \circ \mathcal{E}_i)(\rho)), m_j) \in [\![S_2]\!]_\epsilon\} \\
&= \{((\mathcal{F}_j \circ \mathcal{E}_i)(\rho), n_i * m_j) \mid \forall i,j.\ ((\rho, (\mathcal{F}_j \circ \mathcal{E}_i)(\rho)), n_i * m_j) \in [\![S_1]\!]_{ok} \circ [\![S_2]\!]_\epsilon\}
\end{aligned}$$

$= \{\forall i, j.\, (\mathcal{K}_{ij}(\rho), n_i * m_j)\}$ (Lemma 2.16)

where $\mathcal{K}_{ij} = \mathcal{F}_j \circ \mathcal{E}_i$ is also a super-operator.

By Fig. 2.2b, we have shown $[\![S]\!]_{ok}$ and $[\![S_1]\!]_{ok} \circ [\![S_2]\!]_{er}$. For the *er* case,

$$[\![S]\!]_{er} = [\![S_1]\!]_{ok} \circ [\![S_2]\!]_{er} \uplus [\![S_1]\!]_{er},$$

we only need to show the $[\![S_1]\!]_{er}$ case, that is

$$[\![S]\!]_{er}(\rho) = \{(\rho', n_i) \mid ((\rho, \rho'), n_i) \in [\![S_1]\!]_{er}\} = \{(\mathcal{E}_i(\rho_1), n_i)\}$$

(6) Case $S = \textbf{if}\ (\square m \cdot M[\bar{q}] = m \longrightarrow S_m)\ \textbf{fi}$. By the induction hypothesis on S_m, we have

$$[\![S_m]\!]_\epsilon(\rho) = \{(\mathcal{E}_{m_i}(\rho), \lambda_{m_i}) \mid i, m_i \in \mathbb{Z}^+\}$$

By Fig. 2.2b, we have $[\![S]\!]_\epsilon = \uplus_m(\mathcal{M}_m \circ [\![S_m]\!]_\epsilon)$, then

$$[\![S]\!]_\epsilon(\rho) = \{(\rho', \lambda_{m_i}) \mid (\rho, \rho_m) \in \mathcal{M}_m \text{ and } \forall m, i.\, ((\rho_m, \rho'), \lambda_{m_i}) \in [\![S_m]\!]_\epsilon\}$$
$$= \{(\rho', \lambda_{m_i}) \mid \forall m, i.\, \rho_m = M_m \rho M_m^\dagger, \rho' = \mathcal{E}_{m_i}(\rho_m)\}$$
$$= \{\forall m, i.\, ((\mathcal{E}_{m_i} \circ \mathcal{M}_m)(\rho), \lambda_{m_i})\} \quad = \{\forall m, i.\, (\mathcal{K}_{m_i}(\rho), \lambda_{m_i})\} \text{ (Lemma 2.16)}$$

where $\mathcal{K}_{m_i} = \mathcal{E}_{m_i} \circ \mathcal{M}_m$ is also a super-operator.

(7) Case $S = \textbf{while}\ M[\bar{q}] = 1\ \textbf{do}\ S'\ \textbf{od}$. By the induction hypothesis on S', we have

$$[\![S']\!]_{ok}(\rho) = \{(\mathcal{E}_i(\rho), l_i)\} \qquad [\![S']\!]_{er}(\rho) = \{(\mathcal{F}_j(\rho), m_j)\}$$

To make it compact, we define

$$\textbf{while}^n_{ok} := (\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_0$$
$$\textbf{while}^n_{er} := (\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_1 \circ [\![S']\!]_{er}.$$

For the *ok* case, by Fig. 2.2b we have $[\![S]\!]_{ok} = \biguplus_{n \in \mathbb{N}} \textbf{while}^n_{ok}$, it suffices to show that the lemma holds for $\textbf{while}^n_{ok}$ for any $n \in \mathbb{N}$.

$$\textbf{while}^n_{ok}(\rho) = ((\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_0)(\rho)$$

$$
\begin{aligned}
&= \{(\rho_n', \textstyle\prod_{k=1}^n l_{i_k}) \mid \forall i_1 \ldots i_n.\, (\rho_0, \rho_0') \in \mathcal{M}_1.\, ((\rho_0', \rho_1), l_{i_1}) \in [\![S']\!]_{ok}, \cdots, \\
&\quad ((\rho_{n-1}', \rho_n), l_{i_n}) \in [\![S']\!]_{ok}, (\rho_n, \rho_n') \in \mathcal{M}_0\} \\
&= \{(\rho_n', \textstyle\prod_{k=1}^n l_{i_k}) \mid \forall i_1 \ldots i_n.\, \rho_0' = M_1\rho_0 M_1^\dagger, \rho_1 = \mathcal{E}_{i_1}(\rho_0'), \cdots, \\
&\quad \rho_n = \mathcal{E}_{i_n}(\rho_{n-1}'), \rho_n' = M_0 \rho_n M_0^\dagger\} \\
&= \{\forall i_1 \ldots i_n.\, ((\mathcal{M}_0 \circ \mathcal{E}_{i_n} \circ \mathcal{M}_1 \cdots \mathcal{E}_{i_1} \circ \mathcal{M}_1)(\rho_0), \textstyle\prod_{k=1}^n l_{i_k})\}
\end{aligned}
$$

where $\mathcal{M}_0 \circ \mathcal{E}_{i_n} \circ \mathcal{M}_1 \cdots \mathcal{E}_{i_1} \circ \mathcal{M}_1$ is also a super-operator.

For the *er* case, by Fig. 2.2b we have $[\![S]\!]_{er} = \biguplus_{n\in\mathbb{N}} \mathbf{while}_{er}^n$. Similarly, it suffices to show that the lemma holds for $\mathbf{while}_{er}^n$ for any $n \in \mathbb{N}$.

$$
\begin{aligned}
&\mathbf{while}_{er}^n(\rho) = ((\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ (\mathcal{M}_1 \circ [\![S']\!]_{er}))(\rho) \\
&= \{(\rho_0, m_j * \textstyle\prod_{k=1}^n l_{i_k}) \mid \forall i_1 \ldots i_n, j.\, (\rho_0, \rho_0') \in \mathcal{M}_1, ((\rho_0', \rho_1), l_{i_1}) \in [\![S']\!]_{ok}, \cdots, \\
&\quad ((\rho_{n-1}', \rho_n), l_{i_n})\} \in [\![S']\!]_{ok}, (\rho_n, \rho_n') \in \mathcal{M}_1, ((\rho_n', \rho_{n+1}), l_{m_j}) \in [\![S']\!]_{er}\} \\
&= \{(\rho_0, m_j * \textstyle\prod_{k=1}^n l_{i_k}) \mid \forall i_1 \ldots i_n, j.\, \rho_0' = M_1\rho_0 M_1^\dagger, \rho_1 = \mathcal{E}_{i_1}(\rho_0'), \cdots, \\
&\quad \rho_n' = M_1 \rho_n M_1^\dagger, \rho_{n+1} = \mathcal{F}_j(\rho_n')\} \\
&= \{\forall i_1 \ldots i_n, j.\, ((\mathcal{F}_j \circ \mathcal{M}_1 \circ \mathcal{E}_{i_n} \circ \mathcal{M}_1 \cdots \mathcal{E}_{i_1} \circ \mathcal{M}_1)(\rho_0), m_j * \textstyle\prod_{k=1}^n l_{i_k})\}
\end{aligned}
$$

where $\mathcal{F}_j \circ \mathcal{M}_1 \circ \mathcal{E}_{i_n} \circ \mathcal{M}_1 \cdots \mathcal{E}_{i_1} \circ \mathcal{M}_1$ is also a super-operator.

■

Lemma 2.21. *For any quantum program S_1, S_2 and any $\rho \in \mathcal{D}(\mathcal{H}_S)$, we have*

$$\textstyle\sum([\![S_1]\!]_{ok} \circ [\![S_2]\!]_\epsilon)(\rho) = \sum [\![S_2]\!]_\epsilon(\sum [\![S_1]\!]_{ok}(\rho))$$

Proof. Lemma 2.21 comes from the linearity of super-operator. We proceed to prove the following fact first. Let $\rho_1, \rho_2 \in \mathcal{D}(\mathcal{H}_S)$ and $\lambda_1, \lambda_2 \geq 0$. If $\lambda_1\rho_1 + \lambda_2\rho_2 \in \mathcal{D}(\mathcal{H}_S)$, then for any program S we have,

$$[\![S]\!]_\epsilon(\lambda_1\rho_1 + \lambda_2\rho_2) = \{(\lambda_1\mathcal{E}_i(\rho_1) + \lambda_2\mathcal{E}_i(\rho_2), n_i) \mid ((\rho, \mathcal{E}_i(\rho)), n_i) \in [\![S]\!]_\epsilon, \rho \in \mathcal{D}(\mathcal{H}_S)\}$$

For any super-operator $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$, it is direct to see

$$\mathcal{E}(\lambda_1\rho_1 + \lambda_2\rho_2) = \textstyle\sum_k E_k(\lambda_1\rho_1 + \lambda_2\rho_2)E_k^\dagger = \lambda_1 \sum_k E_k\rho_1 E_k^\dagger + \lambda_2 \sum_k E_k \rho_2 E_k^\dagger$$

$$= \lambda_1 \mathcal{E}(\rho_1) + \lambda_2 \mathcal{E}(\rho_2)$$

Thus this fact is directly derivable from Lemma 2.20, where the whole operation on the input state through any path of our quantum program can be viewed as a super-operator. Thus we have

$$\sum[\![S]\!]_\epsilon(\lambda_1\rho_1 + \lambda_2\rho_2) = \textstyle\sum_i n_i * (\lambda_1\mathcal{E}_i(\rho_1) + \lambda_2\mathcal{E}_i(\rho_2))$$
$$= \lambda_1 \textstyle\sum_i n_i * \mathcal{E}_i(\rho_1) + \lambda_2 \sum_i n_i * \mathcal{E}_i(\rho_2) = \lambda_1 \sum[\![S]\!]_\epsilon(\rho_1) + \lambda_2 \sum[\![S]\!]_\epsilon(\rho_2)$$

Thus for $[\![S_1]\!]_{ok} \circ [\![S_2]\!]_\epsilon$, we have

$$([\![S_1]\!]_{ok} \circ [\![S_2]\!]_\epsilon)(\rho) = \{(\rho''_{ij}, \alpha_i * \beta_{ij}) \mid ((\rho, \rho'_i), \alpha_i) \in [\![S_1]\!]_{ok}, ((\rho'_i, \rho''_{ij}), \beta_{ij}) \in [\![S_2]\!]_\epsilon\}$$

by our semantics, then apply $\sum[\![S]\!]_\epsilon(\lambda_1\rho_1 + \lambda_2\rho_2) = \lambda_1 \sum[\![S]\!]_\epsilon(\rho_1) + \lambda_2 \sum[\![S]\!]_\epsilon(\rho_2)$ to have

$$\sum[\![S_2]\!]_\epsilon(\textstyle\sum[\![S_1]\!]_{ok}(\rho)) = \sum[\![S_2]\!]_\epsilon(\sum_i \alpha_i\rho'_i) = \sum_i \alpha_i(\sum[\![S_2]\!]_\epsilon(\rho'_i)) = \sum_i \alpha_i(\sum_j \beta_{ij}\rho''_{ij})$$
$$= \textstyle\sum_{i,j} \alpha_i\beta_{ij}\rho''_{ij} = \sum([\![S_1]\!]_{ok} \circ [\![S_2]\!]_\epsilon)(\rho)$$

thus we have $\sum([\![S_1]\!]_{ok} \circ [\![S_2]\!]_\epsilon)(\rho) = \sum[\![S_2]\!]_\epsilon(\sum[\![S_1]\!]_{ok}(\rho))$. ■

Lemma 2.22. *Let $\rho, \sigma \in \mathcal{D}(\mathcal{H}_S)$ and $\text{supp}(\rho) \supseteq \text{supp}(\sigma)$, then for any program S, we have $\text{supp}(\sum[\![S]\!]_\epsilon(\rho)) \supseteq \text{supp}(\sum[\![S]\!]_\epsilon(\sigma))$. Particularly, we have $\text{supp}(\sum[\![S]\!]_\epsilon(\rho)) = \text{supp}(\sum[\![S]\!]_\epsilon(\sigma))$ if $\text{supp}(\rho) = \text{supp}(\sigma)$.*

Proof. By the Lemma 2.20, we have

$$[\![S]\!]_\epsilon(\rho) = \{.(\mathcal{E}_i(\rho), \lambda_i) \mid i \in \mathbb{Z}^+\}.$$

where $\mathcal{E}_i$ is a super-operator. Notice that we have $\text{supp}(\mathcal{E}_i(\rho)) \supseteq \text{supp}(\mathcal{E}_i(\sigma))$ if $\text{supp}(\rho) \supseteq \text{supp}(\sigma)$ by Lemma 2.14, then

$$\text{supp}(\textstyle\sum[\![S]\!]_\epsilon(\rho)) = \text{supp}(\sum_i \lambda_i\mathcal{E}_i(\rho)) = \vee_i \text{supp}(\mathcal{E}_i(\rho)) \quad \text{(Lemma 2.18)}$$
$$\supseteq \vee_i \text{supp}(\mathcal{E}_i(\sigma)) = \text{supp}(\textstyle\sum_i \mathcal{E}_i(\sigma)) \quad \text{(Lemma 2.18)}$$

$$= \mathrm{supp}(\sum [\![S]\!]_\epsilon(\sigma))$$

The particular case is direct since $\mathrm{supp}(\rho) = \mathrm{supp}(\sigma) \Leftrightarrow (\mathrm{supp}(\rho) \supseteq \mathrm{supp}(\sigma)) \wedge (\mathrm{supp}(\rho) \subseteq \mathrm{supp}(\sigma))$. ■

Lemma 2.23. *To be specific, for any statement defined in Fig. 3.2 and quantum predicate P, we have*

(1) $post([\![\mathbf{skip}]\!]_{ok})P = P$ $\quad post([\![\mathbf{skip}]\!]_{er})P = \mathbf{0}$

(2) $post([\![\mathbf{error}]\!]_{ok})P = \mathbf{0}$ $\quad post([\![\mathbf{error}]\!]_{er})P = P$

(3) $post([\![\bar{q} := U\bar{q}]\!]_{ok})P = UPU^\dagger$ $\quad post([\![\bar{q} := U\bar{q}]\!]_{er})P = \mathbf{0}$

(4) $post([\![q := |0\rangle]\!]_{ok})P = \mathrm{supp}(\sum_n |0\rangle_q \langle n|P|n\rangle_q \langle 0|)$ $\quad post([\![q := |0\rangle]\!]_{er})P = \mathbf{0}$

(5) $post([\![S_1; S_2]\!]_{ok})P = post([\![S_2]\!]_{ok})(post([\![S_1]\!]_{ok})P)$
$post([\![S_1; S_2]\!]_{er})P = post([\![S_2]\!]_{er})(post([\![S_1]\!]_{ok})P) \vee post([\![S_1]\!]_{er})P$

(6) $post([\![\mathbf{if}\ (\square m \cdot M[\bar{q}] = m \rightarrow S_m)\ \mathbf{fi}]\!]_\epsilon)P = \vee_m post(\mathcal{M}_m \circ [\![S_m]\!]_\epsilon)P$

(7) $post([\![\mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}]\!]_{ok})P = \vee_n post((\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_0)P$
$post([\![\mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}]\!]_{er})P = \vee_n post((\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_1 \circ [\![S']\!]_{er})P$

Proof. We prove the specific form of $post([\![S]\!]_\epsilon)P$ by induction on the structure of S.

(1) It is direct to check that the lemma holds for the **error** and **skip**.

(2) Case $S = q := |0\rangle$. For the *ok* case, let $\rho = P/\mathrm{Tr}(P)$, then we have $\mathrm{supp}(\sum [\![S]\!]_{ok}(\rho)) = \mathrm{supp}(\sum_n |0\rangle_q \langle n|\rho|n\rangle_q \langle 0|) = \mathrm{supp}(\sum_n |0\rangle_q \langle n|P|n\rangle_q \langle 0|)$ by Lemma 2.12 and 2.14, where $\mathrm{supp}(\rho) = \mathrm{supp}(P)$. For the *er* case, it is direct to have $post([\![S]\!]_{er})P = \mathbf{0}$.

(3) Case $S = \bar{q} := U\bar{q}$. Similar to the case (2).

(4) Case $S = S_1; S_2$. We have $post(\llbracket S_i \rrbracket_\epsilon)P = \mathrm{supp}(\sum \llbracket S_i \rrbracket_\epsilon(P/\mathrm{Tr}(P)))$ from the induction hypothesis for $i \in \{1, 2\}$. Let $\rho = P/\mathrm{Tr}(P)$, $\rho' = \sum \llbracket S_1 \rrbracket_{ok}(\rho)$, thus we have $R = \mathrm{supp}(\rho') = post(\llbracket S_1 \rrbracket_{ok})P$.

For the *ok* case, we have

$$\begin{aligned} \mathrm{supp}(\textstyle\sum \llbracket S \rrbracket_{ok}(\rho)) &= \mathrm{supp}(\textstyle\sum \llbracket S_2 \rrbracket_{ok}(\rho')) && \text{(Lemma 2.21)} \\ &= \mathrm{supp}(\textstyle\sum \llbracket S_2 \rrbracket_{ok}(R/\mathrm{Tr}(R))) && \text{(Lemma 2.22)} \\ &= post(\llbracket S_2 \rrbracket_{ok})R && \text{(hypothesis on } S_2) \\ &= post(\llbracket S_2 \rrbracket_{ok})(post(\llbracket S_1 \rrbracket_{ok})P) \end{aligned}$$

For the *er* case, we have

$$\begin{aligned} &\mathrm{supp}(\textstyle\sum \llbracket S \rrbracket_{er}(\rho)) = \mathrm{supp}(\textstyle\sum \llbracket S_2 \rrbracket_{er}(\rho') + \textstyle\sum \llbracket S_1 \rrbracket_{er}(\rho)) && \text{(Lemma 2.21)} \\ &= \mathrm{supp}(\textstyle\sum \llbracket S_2 \rrbracket_{er}(\rho')) \vee \mathrm{supp}(\textstyle\sum \llbracket S_1 \rrbracket_{er}(\rho)) && \text{(Lemma 2.18)} \\ &= \mathrm{supp}(\textstyle\sum \llbracket S_2 \rrbracket_{er}(R/\mathrm{Tr}(R))) \vee \mathrm{supp}(\textstyle\sum \llbracket S_1 \rrbracket_{er}(P/\mathrm{Tr}(P))) && \text{(Lemma 2.22)} \\ &= post(\llbracket S_2 \rrbracket_{er})(post(\llbracket S_1 \rrbracket_{ok})P) \vee post(\llbracket S_1 \rrbracket_{er})P && \text{(hypothesis on } S_1 \text{ and } S_2) \end{aligned}$$

(5) Case $S = \textbf{if}\ (\square m \cdot M[\bar{q}] = m \to S_m)\ \textbf{fi}$. We have $post(\llbracket S_m \rrbracket_\epsilon)P = \mathrm{supp}(\sum \llbracket S_m \rrbracket_\epsilon(P/\mathrm{Tr}(P)))$ from the induction hypothesis for every m. Let $\rho = P/\mathrm{Tr}(P)$, then we have

$$\begin{aligned} \mathrm{supp}(\textstyle\sum \llbracket S \rrbracket_\epsilon(\rho)) &= \mathrm{supp}(\textstyle\sum_m \sum (\mathcal{M}_m \circ \llbracket S_m \rrbracket_\epsilon)(\rho)) \\ &= \vee_m \mathrm{supp}(\textstyle\sum (\mathcal{M}_m \circ \llbracket S_m \rrbracket_\epsilon)(\rho)) && \text{(Lemma 2.18)} \\ &= \vee_m post(\mathcal{M}_m \circ \llbracket S_m \rrbracket_\epsilon)P && \text{(induction on the sequence)} \end{aligned}$$

(6) Case $S = \textbf{while}\ M[\bar{q}] = 1\ \textbf{do}\ S'\ \textbf{od}$. To make it compact, we define

$$\begin{aligned} \textbf{while}^n_{ok} &:= (\mathcal{M}_1 \circ \llbracket S' \rrbracket_{ok})^n \circ \mathcal{M}_0 \\ \textbf{while}^n_{er} &:= (\mathcal{M}_1 \circ \llbracket S' \rrbracket_{ok})^n \circ \mathcal{M}_1 \circ \llbracket S' \rrbracket_{er} \end{aligned}$$

Let $\rho = P/\mathrm{Tr}(P)$, by our semantics, we have

$$\mathrm{supp}(\textstyle\sum \llbracket S \rrbracket_{ok}(\rho)) = \mathrm{supp}(\textstyle\sum_n \sum \llbracket \textbf{while} \rrbracket^n_{ok}(\rho))$$

$$= \vee_n \mathrm{supp}(\textstyle\sum((\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_0)(\rho)) \quad \text{(Lemma 2.18)}$$

$$= \vee_n post((\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_0)P \quad \text{(induction on the sequence)}$$

Similarly, we have $\mathrm{supp}(\sum[\![S]\!]_{er}(\rho)) = \vee_n post((\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_1 \circ [\![S']\!]_{er})P$.

■

Chapter 3

Quantum Incorrectness Logic

Bug-catching is important for developing reliable programs. Motivated by O'Hearn's incorrectness logic (IL) [O'H19] for classical programs, this chapter proposes the quantum incorrectness logic (QIL) [YJY22] towards a logical foundation for static bug-catching in quantum programming. 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 that is robust to logical bugs (instead of noise that arises in hardware). In this chapter, 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 logic. 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 quantum teleportation (5.1), Grover's search (5.2), and a repeat-until-success program (5.3). We also automate the reasoning procedure with a prototyped static analyzer built on top of the logic rules.

Organization of Chapter 3. In Sec. 3.1, we explain the motivations for characterizing quantum errors using projections and extending the concept of reachability in classical programs into quantum settings. In Sec. 3.2, we introduce the modified language based on Ying's work [Yin12] to characterize abnormal terminations and match our incorrectness logic. We formally introduce the under-approximation relation, a quantum version of incorrectness triple, and the duality between quantum correctness and incorrectness triples in Sec. 3.3. In Sec. 3.4, a proof system for QIL triples is presented. In Sec. 3.5, we prove that our proof system is sound and complete. In Sec. 3.6, we discuss the reasons for our choice by comparing our triple with other alternatives. Examples are given in Chapter 5 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. [HM19b]. We also developed a prototyped static analyzer built on top of our proof rules to automate the reasoning.

3.1 Motivations and Ideas

We choose to adopt the convention of using projection-based quantum predicates from the quantum logic [BN36]. This approach has proven successful in reasoning about the correctness [ZYY19] of quantum programs and designing dynamic assertions [LZY⁺20]. Now we hope to extend the ideas of classical incorrectness logic into the quantum settings while still using projection-based quantum predicates. The triple [*presumption*] *code* [*result*] of classical incorrectness logic [O'H19] can be interpreted as

Every state in the result *is* reachable *from some state in the* presumption. (3.1)

There are some naturally arising problems if we want to apply this idea to the quantum settings. The first problem is how to characterize an erroneous quantum state using a projection. We can not use a projection to characterize erroneous states directly since it may lead to false positives, which must be avoided in bug-finding. The next problem is how to explain the reachability of states in quantum settings. That is, we need to figure out the meaning of reaching everything in (that is, achieving) the *result* predicate in quantum settings. In this subsection, we explain our ideas for these two problems in detail.

How to characterizing errors?

We first need to answer how to *precisely* characterize an erroneous state ρ that does not satisfy a quantum projective predicate P. Here we use a simple example to demonstrate that projection and satisfaction (Def. 2.7) cannot capture some erroneous states without introducing false positives.

Let P_c be the projection $|0\rangle\langle 0|$ for characterizing correct states, and let $\rho_e = \frac{1}{2}(|0\rangle\langle 0|_c + |1\rangle\langle 1|_e)$ is a probabilistic mixture of the correct state $|0\rangle\langle 0|_c \vDash P_c$ and the erroneous state $|1\rangle\langle 1|_e \nvDash P_c$, which means that ρ_e is also an erroneous state. 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 $\rho_e \vDash Q_e$. Assume such a Q_e exists, it means that

$$|0\rangle\langle 0|_c \subseteq \mathrm{supp}(\rho_e) \subseteq Q_e.$$

That is, a correct state $|0\rangle\langle 0|_c$ 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. Characterizing ρ_e using satisfaction and projection may falsely capture correct states. Notice that 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 solution is that we can replace the satisfaction relation with an under-approximation relation for characterizing errors. We say state ρ is *under-approximated* by projection 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.

How to interpret reachability?

In classical incorrectness logic, the triple requires the *result* predicate to be *achieved*, which means every state σ satisfying *result* can be obtained after the execution. For quantum programs, it is unreasonable to directly adopt this idea by replacing $\vDash$ with $\Dashv$ because it is commonly impossible to reach every ρ in the set $\{\rho \mid \rho \Dashv P\}$.

Here we also use a simple example to demonstrate the difficulty. Let us consider a program measuring a single qubit q with $M = \{|0\rangle\langle 0|, |1\rangle\langle 1|\}$:

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

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?

Actually, we can interpret achieving P as "P under-approximates the probabilistic *mixture* of all reachable states". This interpretation is reasonable in the sense that if the projection 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.

Now we can informally give the semantics of a QIL triple $[P]S[Q]$ that follows directly from the new interpretation of "achieving":

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

This statement can be equivalently described as $post(S)P \supseteq Q$, where *post* is the largest achievable postcondition defined formally in Sec. 3.3. We list the key ingredients of IL and QIL discussed above in Table 3.1 for comparison.

Table 3.1: 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	$\sigma \in \neg p$	$\rho \Vdash Q$ for some $Q \not\subseteq P$
p or P is achieved	$(\bigcup_{\sigma \text{ reachable}}\{\sigma\}) \supseteq p$	$\left(\sum_{\rho \text{ reachable}} \rho\right) \Vdash P$

3.2 Extended Quantum While Language

In this subsection, we introduce the extended quantum programming language with classical controls to match our quantum incorrectness logic. We extend the quantum **while** language [Yin12, Per08b, Per08a] by adding the **error** statement to capture errors and encode the **assert** statement.

Syntax

The modified syntax of the extended program language is defined in Def. 3.1. Compared with the syntax described in Def. 2.3, we add the **error** statement to capture errors and ignore general quantum operations for simplicity. 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 [LZY+20] for quantum programs, to test whether a property holds at a particular program point.

Definition 3.1 (Syntax). The extended quantum while-programs in [YJY22] is defined as follows:

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

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 similar quantum counterpart of an assertion, we restrict the predicate to be a projective measurement, following the approach of [LZY+20]. Concretely, we encode the **assert**$(\bar{q}, P)$ statement as follows,

$$\begin{aligned}\mathbf{assert}(\bar{q}, P) ::= \mathbf{if}\ (M_P[\bar{q}] = \mathbf{true} \rightarrow \mathbf{skip} \ \square\ \mathbf{false} \rightarrow \mathbf{error})\ \mathbf{fi} \\ \text{where } M_P = \{M_{\mathbf{true}}, M_{\mathbf{false}}\}, M_{\mathbf{true}} = P, M_{\mathbf{false}} = I - P.\end{aligned}$$

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 $\text{supp}(\rho) \subseteq P$, 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.

Compared with classical assertions, it is clear that quantum projective assertion **assert**$(\bar{q}, P)$ can not certainly assert the property of ρ lying in the subspace P for only one test. An abnormal termination caused by **assert**$(\bar{q}, P)$ can then be viewed as evidence of a bug. Intuitively, it means some part in ρ lies out of P. Notice that projective assertions will not bring any false positives, but they may suffer from false negatives since an erroneous state that does not satisfy P still has a probability of passing the test. Generally speaking, false positives are not allowed due to the requirement of soundness, while false negatives still turn up from time to time due to the lack of completeness.

Semantics

The semantics of the extended quantum while programs are 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'H19].

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

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.

Operational Semantics

Similarly, we model the operational semantics by labeled transitions over program configurations of the form $\langle S, \rho\rangle$. 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, and extra "*ExitCond*" is added to match our syntax in Def. 3.1. A transition is denoted by $\langle S, \rho\rangle \xrightarrow{\epsilon} \langle S', \rho'\rangle$, and the label ϵ ranges from $\{\mathit{ok}, \mathit{er}\}$ to indicate a normal/abnormal transition. The operational semantics of the extended quantum while-language is formalized in Fig. 3.1, where transition relations labeled with *er* are included.

Transitions labeled by *ok* in Fig. 3.1 are essentially the same as the standard operational semantics in Fig. 2.2a of the quantum Hoare logic [Yin12]. We explain the two transitions with the *er* label. The transition rule for **error** is intuitive. It terminates the execution, raises an *er* label, and returns the quantum state untouched. 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 label of final execution is interesting, we use the notation $\xrightarrow{\epsilon}^{*}$, where ϵ is the exiting condition, and $\epsilon = ok$ if no step is taken.

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

$$\langle q := |0\rangle, \rho\rangle \xrightarrow{ok} \langle \downarrow, \textstyle\sum_n |0\rangle_q\langle n|\rho|n\rangle_q\langle 0|\rangle$$

$$\langle \mathbf{if}\ (\Box m \cdot M[\bar{q}] = m \rightarrow S_m)\ \mathbf{fi}, \rho\rangle \xrightarrow{ok} \langle S_m, M_m\rho M_m^{\dagger}\rangle \qquad \frac{\langle S_1, \rho\rangle \xrightarrow{ok} \langle \downarrow, \rho'\rangle}{\langle S_1; S_2, \rho\rangle \xrightarrow{ok} \langle S_2, \rho'\rangle}$$

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

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

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

Figure 3.1: Operational semantics for QIL

Denotational Semantics

The "mixture of all reachable states" mentioned in Sec. 3.1 is a critical component of our incorrectness triple. The operational semantics in Fig. 3.1 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.

Similarly, 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.

$$[\![S]\!]_\epsilon : \mathcal{D}(\mathcal{H}) \rightarrow Mset(\mathcal{D}(\mathcal{H}))$$

Here $Mset(A)$ 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.

$$
\begin{array}{ll}
[\![\mathbf{error}]\!]_{ok}\rho = \mathbf{0} & [\![\mathbf{error}]\!]_{er}\rho = \rho \\
[\![\mathbf{skip}]\!]_{ok}\rho = \rho & [\![\mathbf{skip}]\!]_{er}\rho = \mathbf{0} \\
[\![q := |0\rangle]\!]_{ok}\rho = \sum |0\rangle_q\langle n|\rho|n\rangle_q\langle 0| & [\![q := |0\rangle]\!]_{er}\rho = \mathbf{0} \\
[\![\bar{q} := U\bar{q}]\!]_{ok}\rho = U\rho U^\dagger & [\![\bar{q} := U\bar{q}]\!]_{er}\rho = \mathbf{0} \\
[\![S_1; S_2]\!]_{ok} = [\![S_1]\!]_{ok} \circ [\![S_2]\!]_{ok} & [\![S_1; S_2]\!]_{er} = ([\![S_1]\!]_{ok} \circ [\![S_2]\!]_{er}) \uplus [\![S_1]\!]_{er}
\end{array}
$$

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

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

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

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

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

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

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

Figure 3.2: Denotational semantics for QIL.

Formally, we define the semantic function in Fig. 3.2, 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 $M[\bar{q}]$, and we use $\mathcal{R}$ for a function of type $\mathcal{D}(\mathcal{H}) \to Mset(\mathcal{D}(\mathcal{H}))$, $\mathcal{R}\rho$ 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, denoted by $\sum[\![S]\!]$. The sum $\sum[\![S]\!]$ converges [Sel04b, Yin12] and thus is well-defined. Noticed that our denotational semantics for QIL is different from Fig. 2.2b. In Fig. 2.2b, $[\![S]\!]$ is a direct mapping from $\mathcal{D}(\mathcal{H})$ to $\mathcal{D}(\mathcal{H})$, which is the sum of all outputs.

It is straightforward to prove that the denotational semantics is equivalent to the operational semantics, as formulated in Theorem. 3.2.

Theorem 3.2. *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.*

Proof. We proceed by induction on the structure of program S and show the equivalence between denotational semantics and operational semantics, i.e., $((\rho, \rho'), \lambda) \in [\![S]\!]_\epsilon \Leftrightarrow \langle S, \rho\rangle \xrightarrow{\epsilon}{}^{*}_{\lambda} \langle \downarrow, \rho'\rangle$ where $\rho' \neq \mathbf{0}$. Obviously, it is direct to have

$$\langle S, \rho\rangle \xrightarrow{ok}{}^{*}_{\lambda_1} \langle S', \rho'\rangle \xrightarrow{\epsilon}{}^{*}_{\lambda_2} \langle S'', \rho''\rangle \;\Leftrightarrow\; \langle S, \rho\rangle \xrightarrow{\epsilon}{}^{*}_{\lambda_1 * \lambda_2} \langle S'', \rho''\rangle$$

by combination. In this proof, we view $([\![S]\!]_\epsilon \rho)\{\mathbf{0} \rightsquigarrow 0\}$ as the multi-set without any member being $\mathbf{0}$ for convenience.

(1) It is direct to check that these two kinds of semantics are same for the basic deterministic syntax such as **error**, **skip**, $q := |0\rangle$ and $\bar{q} := U\bar{q}$.

$$[\![\mathbf{error}]\!]_{er} = \{|(\rho, \rho)|\} \;\Leftrightarrow\; \langle \mathbf{error}, \rho\rangle \xrightarrow{er} \langle \downarrow, \rho\rangle \text{ for any } \rho \in \mathcal{D}(\mathcal{H}_S)$$
$$[\![\mathbf{skip}]\!]_{ok} = \{|(\rho, \rho)|\} \;\Leftrightarrow\; \langle \mathbf{skip}, \rho\rangle \xrightarrow{ok} \langle \downarrow, \rho\rangle \text{ for any } \rho \in \mathcal{D}(\mathcal{H}_S)$$
$$[\![q := |0\rangle]\!]_{ok} = \{|(\rho, \mathcal{E}(\rho))|\} \;\Leftrightarrow\; \langle q := |0\rangle, \rho\rangle \xrightarrow{ok} \langle \downarrow, \mathcal{E}(\rho)\rangle \text{ for any } \rho \in \mathcal{D}(\mathcal{H}_S)$$
$$[\![\bar{q} := U\bar{q}]\!]_{ok} = \{|(\rho, U\rho U^\dagger)|\} \;\Leftrightarrow\; \langle \bar{q} := U\bar{q}, \rho\rangle \xrightarrow{ok} \langle \downarrow, U\rho U^\dagger\rangle \text{ for any } \rho \in \mathcal{D}(\mathcal{H}_S)$$

where $\mathcal{E}(\rho) = \sum |0\rangle_q\langle n|\rho|n\rangle_q\langle 0|$.

(2) $S = S_1; S_2$. We have $((\rho,\rho'),\lambda) \in [\![S]\!]_\epsilon \Leftrightarrow \langle S,\rho\rangle \xrightarrow{\epsilon}{}^{*}_{\lambda} \langle\downarrow,\rho'\rangle$ for S_1 and S_2 respectively by inductive hypothesis, then we need to show this hypothesis also holds for $S_1; S_2$. For the *ok* case, we have

$$\begin{aligned}
&[\![S_1;S_2]\!]_{ok} = [\![S_1]\!]_{ok} \circ [\![S_2]\!]_{ok} \\
&= \{((\rho_1,\rho_3),\lambda_1 * \lambda_2) \mid ((\rho_1,\rho_2),\lambda_1) \in [\![S_1]\!]_{ok} \text{ and } ((\rho_2,\rho_3),\lambda_2) \in [\![S_2]\!]_{ok}\} \\
&= \{((\rho_1,\rho_3),\lambda_1 * \lambda_2) \mid \langle S_1,\rho_1\rangle \xrightarrow{ok}{}^{*}_{\lambda_1} \langle\downarrow;\rho_2\rangle \text{ and } \langle S_2;\rho_2\rangle \xrightarrow{ok}{}^{*}_{\lambda_2} \langle\downarrow,\rho_3\rangle\} \\
&= \{((\rho_1,\rho_3),\lambda_1 * \lambda_2) \mid \langle S_1;S_2,\rho_1\rangle \xrightarrow{ok}{}^{*}_{\lambda_1} \langle S_2,\rho_2\rangle \xrightarrow{ok}{}^{*}_{\lambda_2} \langle\downarrow,\rho_3\rangle\} \\
&= \{((\rho_1,\rho_3),\lambda_1 * \lambda_2) \mid \langle S_1;S_2,\rho_1\rangle \xrightarrow{ok}{}^{*}_{\lambda_1*\lambda_2} \langle\downarrow,\rho_3\rangle\}
\end{aligned}$$

For the *er* case,

$$\begin{aligned}
&[\![S_1;S_2]\!]_{er} = [\![S_1]\!]_{ok} \circ [\![S_2]\!]_{er} \uplus [\![S_1]\!]_{er} \\
&= \{((\rho_1,\rho_3),\lambda_1 * \lambda_2) \mid ((\rho_1,\rho_2),\lambda_1) \in [\![S_1]\!]_{ok} \text{ and } ((\rho_2,\rho_3),\lambda_2) \in [\![S_2]\!]_{er}\} \\
&\qquad \uplus \{((\rho_1,\rho_3),\lambda_3) \mid ((\rho_1,\rho_3),\lambda_3) \in [\![S_1]\!]_{er}\} \\
&= \{((\rho,\rho'),\lambda_1 * \lambda_2 + \lambda_3) \mid ((\rho,\rho'),\lambda_1 * \lambda_2 + \lambda_3) \in [\![S_1;S_2]\!]_{er}\} \\
&= \{((\rho_1,\rho_3),\lambda_1 * \lambda_2) \mid \langle S_1;S_2,\rho_1\rangle \xrightarrow{ok}{}^{*}_{\lambda_1} \langle S_2,\rho_2\rangle \xrightarrow{er}{}^{*}_{\lambda_2} \langle\downarrow,\rho_3\rangle\} \\
&\qquad \uplus \{((\rho_1,\rho_3),\lambda_3) \mid \langle S_1,\rho_1\rangle \xrightarrow{er}{}^{*}_{\lambda_3} \langle\downarrow,\rho_3\rangle\} \\
&= \{((\rho,\rho'),\lambda_1 * \lambda_2 + \lambda_3) \mid \langle S_1;S_2,\rho\rangle \xrightarrow{er}{}^{*}_{\lambda_1*\lambda_2+\lambda_3} \langle\downarrow,\rho'\rangle\}
\end{aligned}$$

(3) $S =$ **if** $(\square m \cdot M[\bar{q}] = m \rightarrow S_m)$ **fi**. We have $((\rho_m,\rho'),\lambda_m) \in [\![S_m]\!]_\epsilon \Leftrightarrow \langle S_m,\rho_m\rangle \xrightarrow{\epsilon}{}^{*}_{\lambda_m} \langle\downarrow,\rho'\rangle$ for S_m by inductive hypothesis.

$$\begin{aligned}
&[\![S]\!]_\epsilon = \uplus_m \mathcal{M}_m \circ [\![S_m]\!]_\epsilon \\
&= \{((\rho,\rho'),\lambda_m) \mid \forall m.\ (\rho,\rho_m) \in \mathcal{M}_m \text{ and } ((\rho_m,\rho'),\lambda_m) \in [\![S_m]\!]_\epsilon\} \\
&= \{((\rho,\rho'),\lambda_m) \mid \forall m.\ \langle S,\rho\rangle \xrightarrow{ok} \langle S_m,\rho_m\rangle \xrightarrow{\epsilon}{}^{*}_{\lambda_m} \langle\downarrow,\rho'\rangle\} \\
&= \{((\rho,\rho'),\lambda_m) \mid \forall m.\ \langle S,\rho\rangle \xrightarrow{\epsilon}{}^{*}_{\lambda_m} \langle\downarrow,\rho'\rangle\}
\end{aligned}$$

(4) $S = \textbf{while } M[\bar{q}] = 1 \textbf{ do } S' \textbf{ od} = \textbf{while}$. We have $((\rho, \rho'), \lambda) \in [\![S']\!]_\epsilon \Leftrightarrow \langle S', \rho\rangle \xrightarrow{\epsilon}{}^{*}_{\lambda} \langle \downarrow, \rho'\rangle$ for S' by inductive hypothesis. To make it clear, we use $(\rho_i, \rho_i') \in \mathcal{M}_1$ and $((\rho_i', \rho_{i+1}), \lambda_i) \in [\![S']\!]_{ok}$ to denote the transition of states in the loop. we also define

$$\textbf{while}^n_{ok} := (\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_0$$
$$\textbf{while}^n_{er} := (\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_1 \circ [\![S']\!]_{er}.$$

For the *ok* case, we have $[\![S]\!]_{ok} = \uplus_{n\in\mathbb{N}}((\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_0) = \uplus_{n\in\mathbb{N}} \textbf{while}^n_{ok}$. Thus it suffices to show $((\rho, \rho'), \lambda) \in \textbf{while}^n_{ok} \Leftrightarrow \langle \textbf{while}^n_{ok}, \rho\rangle \xrightarrow{ok}{}^{*}_{\lambda} \langle \downarrow, \rho'\rangle$ for any $n \in \mathbb{N}$. For any $n \in \mathbb{N}$, we have

$$\begin{aligned}
&\textbf{while}^n_{ok} = (\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_0 = \{((\rho, \rho'), \lambda) \mid ((\rho, \rho'), \lambda) \in \textbf{while}^n_{ok}\}\\
&= \{((\rho_0, \rho'), \lambda) \mid (\rho_0, \rho_0') \in \mathcal{M}_1 \text{ and } ((\rho_0', \rho'), \lambda) \in [\![S']\!]_{ok} \circ (\mathcal{M}_1 \circ [\![S']\!]_{ok})^{n-1} \circ \mathcal{M}_0\}\\
&= \{((\rho_0, \rho'), \lambda) \mid (\rho_0, \rho_0') \in \mathcal{M}_1 \text{ and } ((\rho_0', \rho_1), \lambda_0) \in [\![S']\!]_{ok} \text{ and } ((\rho_1, \rho'), \lambda/\lambda_0) \in \textbf{while}^{n-1}_{ok}\}\\
&= \{((\rho_0, \rho'), \lambda) \mid ((\rho_0, \rho_1), \lambda_0) \in \mathcal{M}_1 \circ [\![S']\!]_{ok} \text{ and } ((\rho_1, \rho'), \lambda/\lambda_0) \in \textbf{while}^{n-1}_{ok}\}\\
&= \{((\rho_0, \rho'), \lambda) \mid ((\rho_0, \rho_n), \lambda = \textstyle\prod_{i=0}^{n-1} \lambda_i) \in (\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \text{ and } (\rho_n, \rho') \in \mathcal{M}_0\}\\
&= \{((\rho_0, \rho'), \lambda) \mid ((\rho_0, \rho'), \lambda = \textstyle\prod_{i=0}^{n-1} \lambda_i) \in \textbf{while}^n_{ok}\}\\
&= \{((\rho_0, \rho'), \lambda) \mid \langle \textbf{while}^n_{ok}, \rho_0\rangle \xrightarrow{ok} \langle [\![S']\!]_{ok} \circ \textbf{while}^{n-1}_{ok}, \rho_0'\rangle \xrightarrow{ok}{}^{*}_{\lambda_0} \langle \textbf{while}^{n-1}_{ok}, \rho_1\rangle \text{ and}\\
&\qquad ((\rho_1, \rho'), \lambda/\lambda_0) \in \textbf{while}^{n-1}_{ok}\}\\
&= \{((\rho_0, \rho'), \lambda) \mid \langle \textbf{while}^n_{ok}, \rho_0\rangle \xrightarrow{ok}{}^{*}_{\lambda_0} \langle \textbf{while}^{n-1}_{ok}, \rho_1\rangle \text{ and } ((\rho_1, \rho'), \lambda/\lambda_0) \in \textbf{while}^{n-1}_{ok}\}\\
&= \{((\rho_0, \rho'), \lambda) \mid \langle \textbf{while}^n_{ok}, \rho_0\rangle \xrightarrow{ok}{}^{*}_{\lambda=\prod_{i=0}^{n-1}\lambda_i} \langle \textbf{while}^0_{ok}, \rho_n\rangle \text{ and } (\rho_n, \rho') \in \textbf{while}^0_{ok} = \mathcal{M}_0\}\\
&= \{((\rho_0, \rho'), \lambda) \mid \langle \textbf{while}^n_{ok}, \rho_0\rangle \xrightarrow{ok}{}^{*}_{\lambda} \langle \downarrow, \rho'\rangle,\ \lambda = \textstyle\prod_{i=0}^{n-1} \lambda_i\}
\end{aligned}$$

For the *er* case, similarly, we have $[\![S]\!]_{er} = \uplus_{n\in\mathbb{N}}((\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ (\mathcal{M}_1 \circ [\![S']\!]_{er})) = \uplus_{n\in\mathbb{N}} \textbf{while}^n_{er}$. It suffices to show $((\rho, \rho'), \lambda) \in \textbf{while}^n_{er} \Leftrightarrow \langle \textbf{while}^n_{er}, \rho\rangle \xrightarrow{er}{}^{*}_{\lambda} \langle \downarrow, \rho'\rangle$ holds for any $n \in \mathbb{N}$. For any $n \in \mathbb{N}$, we have

$$\begin{aligned}
&\textbf{while}^n_{er} = (\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ (\mathcal{M}_1 \circ [\![S']\!]_{er}) = \{((\rho, \rho'), \lambda) \mid ((\rho, \rho'), \lambda) \in \textbf{while}^n_{er}\}\\
&= \{((\rho_0, \rho'), \lambda) \mid ((\rho_0, \rho_n), \lambda/\lambda_n = \textstyle\prod_{i=0}^{n-1} \lambda_i) \in (\mathcal{M}_1 \circ [\![S']\!]_{ok})^n,\ (\rho_n, \rho_n') \in \mathcal{M}_1,
\end{aligned}$$

$$
\begin{aligned}
&\quad ((\rho_n', \rho'), \lambda_n) \in [\![S']\!]_{er}\} \\
&= \{((\rho_0, \rho'), \lambda) \mid ((\rho_0, \rho'), \lambda = \textstyle\prod_{i=0}^{n} \lambda_i) \in \textbf{while}_{er}^{n}\} \\
&= \{((\rho_0, \rho'), \lambda) \mid \langle \textbf{while}_{er}^{n}, \rho_0\rangle \xrightarrow{ok}{}^{*}_{\lambda/\lambda_n = \prod_{i=0}^{n-1} \lambda_i} \langle \mathcal{M}_1 \circ [\![S']\!]_{er}, \rho_n\rangle \xrightarrow{ok} \\
&\quad \langle [\![S']\!]_{er}, \rho_n'\rangle \xrightarrow{er}{}^{*}_{\lambda_n} \langle \downarrow, \rho'\rangle\} \\
&= \{((\rho_0, \rho'), \lambda) \mid \langle \textbf{while}_{er}^{n}, \rho_0\rangle \xrightarrow{ok}{}^{*}_{\lambda/\lambda_n = \prod_{i=0}^{n-1} \lambda_i} \langle [\![S']\!]_{er}, \rho_n'\rangle \xrightarrow{er}{}^{*}_{\lambda_n} \langle \downarrow, \rho'\rangle\} \\
&= \{((\rho_0, \rho'), \lambda) \mid \langle \textbf{while}_{er}^{n}, \rho_0\rangle \xrightarrow{er}{}^{*}_{\lambda} \langle \downarrow, \rho'\rangle,\ \lambda = \textstyle\prod_{i=0}^{n} \lambda_i\}
\end{aligned}
$$

where $((\rho_i', \rho_{i+1}), \lambda_i) \in [\![S']\!]_{ok}$ holds for $i < n$ and $((\rho_n', \rho'), \lambda_n) \in [\![S']\!]_{er}$.

■

A Simple Program Example

```
// assume span{|00⟩}
$H(q_0)$; $CNOT(q_0, q_1)$;
if ( $M[q_0]$ = true → skip
              □ false → skip ) fi;
if ( $M[q_1]$ = true → skip
              □ false → skip ) fi;
// ensures span{|00⟩, |11⟩}
```

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

$$\begin{tikzcd}[column sep=small]
 & & & & |00\rangle \arrow[d, "1"] & & & \\
H, CNOT & & & & (|00\rangle + |11\rangle)/\sqrt{2} \arrow[dll, "0.5"'] \arrow[drr, "0.5"] & & & \\
M[q_0] & & |11\rangle \arrow[dl, "1"'] \arrow[dr, "0"] & & & & |00\rangle \arrow[dl, "0"'] \arrow[dr, "1"] & \\
M[q_1] & |11\rangle & & - & & - & & |00\rangle
\end{tikzcd}$$

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

Figure 3.3: A simple example that prepares and measures a Bell state.

Here we present a simple program in Fig. 3.3a to explain our semantics. After performing two unitary gates H and $CNOT$ on quantum state $|00\rangle$, we obtain the Bell state $(|00\rangle + |11\rangle)/\sqrt{2}$. 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. Then quantum 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. In particular, the measurements in Fig. 3.3 are projective. The first quantum measurement collapses the Bell state into $|11\rangle$ or $|00\rangle$ with probability $1/2$, while the second quantum measurement leaves the state unchanged since the input state satisfies the predicates $|0\rangle\langle 0|$ or $|1\rangle\langle 1|$ for sure. Let S_{Bell} be the program in Fig. 3.3a, the denotational semantics can be expressed as

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

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 already absorbed the probability[1] 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:

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

3.3 Specification Formula

In this section, we develop the quantum incorrectness triple of the form $[P]S[\epsilon{:}Q]$ based on the ideas in Sec. 3.1. 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.

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 straightforward by using satisfaction and negation of the predicate:

$$\sigma \nvDash p \;\Leftrightarrow\; \sigma \vDash \neg p. \tag{3.2}$$

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

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

cannot find an appropriate Q_e such that $\rho_e \models Q_e$ and Q_e excludes correct states, i.e., any $\rho_c \models P_c$ does not satisfy Q_e. More concretely, let $\mathbf{0} \subset P_c \subset I$, and let $\rho_e = I/\mathrm{Tr}(I) \not\models P_c$, then any Q_e that has $\rho_e \models Q_e$ would falsely capture any state $\rho_c \models P_c$, 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 (3.2). We achieve this goal by introducing the under-approximation relation.

Definition 3.3 (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:

$$\rho_e \not\models P_c \implies \exists Q_e \neq \mathbf{0}.\, (\rho_e \dashv Q_e) \wedge (\forall \rho_e' \dashv Q_e.\, \rho_e' \not\models P_c).$$

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 counterintuitive compared with those under the satisfaction, for example:

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

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 3.4 (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

$$\rho \dashv P \;\Rightarrow\; \sum [\![S]\!]_\epsilon \rho \dashv Q.$$

We use the term *strong validity* to distinguish this formulation from those alternatives discussed in Sec. 3.6. 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 postconditions 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 3.6, 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'H19]:

> "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 [LZY+20]. It is straightforward from

Def. 3.4 that for the $\textbf{assert}(\bar{q}, R)$ statement and any presumption P, we have $\vDash [P]$ $\textbf{assert}(\bar{q}, R)[er\!:\mathrm{supp}(R^{\perp}PR^{\perp})]$.[2] While the correctness triple $\vDash \{R\}\textbf{assert}(\bar{q}, R)\{R\}$ of the applied quantum Hoare logic [ZYY19] guarantees that we can safely ignore the **assert** statement in the reasoning when the assertion is satisfied, an incorrectness triple $\vDash [P]\textbf{assert}(\bar{q}, R)[er\!:\mathrm{supp}(R^{\perp}PR^{\perp})]$ with $R^{\perp}PR^{\perp} \neq \mathbf{0}$ ensures the assertion would raise an *er* with non-zero probability for some state $\rho \Vdash P$. More discussions about the validity of incorrectness triples are given in Sec 3.6 if readers are interested in why we choose such a kind of formulation.

Duality between Correctness and Incorrectness Triples

Validity of triples in QIL and the applied quantum Hoare logic [ZYY19] are two sides of the same coin when interpreted with predicate transformers.

Definition 3.5. For any quantum program S defined in Fig. 3.2 and quantum predicate P, we define the post image of program S with respect to P as follows

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

Note that the choice of ρ is not unique: any ρ that has $\mathrm{supp}(\rho) = P$ would result in an equivalent definition.[3] Based on the operator $post(\llbracket S \rrbracket_\epsilon)$, we give an equivalent formulation for the validity of incorrectness triple in Lemma 3.6.

Lemma 3.6. *For a quantum program S and a quantum predicate P, we have*

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

Proof. Give a precondition P and program S, for any valid triple $\vDash [P]S[\epsilon\!:\!Q]$, we have

$$\forall \rho \Vdash P \;\Rightarrow\; \textstyle\sum \llbracket S \rrbracket_\epsilon(\rho) \Vdash Q$$

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

[3]The predicate P in this context represents a matrix that may not be 1-dimensional, and it needs to be divided by Tr(P) for normalization.

If $P = \mathbf{0}$, let $\rho = \mathbf{0}$, then we have $\sum[\![S]\!]_\epsilon(\rho) = \mathbf{0} \dashv Q$, i.e. $Q = \mathbf{0}$. And $post([\![S]\!]_\epsilon)P = \operatorname{supp}(\sum[\![S]\!]_\epsilon(\mathbf{0})) = \mathbf{0} = Q$. For the "only if" part, let $\rho = P/\mathrm{Tr(P)}$ such that $\operatorname{supp}(\rho) = P$, then we have

$$post([\![S]\!]_\epsilon)P = \operatorname{supp}(\textstyle\sum[\![S]\!]_\epsilon(\rho)) = \operatorname{supp}(\sum[\![S]\!]_\epsilon(P/\mathrm{Tr}(P))) \supseteq Q$$

For the "if" part, by the Lemma 2.22, we have

$$\forall \rho \dashv P \;\Rightarrow\; \textstyle\sum[\![S]\!]_\epsilon(\rho) \supseteq \operatorname{supp}(\sum[\![S]\!]_\epsilon(P/\mathrm{Tr}(P))) = post([\![S]\!]_\epsilon)P \supseteq Q$$

i.e. $\vDash [P]S[\epsilon : Q]$. ∎

The operator $post([\![S]\!]_\epsilon)$ reveals the connection between the applied quantum Hoare logic [ZYY19] and QIL. It is straightforward that when S does not contain the **error** statement, $post([\![S]\!]_{ok})P \subseteq Q$ is exactly the partial correctness validity $\vDash^{\mathrm{a}}_{\mathrm{par}} \{P\}S\{Q\}$ in the applied quantum Hoare logic. The duality is then obvious, as shown below.

$$\vDash^{\mathrm{a}}_{\mathrm{par}} \{P\}S\{Q\} \quad \text{iff} \quad post([\![S]\!]_{ok})P \subseteq Q$$
$$\vDash [P]S[\epsilon : Q] \quad \text{iff} \quad post([\![S]\!]_\epsilon)P \supseteq Q$$

Specifically, we prove that the projection $post([\![S]\!]_{ok})P$ is the strongest over-approximate post for applied Hoare logic and the weakest under-approximate post for QIL, as shown in Lemma 3.7.

Lemma 3.7. *When S does not contain the* **error** *statement, for any projection P we have*

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

Proof. Recall the definition of validity of applied projective Hoare triple $\{P\}S\{Q\}$ in the sense of partial correctness,

$$\vDash^{a}_{par} \{P\}S\{Q\} \quad \text{if} \quad \forall \rho \vDash P \;\Rightarrow \textstyle\sum[\![S]\!]_{ok}(\rho) \vDash Q \qquad \text{(HL)}$$

Note that the subscript ϵ is limited to ok since the error transition is not defined in the semantics in [ZYY19]. It is direct to see such a definition is exactly dual to Def. 3.4 when $\epsilon = ok$,

$$\vDash [P]S[\epsilon : Q] \quad \text{if} \quad \forall \rho \dashv P \;\Rightarrow \sum [\![S]\!]_\epsilon(\rho) \dashv Q$$

and $post([\![S]\!]_{ok})P$ is the post predicate where these two kinds of triples reach their limits respectively. By Lemma 3.6, we have $post([\![S]\!]_\epsilon)P \supseteq Q$ for any valid $\vDash [P]S[\epsilon : Q]$. Let $Q = post([\![S]\!]_\epsilon)P$, it is direct to see that $post([\![S]\!]_\epsilon)P$ is also a valid postcondition for $\vDash [P]S[\epsilon : Q]$ by Lemma 3.6, i.e.

$$post([\![S]\!]_\epsilon)P \subseteq \;\vee\{Q \mid\vDash [P]S[\epsilon : Q]\}$$

On the other hand, we also have

$$post([\![S]\!]_\epsilon)P = \vee post([\![S]\!]_\epsilon)P \supseteq \vee\{Q \mid\vDash [P]S[\epsilon : Q]\}$$

since $P \vee P = P$, Lemma 2.13 and 2.18. Thus we have $post([\![S]\!]_\epsilon)P = \vee\{Q \mid\vDash [P]S[\epsilon : Q]\}$.

Similarly, we just need to show the following equivalence for Hoare triple.

$$\vDash^a_{par} \{P\}S\{Q\} \quad \text{iff} \quad post([\![S]\!]_{ok})P \subseteq Q$$

For the special case when $P = \mathbf{0}$, $\vDash \{P\}S\{Q\}$ holds for any Q since $Q \supseteq post([\![S]\!]_\epsilon)P = \mathbf{0}$ by Eq. (HL). For the "only if" part, let $\rho = P/\mathrm{Tr(P)}$ such that $\mathrm{supp}(\rho) = P$, by Eq. (HL), then we have

$$post([\![S]\!]_{ok})P = \mathrm{supp}(\sum [\![S]\!]_{ok}(\rho)) = \mathrm{supp}(\sum [\![S]\!]_{ok}(P/\mathrm{Tr}(P))) \subseteq Q$$

For the "if" part, by the Lemma 2.22, we have

$$\forall \rho \vDash P \;\Rightarrow\; \sum [\![S]\!]_{ok}(\rho) \subseteq \mathrm{supp}(\sum [\![S]\!]_{ok}(P/\mathrm{Tr}(P))) = post([\![S]\!]_{ok})P \subseteq Q$$

i.e. $\vDash^a_{par} \{P\}S\{Q\}$. Thus we have $post([\![S]\!]_{ok})P = \wedge\{Q \mid\vDash^a_{par} \{P\}S\{Q\}\}$ in a similar way to the incorrectness triple. ∎

Since the weakest under-approximate post is the disjunction of all quantum post predicates satisfying $[P]S[\epsilon : Q]$, Lemma 3.7 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 $\{|\psi\rangle \mid |\psi\rangle = \cos(x_i)\,|00\rangle + \sin(x_i)\,|11\rangle\}$, 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).

3.4 Proof System

In this section, we develop the proof system for QIL based on the strong validity in Def. 3.4. The proof rules of quantum incorrectness logic are shown in Fig. 3.4. Following O'Hearn [O'H19], 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.

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

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

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

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

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

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

Figure 3.4: Proof rules for QIL.

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 states. 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 as 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.

We list several other derived rules in Fig. 3.5. 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 $n \geq N$.

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

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

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

Figure 3.5: Useful derived rules.

3.5 Soundness & Completeness Theorem

Our logic is both sound and complete, as formulated by the following theorem.

Theorem 3.8. [Soundness & Completeness] *For any program S, exit condition ϵ, projections P and Q, we have,*

$$\vdash [P]S[\epsilon{:}Q] \ \Leftrightarrow\ \vDash [P]S[\epsilon{:}Q]$$

Proof. The soundness is proved by showing the validity of axioms and inference rules in Figure 3.4 with respect to Def. 3.4 by the induction on the proof structural of $\vdash [P]S[\epsilon{:}Q]$. To make it compact, the partial density operators mentioned in the proof are all in $\mathcal{D}(\mathcal{H}_S)$.

(Empty) It is clear that the trivial triple $[P]S[\epsilon{:}\mathbf{0}]$ always holds since $\rho \sqsupseteq \mathbf{0}$ for any partial density operators ρ.

(Error) Since $[\![\mathbf{error}]\!]_{ok} = \{|(\rho, \mathbf{0})|\}$ and $[\![\mathbf{error}]\!]_{er} = \{|(\rho, \rho)|\}$ by Fig. 2.2b, then for any predicate P we have

$$\forall \rho \sqsupseteq P \ \Rightarrow\ \sum [\![\mathbf{error}]\!]_{ok}(\rho) = \mathbf{0} \sqsupseteq \mathbf{0}$$

$$\forall \rho \sqsupseteq P \ \Rightarrow\ \sum [\![\mathbf{error}]\!]_{er}(\rho) = \rho \sqsupseteq P$$

Thus we have shown the validity of rule Error $\vDash [P]\mathbf{error}[ok{:}\mathbf{0}][er{:}P]$.

(Skip) Since $[\![\textbf{skip}]\!]_{ok} = \{|(\rho,\rho)|\}$ and $[\![\textbf{skip}]\!]_{er} = \{|(\rho,\mathbf{0})|\}$ by Fig. 2.2b, then for any predicate P we have

$$\forall\rho \sqsupseteq P \;\Rightarrow\; \sum[\![\textbf{skip}]\!]_{ok}(\rho) = \rho \sqsupseteq P$$

$$\forall\rho \sqsupseteq P \;\Rightarrow\; \sum[\![\textbf{skip}]\!]_{er}(\rho) = \mathbf{0} \sqsupseteq \mathbf{0}$$

Thus we have shown the validity of rule Skip $\models [P]\textbf{skip}[ok{:}P][er{:}\mathbf{0}]$.

(Unitary) First we need to show the validity of $\models [P]\bar{q} := U[\bar{q}][ok{:}UPU^\dagger]$. Let $S = \bar{q} := U[\bar{q}]$, we have $[\![S]\!]_{ok} = \{|(\rho,\rho')|\} = \{|(\rho,U\rho U^\dagger)|\}$ by Fig. 2.2b. Now it suffices to show

$$\forall\rho \sqsupseteq P \;\Rightarrow\; \sum[\![S]\!]_{ok}(\rho) = U\rho U^\dagger \sqsupseteq UPU^\dagger$$

by Def. 3.4. The proof for trivial case $P = \mathbf{0}$ is direct since $\sum[\![S]\!]_{ok}(\rho) \sqsupseteq \mathbf{0}$ always holds. It is directly provable from Lemma 2.14 by setting $\sigma = P/\mathrm{Tr}(P)$ and $\mathcal{E}(\rho) = U\rho U^\dagger$, thus we have $\models [P]\bar{q} := U[\bar{q}][ok{:}UPU^\dagger]$. On the other hand, the validity of $\models [P]\bar{q} := U[\bar{q}][er{:}\mathbf{0}]$ just derives from the rule Empty. Now we have proved the validity of rule Unitary $\models [P]\bar{q} := U[\bar{q}][ok{:}UPU^\dagger][er{:}\mathbf{0}]$.

(Init) First we need to show the validity of $\models [P]q := |0\rangle\;[ok{:}\sum|0\rangle_q\langle n|P|n\rangle_q\langle 0|]$. Let $S = q := |0\rangle$, then we have $[\![S]\!]_{ok} = \{|(\rho,\sum|0\rangle_q\langle n|\rho|n\rangle_q\langle 0|)|\}$ by Fig. 2.2b. Now it suffices to show

$$\forall\rho \sqsupseteq P \;\Rightarrow\; \sum[\![S]\!]_{ok}(\rho) \sqsupseteq \sum|0\rangle_q\langle n|P|n\rangle_q\langle 0|$$

by Def. 3.4. The proof for trivial case $P = \mathbf{0}$ is direct since $\sum[\![S]\!]_{ok}(\rho) \sqsupseteq \mathbf{0}$ always holds. The initialization can be written as $\sum[\![S]\!]_{ok}(\rho) = \rho' = \sum_n E_n\rho E_n^\dagger$ with $E_n = |0\rangle_q\langle n|$, where $\sum_n E_nE_n^\dagger = I$. Applying Lemma 2.14 with $\rho = P/\mathrm{Tr}(P)$, we have

$$\rho \sqsupseteq P \;\Rightarrow\; \rho' \sqsupseteq \mathrm{supp}(\sum E_nPE_n^\dagger)$$

Thus we have $\models [P]q := |0\rangle\;[ok{:}\sum|0\rangle_q\langle n|P|n\rangle_q\langle 0|]$. On the other hand, the validity of $\models [P]\bar{q} := |0\rangle\;[er{:}\mathbf{0}]$ just derives from the rule Empty. Now we have proved the validity of rule Init $[P]q := |0\rangle\;[ok{:}\sum|0\rangle_q\langle n|P|n\rangle_q\langle 0|][er{:}\mathbf{0}]$.

(Seq1) We have $\vDash [P]S_1[ok:R]$ and $\vDash [R]S_2[\epsilon:Q]$ by the induction hypothesis on S_1 and S_2, that is

$$\forall \rho \dashv P \implies \sum[\![S_1]\!]_{ok}(\rho) \dashv R \qquad \forall \sigma \dashv R \implies \sum[\![S_2]\!]_{\epsilon}(\sigma) \dashv Q$$

by Def. 3.4, and we need to show $\vDash [P]S_1;S_2[\epsilon:Q]$. If $P = \mathbf{0}$, we can have $\sum[\![S_1]\!]_{ok}(\mathbf{0}) = \mathbf{0} \dashv R$, i.e. $R = \mathbf{0}$. Similarly, we can also have $Q = \mathbf{0}$ since $\sum[\![S_2]\!]_{\epsilon}(\mathbf{0}) = \mathbf{0} \dashv Q$. Thus $\vDash [P]S_1;S_2[\epsilon:Q]$ is directly provable if $P = \mathbf{0}$ or $R = \mathbf{0}$ since it always has $Q = \mathbf{0}$.

Then we consider the nontrivial case $P \neq \mathbf{0}$ and $R \neq \mathbf{0}$. Let $\rho = P/\mathrm{Tr}(P)$ and $\sigma = R/\mathrm{Tr}(R)$, we have

$$\mathrm{supp}(\sum[\![S_1]\!]_{ok}(P/\mathrm{Tr}(P))) \supseteq R \qquad \mathrm{supp}(\sum[\![S_2]\!]_{\epsilon}(R/\mathrm{Tr}(R))) \supseteq Q$$

By Lemma 2.22, we have

$$\mathrm{supp}(\sum[\![S_2]\!]_{\epsilon}(\sum[\![S_1]\!]_{ok}(P/\mathrm{Tr}(P)))) \supseteq \mathrm{supp}(\sum[\![S_2]\!]_{\epsilon}(R/\mathrm{Tr}(R))) \supseteq Q$$

Case $\epsilon = ok$. Then we have $[\![S]\!]_{ok}(\rho) = \sum[\![S_2]\!]_{ok}(\sum[\![S_1]\!]_{ok}(\rho))$ by Lemma 2.21. Combine the hypothesis and Lemma 2.23 to have

$$\forall \rho \dashv P \implies \sum[\![S]\!]_{ok}(\rho) \dashv \mathrm{supp}(\sum[\![S_2]\!]_{ok}(\sum[\![S_1]\!]_{ok}(P/\mathrm{Tr}(P)))) \supseteq Q$$

thus we have $\vDash [P]S_1;S_2[ok:Q]$ by Def. 3.4.

Case $\epsilon = er$. Then we have $[\![S]\!]_{er}(\rho) = \sum[\![S_2]\!]_{er}(\sum[\![S_1]\!]_{ok}(\rho)) + \sum[\![S_1]\!]_{er}(\rho)$ by Lemma 2.21. Combine the hypothesis and Lemma 2.23 and 2.18 to have

$$\begin{aligned}\forall \rho \dashv P &\implies \sum[\![S]\!]_{er}(\rho) \dashv \mathrm{supp}(\sum[\![S_2]\!]_{er}(\sum[\![S_1]\!]_{ok}(P/\mathrm{Tr}(P))) + \sum[\![S_1]\!]_{er}(P/\mathrm{Tr}(P))) \\ &\implies \sum[\![S]\!]_{er}(\rho) \dashv \mathrm{supp}(\sum[\![S_2]\!]_{er}(\sum[\![S_1]\!]_{ok}(P/\mathrm{Tr}(P)))) \\ &\qquad \vee\, \mathrm{supp}(\sum[\![S_1]\!]_{er}(P/\mathrm{Tr}(P))) \supseteq Q\end{aligned}$$

thus we have $\vDash [P]S_1;S_2[er:Q]$ by Def. 3.4.

(Seq2) Similar to the proof of rule Seq1, we have $\vDash [P]S_1[er:Q]$ by the induction hypothesis on S_1, that is,

$$\forall \rho \dashv P \implies \sum[\![S_1]\!]_{er}(\rho) \dashv Q$$

by Def. 3.4, and we need to show $\vDash [P]S_1;S_2[er:Q]$. If $P = \mathbf{0}$, we can have $Q = \mathbf{0}$ and thus $\vDash [P]S_1;S_2[er:Q]$ always holds. Otherwise, let $\rho = P/\mathrm{Tr}(P)$, we have

$$\mathrm{supp}(\sum[\![S_1]\!]_{er}(P/\mathrm{Tr}(P))) \supseteq Q$$

Similarly, Combine the hypothesis and Lemma 2.23 to have,

$$\begin{aligned}\forall\rho \dashv P \;\Rightarrow\; \sum[\![S]\!]_{er}(\rho) &\dashv \mathrm{supp}(\sum[\![S_2]\!]_{er}(\sum[\![S_1]\!]_{ok}(P/\mathrm{Tr}(P))))\\ &\vee \mathrm{supp}(\sum[\![S_1]\!]_{er}(P/\mathrm{Tr}(P))) \supseteq Q\end{aligned}$$

thus we have $\vDash [P]S_1;S_2[er:Q]$ by Def. 3.4.

(Order) We have $\vDash [P']S[\epsilon:Q']$ by the induction hypothesis on S, and two premises $P \supseteq P'$ and $Q' \supseteq Q$, then we need to show $\vDash [P]S[\epsilon:Q]$.

$$\begin{aligned}P \supseteq P' \;&\Rightarrow\; \forall\rho \dashv P \;\Rightarrow\; \rho \dashv P'\\ &\Rightarrow\; \forall\rho \dashv P \;\Rightarrow\; \sum[\![S]\!]_{\epsilon}(\rho) \dashv Q' \quad (\vDash [P']S[\epsilon:Q'])\\ &\Rightarrow\; \forall\rho \dashv P \;\Rightarrow\; \sum[\![S]\!]_{\epsilon}(\rho) \dashv Q \quad (Q' \supseteq Q)\\ &\Rightarrow\; \vDash [P]S[\epsilon:Q]\end{aligned}$$

thus we show the validity of triple $\vDash [P]S[\epsilon:Q]$.

(Disjunction) We have two premise $\vDash [P_1]S[\epsilon:Q_1]$ and $\vDash [P_2]S[\epsilon:Q_2]$ given by the induction hypothesis on S says,

$$\begin{aligned}\forall\rho_1 \dashv P_1 \;&\Rightarrow\; \sum[\![S]\!]_{\epsilon}(\rho_1) \dashv Q_1\\ \forall\rho_2 \dashv P_2 \;&\Rightarrow\; \sum[\![S]\!]_{\epsilon}(\rho_2) \dashv Q_2\end{aligned}$$

and we need to show $\vDash [P_1 \vee P_2]S[\epsilon:Q_1 \vee Q_2]$. If $P_1 = \mathbf{0}$, then we will have $Q_1 = \mathbf{0}$ since $[\![S]\!]_{\epsilon}(\mathbf{0}) = \mathbf{0} \dashv Q_1$, thus $[P_1 \vee P_2]S[\epsilon:Q_1 \vee Q_2] = [P_2]S[\epsilon:Q_2]$ holds directly from the premise $\vDash [P_2]S[\epsilon:Q_2]$. The same goes for the case $P_2 = \mathbf{0}$. Now we discuss the general case $P_1 \neq \mathbf{0}$ and $P_2 \neq \mathbf{0}$. Let $\rho_1 = P_1/\mathrm{Tr}(P_1)$, $\rho_2 = P_2/\mathrm{Tr}(P_2)$ to have

$$\mathrm{supp}(\sum[\![S]\!]_{\epsilon}(P_1/\mathrm{Tr}(P_1))) \supseteq Q_1 \qquad \mathrm{supp}(\sum[\![S]\!]_{\epsilon}(P_2/\mathrm{Tr}(P_2))) \supseteq Q_2$$

Thus we can apply these two premises to have

$$\begin{aligned}
&\forall\rho \dashv P_1 \vee P_2 = \mathrm{supp}((P_1+P_2)/\mathrm{Tr}(P_1+P_2)) \quad \text{(Lemma 2.18)}\\
&\Rightarrow \mathrm{supp}(\textstyle\sum[\![S]\!]_\epsilon(\rho)) \supseteq \mathrm{supp}(\sum[\![S]\!]_\epsilon((P_1+P_2)/\mathrm{Tr}(P_1+P_2))) \quad \text{(Lemma 2.22)}\\
&\Rightarrow \mathrm{supp}(\textstyle\sum[\![S]\!]_\epsilon(\rho)) \supseteq \mathrm{supp}(\sum[\![S]\!]_\epsilon(P_1/\mathrm{Tr}(P_1+P_2))+\\
&\qquad\qquad\qquad \textstyle\sum[\![S]\!]_\epsilon(P_2/\mathrm{Tr}(P_1+P_2))) \quad \text{(Lemma 2.21)}\\
&\Rightarrow \mathrm{supp}(\textstyle\sum[\![S]\!]_\epsilon(\rho)) \supseteq \mathrm{supp}(\sum[\![S]\!]_\epsilon(P_1/\mathrm{Tr}(P_1+P_2)))\vee\\
&\qquad\qquad\qquad \mathrm{supp}(\textstyle\sum[\![S]\!]_\epsilon(P_2/\mathrm{Tr}(P_1+P_2))) \quad \text{(Lemma 2.18)}\\
&\Rightarrow \mathrm{supp}(\textstyle\sum[\![S]\!]_\epsilon(\rho)) \supseteq \mathrm{supp}(\sum[\![S]\!]_\epsilon(P_1/\mathrm{Tr}(P_1)))\vee\\
&\qquad\qquad\qquad \mathrm{supp}(\textstyle\sum[\![S]\!]_\epsilon(P_2/\mathrm{Tr}(P_2))) \quad \text{(Lemma 2.22)}\\
&\Rightarrow \mathrm{supp}(\textstyle\sum[\![S]\!]_\epsilon(\rho)) \supseteq Q_1 \vee Q_2\\
&\Rightarrow \vDash [P_1 \vee P_2]S[\epsilon : Q_1 \vee Q_2]
\end{aligned}$$

(If) $S = \textbf{if } (\square m \cdot M[\bar{q}] = m \longrightarrow S_m) \textbf{ fi}$. By Def. 3.4, the triple $\vDash [M_m P M_m^\dagger] S_m [\epsilon : Q]$ given by the induction hypothesis on S_m says,

$$\exists m. \forall \rho_m \dashv \mathrm{supp}(M_m P M_m^\dagger) \implies \textstyle\sum[\![S_m]\!]_\epsilon(\rho_m) \dashv Q$$

and we need to show $\vDash [P]S[\epsilon : Q]$. By Lemma 2.23, we also have $\vDash [P]\mathcal{M}_m[ok : M_m P M_m^\dagger]$. Then we have

$$\exists m. \forall \rho \dashv P \implies M_m \rho M_m^\dagger \dashv \mathrm{supp}(M_m P M_m^\dagger) \implies \textstyle\sum[\![S_m]\!]_\epsilon(M_m \rho M_m^\dagger) \dashv Q$$

If $P = \mathbf{0}$, we can also have $Q = \mathbf{0}$ and thus $\vDash [P]S[\epsilon : Q]$ always holds. Otherwise, let $\rho = P/\mathrm{Tr}(P)$, then we have $Q_m = \mathrm{supp}(\sum[\![S_m]\!]_\epsilon(M_m P M_m^\dagger/\mathrm{Tr}(P))) \supseteq Q$. By Lemma 2.23 and Lemma 2.18, we have

$$\forall \rho \dashv P \implies \textstyle\sum[\![S]\!]_\epsilon(\rho) \dashv \mathrm{supp}(\sum_m \sum[\![S_m]\!]_\epsilon(M_m P M_m^\dagger/\mathrm{Tr}(P))) = \vee_m Q_m \supseteq Q_m \supseteq Q$$

Thus we show the validity of triple $\vDash [P]\textbf{if } (\square m \cdot M[\bar{q}] = m \longrightarrow S_m) \textbf{ fi}[\epsilon : Q]$.

(While1) By Fig. 2.2b, we have $[\![\textbf{while}]\!]_{ok} = \biguplus_{n\in\mathbb{N}} (\mathcal{M}_1 \circ [\![S]\!]_{ok})^n \circ \mathcal{M}_0$. To make it compact, we denote $\rho_k = \sum(\mathcal{M}_1 \circ [\![S]\!]_{ok})^k(\rho_0)$ and $P_i^m = \mathrm{supp}(M_m P_i M_m^\dagger)$. We have the premise $\vDash [P_n^1]S[ok{:}P_{n+1}]$ holds for all integer n ($0 \le n$) by the induction hypothesis on S, which says

$$\forall \rho \dashv P_n^1 \;\Rightarrow\; \sum [\![S]\!]_{ok}(\rho) \dashv P_{n+1}$$

Just like the proof of Seq1, by the induction hypothesis, we have $P_{n+1} = \mathbf{0}$ if any $P_n^1 = \mathbf{0}$ ($0 \le n \le N$), thus $\vDash [P_0]\mathbf{while}[ok{:}\mathrm{supp}(M_0 P_N M_0^\dagger)]$ always holds for any integer N. For the nontrivial case, let $\rho = P_n^1/\mathrm{Tr}(P_n^1)$, we have $\mathrm{supp}(\sum [\![S]\!]_{ok}(P_n^1/\mathrm{Tr}(P_n^1))) \supseteq P_{n+1}$. By Lemma 2.23, we have $\vDash [P]\mathcal{M}_0[ok{:}M_0 P M_0^\dagger]$ and $\vDash [P]\mathcal{M}_1[ok{:}M_1 P M_1^\dagger]$. Then for any $N \in \mathbb{N}$, we have

$$\begin{aligned}
\forall \rho_0 \dashv P_0 \;&\Rightarrow\; M_1\rho_0 M_1^\dagger \dashv P_0^1 \quad \text{(Lemma 2.23)}\\
&\Rightarrow\; \rho_1 = \sum [\![S]\!]_{ok}(M_1\rho_0 M_1^\dagger) \dashv \mathrm{supp}(\sum [\![S]\!]_{ok}(P_0^1/\mathrm{Tr}(P_0^1))) \supseteq P_1 \quad \text{(Lemma 2.23)}\\
&\Rightarrow\; \rho_N \dashv P_N \quad \text{(apply last two steps N times)}\\
&\Rightarrow\; M_0\rho_N M_0^\dagger \dashv P_N^0 \quad \text{(Lemma 2.23)}\\
&\Rightarrow\; \mathrm{supp}(\sum ((\mathcal{M}_1 \circ [\![S]\!]_{ok})^N \circ \mathcal{M}_0)(\rho_0)) \supseteq P_N^0\\
&\Rightarrow\; \mathrm{supp}(\sum [\![\mathbf{while}]\!]_{ok}(\rho_0)) = \mathrm{supp}(\textstyle\sum_{n=0}^{+\infty}((\mathcal{M}_1 \circ [\![S]\!]_{ok})^n \circ \mathcal{M}_0)(\rho_0))\\
&\qquad \supseteq \vee_{n=0}^{N}\mathrm{supp}(\sum ((\mathcal{M}_1 \circ [\![S]\!]_{ok})^n \circ \mathcal{M}_0)(\rho_0)) \supseteq P_N^0\\
&\Rightarrow\; \vDash [P_0]\mathbf{while}[ok{:}P_N^0]
\end{aligned}$$

(While2) By Fig. 2.2b, we have $[\![\mathbf{while}]\!]_{er} = \uplus_{n\in\mathbb{N}}((\mathcal{M}_1 \circ [\![S]\!]_{ok})^n \circ (\mathcal{M}_1 \circ [\![S]\!]_{er}))$. The triple $\vDash [\mathrm{supp}(M_1 P_N M_1^\dagger)]S[er{:}Q]$ given by the induction hypothesis on S says,

$$\forall \rho \dashv P_N^1 \;\Rightarrow\; \sum [\![S]\!]_{er}(\rho) \dashv Q$$

and the triple $\vDash [\mathrm{supp}(M_1 P_n M_1^\dagger)]S[ok{:}P_{n+1}]$ for all integer n ($0 \le n \le N$) says,

$$\forall \rho \dashv P_n^1 \;\Rightarrow\; \sum [\![S]\!]_{ok}(\rho) \dashv P_{n+1}$$

Similarly, by the induction hypothesis, we have $Q = \mathbf{0}$ if any $P_n^1 = \mathbf{0}$ ($0 \le n \le N$), thus $\vDash [P_0]\mathbf{while}[er{:}Q]$ always holds. Otherwise, let $\rho = P_n^1/\mathrm{Tr}(P_n^1)$, we have $\mathrm{supp}(\sum [\![S]\!]_{ok}($

$P_n^1/\mathrm{Tr}(P_n^1))) \supseteq P_{n+1}$ and $\mathrm{supp}(\sum [\![S]\!]_{er}(P_N^1/\ \mathrm{Tr}(P_N^1))) \supseteq Q$ by the premises. Then for any $N \in \mathbb{N}$, we have

$$
\begin{aligned}
\forall \rho_0 \dashv P_0 \;&\Rightarrow\; M_1\rho_0 M_1^\dagger \dashv P_0^1 \quad \text{(Lemma 2.23)}\\
&\Rightarrow\; \rho_N \dashv P_N \quad \text{(same as the proof of rule WHILE1)}\\
&\Rightarrow\; M_1\rho_N M_1^\dagger \dashv P_N^1 \quad \text{(Lemma 2.23)}\\
&\Rightarrow\; \textstyle\sum [\![S]\!]_{er}(M_1\rho_N M_1^\dagger) \dashv \mathrm{supp}(\sum [\![S]\!]_{er}(P_N^1/\mathrm{Tr}(P_N^1))) \supseteq Q\\
&\qquad \text{(Lemma 2.23 and } \vDash [P_N^1]S[er{:}Q])\\
&\Rightarrow\; \mathrm{supp}(\textstyle\sum((\mathcal{M}_1 \circ [\![S]\!]_{ok})^N \circ \mathcal{M}_0 \circ [\![S]\!]_{er})(\rho_0)) \supseteq Q\\
&\Rightarrow\; \mathrm{supp}(\textstyle\sum [\![\mathbf{while}]\!]_{er}(\rho)) = \mathrm{supp}(\sum_{n=0}^{+\infty}((\mathcal{M}_1 \circ [\![S]\!]_{ok})^n \circ \mathcal{M}_0 \circ [\![S]\!]_{er})(\rho_0))\\
&\qquad \supseteq \vee_{n=0}^{N}\mathrm{supp}(\textstyle\sum((\mathcal{M}_1 \circ [\![S]\!]_{ok})^n \circ \mathcal{M}_0 \circ [\![S]\!]_{er})(\rho_0)) \supseteq Q\\
&\Rightarrow\; \vDash [P_0]\mathbf{while}[er{:}Q]
\end{aligned}
$$

The completeness can be directly derived from the proof of Theorem 3.9 since the WHILE rules in Fig. 3.4 consist of infinite rules for all $n \in \mathbb{N}$ and include their bounded versions. These bounded WHILE rules form a minimal set of rules which are sufficient for reasoning about loop structure. ■

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

Bounded While1

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

Bounded While2

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

Proof. Similarly, the soundness can be directly derived from the proof of Theorem 3.8 since the While rules in Fig. 3.4 consist of infinite rules for all $n \in \mathbb{N}$ and include their bounded versions.

The completeness of the proof system means any valid incorrectness triple is derivable from the proof rules. Assume $\vDash [P]S[\epsilon\!:\!Q]$, i.e.,

$$\forall \rho.\ \rho \dashv P \implies \sum [\![S]\!]_\epsilon(\rho) \dashv Q$$

we prove $\vdash [P]S[\epsilon\!:\!Q]$ by induction on S. To make it concise, density operators mentioned in the proof are all in $\mathcal{D}(\mathcal{H}_S)$.

(1) $S = \textbf{error}$. We prove by case studying ϵ.

a) Case $\epsilon = ok$. By Fig. 2.2b, $[\![\textbf{error}]\!]_{ok} = \{|(\rho, \mathbf{0})|\}$. The triple $\vDash [P]\textbf{error}[ok\!:\!Q]$ implies $\forall \rho \dashv P \implies \mathbf{0} \dashv Q$, which implies $Q = \mathbf{0}$. It suffices to prove $\vdash [P]\textbf{error}[ok\!:\!\mathbf{0}]$ exactly by the rule Error.

b) Case $\epsilon = er$. By Fig. 2.2b, $[\![\textbf{error}]\!]_{er} = \{|(\rho, \rho)|\}$. We have $\forall \rho \dashv P \implies \rho \dashv Q$, which implies $P \supseteq Q$. The triple is then derivable from rule Error and

Order:
$$\frac{\vdash [P]\mathbf{error}[er\!:\!P] \quad P \supseteq Q}{\vdash [P]\mathbf{error}[er\!:\!Q]}$$

(2) $S = \mathbf{skip}$.

a) Case $\epsilon = ok$. We have $[\![\mathbf{skip}]\!]_{ok} = \{|(\rho, \rho)|\}$ by Fig. 2.2b. The triple $\vDash [P]\mathbf{skip}[ok\!:\!Q]$ implies $\forall \rho \Rdash P \implies \rho \Rdash Q$, which implies $P \supseteq Q$. The triple is derivable from rule Skip and Order:
$$\frac{\vdash [P]\mathbf{skip}[ok\!:\!P] \quad P \supseteq Q}{\vdash [P]\mathbf{skip}[ok\!:\!Q]}$$

b) Case $\epsilon = er$. We have $[\![\mathbf{skip}]\!]_{er} = \{|(\rho, \mathbf{0})|\}$ by Fig. 2.2b. The triple $\vDash [P]\mathbf{skip}[er\!:\!Q]$ implies $\forall \rho \Rdash P \implies \mathbf{0} \Rdash Q$, which implies $Q = \mathbf{0}$. It suffices to prove $\vdash [P]\mathbf{skip}[er\!:\!\mathbf{0}]$ exactly by the rule Skip.

(3) $S = \bar{q} := U[\bar{q}]$.

a) Case $\epsilon = ok$. Using rule Unitary and rule Order, the triple is derivable assuming $UPU^\dagger \supseteq Q$:
$$\frac{\vdash [P]\bar{q} := U[\bar{q}][ok\!:\!UPU^\dagger] \quad UPU^\dagger \supseteq Q}{\vdash [P]S[ok\!:\!Q]}$$

It suffices to prove $UPU^\dagger \supseteq Q$ from $\vDash [P]S[\epsilon\!:\!Q]$. Since $[\![\bar{q} := U[\bar{q}]]\!]_{ok} = \{|(\rho, U\rho U^\dagger)|\}$ by Fig. 2.2b, we have $\forall \rho \Rdash P \implies U\rho U^\dagger \Rdash Q$. Let $\rho = P/\mathrm{Tr}(P)$, we have $\mathrm{supp}(U\rho U^\dagger) = UPU^\dagger \supseteq Q$.

b) Case $\epsilon = er$. Since $[\![\bar{q} := U[\bar{q}]]\!]_{er} = \{|(\rho, \mathbf{0})|\}$ by Fig. 2.2b, we have $\forall \rho \Rdash P \implies \mathbf{0} \Rdash Q$, which implies $Q = \mathbf{0}$. Thus the triple is directly derivable from the rule Unitary $\vdash [P]\bar{q} := U[\bar{q}][er\!:\!\mathbf{0}]$.

(4) $S = \bar{q} := |0\rangle$.

a) Case $\epsilon = ok$. By Fig. 2.2b, $[\![\bar{q} := |0\rangle]\!]_{ok} = \{|(\rho, \sum |0\rangle_q\langle n|\rho|n\rangle_q\langle 0|)|\}$. Let $\rho = P/\mathrm{Tr}(P)$, then by $\vDash [P]S[ok\!:\!Q]$ we have $\mathrm{supp}(\sum |0\rangle_q\langle n|P|n\rangle_q\langle 0|) \supseteq Q$.

The triple is derivable using rule Init and rule Oder:

$$\frac{[P] \vdash \bar{q} := |0\rangle\ [ok{:}\mathrm{supp}(\sum |0\rangle_q\langle n|P|n\rangle_q\langle 0|)] \qquad \mathrm{supp}(\sum |0\rangle_q\langle n|P|n\rangle_q\langle 0|) \supseteq Q}{\vdash [P]S[ok{:}Q]}$$

b) Case $\epsilon = er$. By Fig. 2.2b, $[\![\bar{q} := |0\rangle]\!]_{er} = \{|(\rho, \mathbf{0})|\}$. By $\vDash [P]S[ok{:}Q]$ we have $\forall \rho \dashv P \implies \mathbf{0} \dashv Q$, which implies $Q = \mathbf{0}$. Thus the triple is directly derivable from the rule Init $\vdash [P]\bar{q} := |0\rangle\ [er{:}\mathbf{0}]$.

(5) $S = S_1; S_2$. By Lemma 3.4, $\vDash [P]S_1; S_2[\epsilon{:}Q]$ means that $\forall \rho \dashv P \implies \sum[\![S_1; S_2]\!]_\epsilon(\rho) \dashv Q$. For the trivial case $P = \mathbf{0}$, we can derive $Q = \mathbf{0}$ since $\sum[\![S_1; S_2]\!]_\epsilon(\mathbf{0}) = \mathbf{0} \dashv Q$. Thus $\vdash [P]S_1; S_2[\epsilon{:}Q]$ is directly derivable by the rule Empty.

a) Case $\epsilon = ok$, i.e. $[\![S]\!]_{ok} = [\![S_1]\!]_{ok} \circ [\![S_2]\!]_{ok}$. By Lemma 2.23 applied on program S_1, S_2, we have

$$\vDash [P]S_1[ok{:}R] \qquad \vDash [R]S_2[ok{:}T]$$

where $R = \mathrm{supp}(\sum[\![S_1]\!]_{ok}(P/\mathrm{Tr}(P)))$ and $T = \mathrm{supp}(\sum[\![S_2]\!]_{ok}(R/\mathrm{Tr}(R)))$. By induction hypothesis on program S_1, S_2, we have $\vdash [P]S_1[ok{:}R]$ and $\vdash [R]S_2[ok{:}T]$. Then applying the rule Seq1 to have

$$\frac{\vdash [P]S_1[ok{:}R] \qquad \vdash [R]S_2[ok{:}T]}{\vdash [P]S_1; S_2[ok{:}T]}$$

Combined with the rule Order, the triple is derivable assuming that $T \supseteq Q$.

$$\frac{\vdash [P]S_1; S_2[ok{:}T] \qquad T \supseteq Q}{\vdash [P]S_1; S_2[ok{:}Q]}$$

Then it suffices to prove $T \supseteq Q$. We have

$$\forall \rho \dashv P \implies \sum[\![S_2]\!]_{ok}(\sum[\![S_1]\!]_{ok}(\rho)) \dashv Q$$

from $\vDash [P]S_1;S_2[ok\!:\!Q]$. Let $\rho = P/\mathrm{Tr}(P)$, by Lemma 2.22, we have

$$\mathrm{supp}(\textstyle\sum[\![S_2]\!]_{ok}(\sum[\![S_1]\!]_{ok}(P/\mathrm{Tr}(P)))) = T \supseteq Q$$

.

b) Case $\epsilon = er$, i.e. $[\![S]\!]_{er} = [\![S_1]\!]_{ok} \circ [\![S_2]\!]_{er} \uplus [\![S_1]\!]_{er}$. Similarly, by Lemma 2.23 and induction hypothesis on program S_1, S_2, we have

$$\vdash [P]S_1[ok\!:\!R] \qquad \vdash [R]S_2[er\!:\!R'] \qquad \vdash [P]S_1[er\!:\!T]$$

where $R = \mathrm{supp}(\sum[\![S_1]\!]_{ok}(P/\mathrm{Tr}(P)))$, $R' = \mathrm{supp}(\sum[\![S_1]\!]_{er}(P/\mathrm{Tr}(P)))$, and $T = \mathrm{supp}(\sum[\![S_2]\!]_{er}\,(R/\mathrm{Tr}(R)))$. Then apply the rule Seq1 and Seq2 to have

$$\frac{\vdash [P]S_1[ok\!:\!R] \quad \vdash [R_i]S_2[er\!:\!T]}{\vdash [P]S_1;S_2[er\!:\!T]} \qquad \frac{\vdash [P]S_1[er\!:\!R']}{\vdash [P]S_1;S_2[er\!:\!R']}$$

Then apply the rule Disjunction to have

$$\frac{\vdash [P]S_1;S_2[er\!:\!T] \quad \vdash [P]S_1;S_2[er\!:\!R']}{\vdash [P]S_1;S_2[er\!:\!T \vee R']}$$

Combined with the rule Order, the triple is derivable assuming that $T \vee R' \supseteq Q$.

$$\frac{\vdash [P]S_1;S_2[er\!:\!T \vee R'] \quad T \vee R' \supseteq Q}{\vdash [P]S_1;S_2[er\!:\!Q]}$$

Then it suffices to prove $T \vee R' \supseteq Q$. We have

$$\forall \rho \dashv P \;\Rightarrow\; \textstyle\sum[\![S_2]\!]_{er}(\sum[\![S_1]\!]_{ok}(\rho)) + \sum[\![S_1]\!]_{er}(\rho) \dashv Q$$

from $\vDash [P]S_1;S_2[er\!:\!Q]$. Let $\rho = P/\mathrm{Tr}(P)$, by Lemma 2.18 and Lemma 2.22, then we have

$$\mathrm{supp}(\textstyle\sum[\![S_2]\!]_{er}(\sum[\![S_1]\!]_{ok}(P/\mathrm{Tr}(P))) + \sum[\![S_1]\!]_{er}(P/\mathrm{Tr}(P))) = T \vee R' \supseteq Q$$

(6) $S = \textbf{if}\ (\square m \cdot M[\bar{q}] = m \rightarrow S_m)\ \textbf{fi}$. By Lemma 3.4, $\vDash [P]S[\epsilon\!:\!Q]$ means $\forall \rho \dashv P \;\Rightarrow\; \sum[\![S]\!]_\epsilon(\rho) \dashv Q$. Here we can reason two cases together and have

$\forall\rho \Dashv P \Rightarrow \sum_m [\![S_m]\!]_\epsilon(M_m\rho M_m^\dagger) \Dashv Q$. Similarly, for the trivial case $P = \mathbf{0}$, $\vdash [P]S[\epsilon : Q]$ is directly derivable by the rule Empty since we have $Q = \mathbf{0}$ from $\sum[\![S]\!]_\epsilon(\mathbf{0}) = \mathbf{0} \Dashv Q$. By Lemma 2.23 and induction hypothesis on S_m, we have $\vdash [M_m P M_m^\dagger]S_m[\epsilon : Q_m]$, where $Q_m = \mathrm{supp}(\sum[\![S_m]\!]_\epsilon(M_m P M_m^\dagger/\mathrm{Tr}(M_m P M_m^\dagger)))$. Then for each m, apply the rule If to have

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

Then apply the rule Disjunction to have,

$$\frac{\vdash [P]\,\mathbf{if}\ (\square m \cdot M[\bar{q}] = m \rightarrow S_m)\ \mathbf{fi}[\epsilon : Q_m]}{\vdash [P]\,\mathbf{if}\ (\square m \cdot M[\bar{q}] = m \rightarrow S_m)\ \mathbf{fi}[\epsilon : \vee_m Q_m]}$$

Combined with the rule Order, the triple is derivable assuming that $\vee_m Q_m \supseteq Q$.

$$\frac{\vdash [P]\,\mathbf{if}\ (\square m \cdot M[\bar{q}] = m \rightarrow S_m)\ \mathbf{fi}[\epsilon : \vee_m Q_m] \qquad \vee_m Q_m \supseteq Q}{\vdash [P]\,\mathbf{if}\ (\square m \cdot M[\bar{q}] = m \rightarrow S_m)\ \mathbf{fi}[\epsilon : Q]}$$

Then it suffices to prove $\vee_m Q_m \supseteq Q$. we have $\forall\rho \Dashv P \Rightarrow \sum_m(\sum[\![S_m]\!]_\epsilon(M_m\rho M_m^\dagger)) \Dashv Q$ from $\models [P]S[\epsilon : Q]$. Let $\rho = P/\mathrm{Tr}(P)$, by Lemma 2.18, then we have $\mathrm{supp}(\sum_m(\sum[\![S_m]\!]_\epsilon(M_m P M_m^\dagger/\ \mathrm{Tr}(M_m P M_m^\dagger)))) = \vee_m Q_m \supseteq Q$

(7) $S = \mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ S'\ \mathbf{od}$. Again, for the trivial case $P = \mathbf{0}$, $\vdash [P]S[\epsilon : Q]$ is directly derivable by the rule Empty since we have $Q = \mathbf{0}$.

a) Case $\epsilon = ok$. By $\models [P]S[ok : Q]$ and Fig. 2.2b, we have

$$\forall\rho \Dashv P \Rightarrow \sum_n(\sum((\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_0)(\rho)) \Dashv Q$$

By Lemma 2.23 and induction hypothesis on S', we have

$$\vdash [M_1 P_n M_1^\dagger]S'[ok : P_{n+1}]$$

where $P_n = \mathrm{supp}(\sum(\mathcal{M}_1 \circ [\![S']\!]_{ok})^n(P/\mathrm{Tr}(P)))$ and $P_0 = P$. Now we first prove that it is sufficient to unroll the loop structure finite times to derive

$\vdash [P]S[\epsilon : Q]$. To be more specific, the upper bound of N is the dimension of Hilbert space $\mathcal{H}$ described by program S. Notice that in our paper we only reason about quantum program in finite Hilbert space, i.e the dimension of any projection Q_n is also finite. If we have $P_{N+1} \subseteq \vee_{n=0}^{N} P_n$, then

$$\begin{aligned} P_{N+2} &= post(\llbracket S' \rrbracket_{ok})(\mathrm{supp}(M_1 P_{N+1} M_1^\dagger)) \\ &\subseteq post(\llbracket S' \rrbracket_{ok})(\vee_{n=0}^{N} \mathrm{supp}(M_1 P_n M_1^\dagger)) \\ &\subseteq \vee_{n=0}^{N} post(\llbracket S' \rrbracket_{ok})(\mathrm{supp}(M_1 P_n M_1^\dagger)) \\ &= \vee_{n=0}^{N} P_{n+1} \subseteq \vee_{n=0}^{N+1} P_n = (\vee_{n=0}^{N} P_n) \vee P_{N+1} = \vee_{n=0}^{N} P_n \end{aligned}$$

Thus we have $\vee_{n=0}^{N} P_n = \vee_{n=0}^{+\infty} P_n$ if $P_{N+1} \subseteq \vee_{n=0}^{N} P_n$ for any $N \in \mathbb{N}$. Notice that the dimension of any P_n is not bigger than $\dim(\mathcal{H})$, i.e. $\dim(\vee_{n=0}^{+\infty} P_n) \leq \dim(\mathcal{H})$, and $\dim(\vee_{n=0}^{k} P_n) \leq \dim(\vee_{n=0}^{k+1} P_n)$ for any $k \in \mathbb{N}$, then it takes at most $\dim(\mathcal{H})$ unrolling to have $P_{N+1} \subseteq \vee_{n=0}^{N} P_n$. Thus there exists $N \leq \dim(\mathcal{H})$ such that $\vee_{n=0}^{N} P_n = \vee_{n=0}^{+\infty} P_n$.

Now we apply the rule Bounded While1 to have

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

Then apply the rule Disjunction by induction to have

$$\frac{\vdash [P] \textbf{while } M[\bar{q}] = 1 \textbf{ do } S' \textbf{ od} [ok : \mathrm{supp}(M_0 P_n M_0^\dagger)]}{\vdash [P] \textbf{while } M[\bar{q}] = 1 \textbf{ do } S' \textbf{ od} [ok : \vee_{n=0}^{N} \mathrm{supp}(M_0 P_n M_0^\dagger)]}$$

Finally the triple is derivable by the rule Order assuming that $\vee_{n=0}^{N} \mathrm{supp}(M_0 P_n M_0^\dagger) \supseteq Q$.

$$\frac{\vdash [P] \textbf{while } M[\bar{q}] = 1 \textbf{ do } S' \textbf{ od} [ok : \vee_{n=0}^{N} \mathrm{supp}(M_0 P_n M_0^\dagger)] \qquad \vee_{n=0}^{N} \mathrm{supp}(M_0 P_n M_0^\dagger) \supseteq Q}{\vdash [P] \textbf{while } M[\bar{q}] = 1 \textbf{ do } S' \textbf{ od} [ok : Q]}$$

It suffices to prove $\vee_{n=0}^{N}\mathrm{supp}(M_0P_nM_0^\dagger) \supseteq Q$ by $\vDash [P]S[ok{:}Q]$. Let $\rho = P/\mathrm{Tr}(P)$, by Lemma 2.18, we have

$$\forall \rho \dashv P \implies \textstyle\sum_{n=0}^{+\infty}(\sum((\mathcal{M}_1\circ[\![S']\!]_{ok})^n\circ\mathcal{M}_0)(P/\mathrm{Tr(P)})) = \vee_{n=0}^{N}\mathrm{supp}(M_0P_nM_0^\dagger) \supseteq Q$$

b) Case $\epsilon = er$. Similarly, by $\vDash [P]S[er{:}Q]$ and Fig. 2.2b, we have

$$\forall \rho \dashv P \implies \textstyle\sum_n(\sum((\mathcal{M}_1\circ[\![S']\!]_{ok})^n\circ\mathcal{M}_1\circ[\![S']\!]_{er})(\rho)) \dashv Q$$

where the support of the trivial state $\mathbf{0}$ has no effect. Just like the case $\epsilon = ok$, by Lemma 2.23 and induction hypothesis on program S', we have

$$\vdash [M_1P_nM_1^\dagger]S'[ok{:}P_{n+1}] \qquad \vdash [M_1P_nM_1^\dagger]S'[er{:}P'_n]$$

where $P_n = \mathrm{supp}(\sum(\mathcal{M}_1\circ[\![S']\!]_{ok})^n(P/\mathrm{Tr}(P)))$, $P_0 = P$, and $P'_n = \mathrm{supp}(\sum[\![S']\!]_{er}(M_1P_nM_1^\dagger\,/\mathrm{Tr}\,(M_1P_nM_1^\dagger)))$. By Lemma 2.21 and Lemma 2.22, we have

$$\mathrm{supp}(\textstyle\sum((\mathcal{M}_1\circ[\![S']\!]_{ok})^n\circ\mathcal{M}_0\circ[\![S']\!]_{er})(P/\mathrm{Tr}(P))) = P'_n$$

Again, the proof for the upper bound of N is similar to case $\epsilon = ok$. If we have $P_{N+1}\subseteq \vee_{n=0}^{N}P_n$, then

$$\begin{aligned}P'_{N+1} &= post([\![S']\!]_{er})(\mathrm{supp}(M_1P_{N+1}M_1^\dagger)) \subseteq post([\![S']\!]_{er})(\vee_{n=0}^{N}\mathrm{supp}(M_1P_nM_1^\dagger))\\ &\subseteq \vee_{n=0}^{N}post([\![S']\!]_{er})(\mathrm{supp}(M_1P_nM_1^\dagger)) = \vee_{n=0}^{N}P'_n\end{aligned}$$

Thus we also have $\vee_{n=0}^{N}P'_n = \vee_{n=0}^{+\infty}P'_n$ if $P_{N+1}\subseteq\vee_{n=0}^{N}P_n$ for any $N\in\mathbb{N}$. Notice that $\dim(\vee_{n=0}^{+\infty}P'_n)\le\dim(\mathcal{H})$ and $\dim(\vee_{n=0}^{k}P'_n)\le\dim(\vee_{n=0}^{k+1}P'_n)$ for any $k\in\mathbb{N}$, then there also exists $N\le\dim(\mathcal{H})$ such that $\vee_{n=0}^{N}P'_n = \vee_{n=0}^{+\infty}P'_n$. Now we apply the rule BOUNDED WHILE2 to have

$$\frac{\begin{array}{c}\forall n<\dim(\mathcal{H}).\vdash [\mathrm{supp}(M_1P_nM_1^\dagger)]S'[ok{:}P_{n+1}]\\ \forall m\le N.\vdash [\mathrm{supp}(M_1P_mM_1^\dagger)]S'[er{:}P'_m]\end{array}}{\vdash [P_0]\,\mathbf{while}\ M[\bar{q}]=1\ \mathbf{do}\ S'\ \mathbf{od}[er{:}P'_m]}$$

Then apply the rule Disjunction by induction to have

$$\frac{\forall n \leq N. \vdash [P]\textbf{while } M[\bar{q}] = 1 \textbf{ do } S' \textbf{ od}[er{:}P'_n]}{\vdash [P]\textbf{while } M[\bar{q}] = 1 \textbf{ do } S' \textbf{ od}[er{:}\vee_{n=0}^{N} P'_n]}$$

Finally the triple is derivable by the rule Order assuming that $\vee_{n=0}^{N} P'_n \supseteq Q$.

$$\frac{\vdash [P]\textbf{while } M[\bar{q}] = 1 \textbf{ do } S' \textbf{ od}[er{:}\vee_{n=0}^{N} Q'_n] \qquad \vee_{n=0}^{N} P'_n \supseteq Q}{\vdash [P]\textbf{while } M[\bar{q}] = 1 \textbf{ do } S' \textbf{ od}[er{:}Q]}$$

It suffices to prove $\vee_{n=0}^{N} P'_n \supseteq Q$ by $\vDash [P]S[er{:}Q]$. Let $\rho = P/\mathrm{Tr}(P)$, by Lemma 2.18, we have

$$\forall \rho \dashv P \implies \textstyle\sum_{n=0}^{+\infty}(\sum((\mathcal{M}_1 \circ [\![S']\!]_{ok})^n \circ \mathcal{M}_1 \circ [\![S']\!]_{er})(P/\mathrm{Tr}(P))) = \vee_{n=0}^{N} P'_n \supseteq Q$$

■

The intuition is that the rank of $\vee_{i<N} P_i$ is bounded by $\dim(\mathcal{H})$, and once $\vee_{i \leq N-1} P_i = \vee_{i \leq N} P_i$, 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 postcondition is achievable. Theorem 3.9 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.

3.6 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 that Def. 3.4 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.

A Naive Generalization of IL Validity Formulation

We start with the following naive formulation.

Definition 3.10 (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 (3.1) 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 nontrivial postcondition Q can be satisfied by some unreachable pure state.

It should be noted that the unsuitability of the classical predicate for our projective predicate does not imply that classical predicates are infeasible in quantum reasoning. As shown in [CCL⁺23], they adopt to use a classical predicate that maps a pure state to either 0 or 1. Tree automata serve as a concise representation of these predicates, wherein the semantics of disjunction (conjunction) correspond to the union (intersection) of the tree automata. Implication among tree automata is subsequently determined through language inclusion tests.

A Weak Validity Based on Observable Relation

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

Definition 3.11 (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:

$$\rho \not\models P \Leftrightarrow \rho \succ \neg P. \tag{3.3}$$

The equation (3.3) is a generalization of its classical counterpart (3.2) by observing

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

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 into equality if we replace ρ and $|\psi\rangle$ with classical states. The relation between observable relation and under-approximation is stated in Lemma 3.12.

Lemma 3.12 (Relation between observable and under-approximation). *For any projection P and quantum state $\rho \in \mathcal{D}(\mathcal{H})$, we have*

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

Proof.

- "$\rho \dashv P \implies \forall |\psi\rangle \in P.\, \rho \succ |\psi\rangle$":

  By definition of $\rho \dashv P$, $|\psi\rangle \in P$ implies $|\psi\rangle \in \mathrm{supp}(\rho)$.

  Let $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ where $p_i > 0$, it implies $0 < 1 = \langle\psi| \mathrm{supp}(\rho) |\psi\rangle = \sum_i |\langle\psi|\psi_i\rangle|^2$. Thus $\langle\psi| \rho |\psi\rangle = \sum_i p_i |\langle\psi|\psi_i\rangle|^2 > 0$.

- "$\rho \succ P \implies \exists P'.\, \rho \dashv P' \wedge \mathrm{Tr}(PP') > 0$":

  Let $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$, $P = \sum_j |\phi_j\rangle\langle\phi_j|$ where $p_i > 0$.

  By $\rho \succ P$ we have $0 < \mathrm{Tr}(P\rho) = \sum_{ij} p_i |\langle\psi_i|\phi_j\rangle|^2$, thus $\exists i, j, |\langle\psi_i|\phi_j\rangle|^2 > 0$.

  Let $P' = |\psi_i\rangle\langle\psi_i|$ we have $\mathrm{Tr}(PP') > 0$ and obviously $P' \subseteq \mathrm{supp}(\rho)$.

■

On top of the observable relation, we give another generalization of the classical incorrectness triple that is better than Def. 3.10.

Definition 3.13 (Weak validity). A QIL triple is weakly valid, denoted by $\models_w [P]S[\epsilon\!:\!Q]$, 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 $\models_w [P]S[\epsilon\!:\!Q]$ 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 3.14.

Lemma 3.14 (Connection between weak and strong validities). *For any quantum program S, projections P, Q and exit condition ϵ,*

$$\models [P]S[\epsilon\!:\!Q] \implies \models_w [P]S[\epsilon\!:\!Q].$$

Conversely, given P and S, we have

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

Proof. The first implication is straightforward by applying Lemma 2.18, 3.12 and Theorem 3.2. For the second implication, given $Q \neq \mathbf{0}$, there is some $|\psi'\rangle \in Q$.

By definition of $\vDash_w [P]S[\epsilon : Q]$ and Theorem 3.2, we have $|\psi\rangle \in P$ and $\rho' \in [\![S]\!]_\epsilon(|\psi\rangle\langle\psi|)$ such that $\rho' \succ |\psi'\rangle$.

By Lemma 3.12, there is $Q' \neq \mathbf{0}$ such that $\rho' \dashv Q'$ and $0 < \mathrm{Tr}(Q'|\psi'\rangle\langle\psi'|) \leq \mathrm{Tr}(QQ')$.

By Lemma 2.18, for any $\rho \dashv P$, $\mathrm{supp}(\sum [\![S]\!]_\epsilon \rho) \supseteq \mathrm{supp}(\sum [\![S]\!]_\epsilon(|\psi\rangle\langle\psi|)) \supseteq \mathrm{supp}(\rho') \supseteq Q'$. ∎

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 3.14 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. 3.3a, 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., $P \not\subseteq \mathrm{supp}(\rho)$.

A Strict Validity Based on Under-Approximation

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

Definition 3.15 (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 $\vDash_s [\beta_{00}\beta_{00}^\dagger] q_0 := |0\rangle\,[ok\!:\!|0\rangle\langle 0|\,I \otimes I]$ 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. 3.3a.

The main drawback of the strict validity is too restrictive to preserve the disjunction rule below, thus not being 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** $(M[q]) = \textbf{true} \rightarrow \textbf{skip} \,\Box\, \textbf{false} \rightarrow \textbf{skip})$ **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 post-conditions $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$ is I, and a state $\rho' \dashv I$ 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. 3.4, the strong validity in Def. 3.4 supports the disjunction rule and thus is more efficient compared with other triples.

At the end of this section, we characterize the relationship between these validity formulations in the lemma 3.17, where $\vDash [P]S[\epsilon\!:\!Q]$ and $\vDash_w [P]S[\epsilon\!:\!Q]$ are strictly weaker than the others, thus the most expressive. The ancillary lemma 3.16 is introduced to prove lemma 3.17.

Lemma 3.16. *An incorrectness triple* $\vDash_s [S]P[\epsilon\!:\!Q]$ *holds if and only if*

$$\exists(\rho,\rho') \in [\![S]\!]_\epsilon.\, \mathrm{supp}(\rho) = P \wedge \rho' \Vdash Q$$

Proof. The "only if" part is straightforward from the Def. 3.15. For the "if" part, by the Lemma 2.20, there exists a super-operator $\mathcal{E}$ such that $\rho' = \mathcal{E}(\rho)$ for a given pair $(\rho,\rho') \in [\![S]\!]_{er}$. Thus we have

$$\begin{aligned}
&\exists(\rho,\rho') \in [\![S]\!]_{er}.\, \mathrm{supp}(\rho) = P \wedge \rho' \Vdash Q\\
\Rightarrow\ & \exists\rho,\mathcal{E}.\, \mathrm{supp}(\rho) = P \wedge \mathcal{E}(\rho) \Vdash Q \quad \text{(Lemma 2.20)}\\
\Rightarrow\ & \exists\rho,\mathcal{E}.\, \forall\sigma.\, \mathrm{supp}(\sigma) \supseteq \mathrm{supp}(\rho) = P.\, \mathrm{supp}(\mathcal{E}(\sigma)) \supseteq \mathrm{supp}(\mathcal{E}(\rho)) \Vdash Q \quad \text{(Lemma 2.14)}\\
\Rightarrow\ & \forall\sigma \Vdash P.\, \exists\mathcal{E}.\, (\sigma,\mathcal{E}(\sigma)) \in [\![S]\!]_\epsilon.\, \mathcal{E}(\sigma) \Vdash Q\\
\Rightarrow\ & \vDash_s [P]S[\epsilon\!:\!Q] \quad \text{(Def. 3.15)}
\end{aligned}$$

■

Lemma 3.17. *For arbitrary program S, we have*

$$\vDash_c [P]S[\epsilon\!:\!Q] \;\Rightarrow\; \vDash_s [P]S[\epsilon\!:\!Q] \;\Rightarrow\; \vDash [P]S[\epsilon\!:\!Q] \;\approx\; \vDash_w [P]S[\epsilon\!:\!Q],$$

where $\approx$ *means the weak equivalence described by Lemma 3.14. In particular, if* S *does not contain initialization statements* $q := |0\rangle$*, we have*

$$\vDash_c [P]S[\epsilon\!:\!Q] \;\Leftrightarrow\; \vDash_s [P]S[\epsilon\!:\!Q],$$

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

Proof. To see $\vDash_c [P]S[\epsilon\!:\!Q] \;\Rightarrow\; \vDash_s [P]S[\epsilon\!:\!Q]$, we have

$$\vDash_c [P]S[\epsilon\!:\!Q]$$

$$\begin{aligned}
&\Rightarrow \forall \rho' \vDash Q.\, \exists (\rho,\rho') \in [\![S]\!]_\epsilon .\, \rho \vDash P \quad \text{(Def. 3.10)}\\
&\Rightarrow \exists (\rho,\rho') \in [\![S]\!]_\epsilon .\, \rho \vDash P .\, \mathrm{supp}(\rho') = Q\\
&\Rightarrow \exists \rho, \mathcal{E}.\, (\rho, \mathcal{E}(\rho)) \in [\![S]\!]_\epsilon .\, \rho \vDash P .\, \mathrm{supp}(\mathcal{E}(\rho)) = Q \quad \text{(Lemma 2.20)}\\
&\Rightarrow \exists \rho, \sigma, \mathcal{E}.\, P = \mathrm{supp}(\sigma) \supseteq \mathrm{supp}(\rho).\, \mathrm{supp}(\mathcal{E}(\sigma)) \supseteq \mathrm{supp}(\mathcal{E}(\rho)) = Q \quad \text{(Lemma 2.14)}\\
&\Rightarrow \exists \sigma, \mathcal{E}.\, (\sigma, \mathcal{E}(\sigma)) \in [\![S]\!]_\epsilon .\, \mathrm{supp}(\sigma) = P .\, \mathcal{E}(\sigma) \dashv Q\\
&\Rightarrow \vDash_s [P]S[\epsilon : Q] \quad \text{(Lemma 3.16)}
\end{aligned}$$

It is direct to see $\vDash_s [P]S[\epsilon : Q] \Rightarrow \vDash [P]S[\epsilon : Q]$ by Def. 3.4 and Def. 3.15.

$$\begin{aligned}
&\vDash_s [P]S[\epsilon : Q]\\
&\Rightarrow \forall \rho \dashv P \Rightarrow \exists \rho' .\, (\rho,\rho') \in [\![S]\!]_\epsilon \wedge \rho' \dashv Q \quad \text{(Def. 3.15)}\\
&\Rightarrow \forall \rho \dashv P \Rightarrow \mathrm{supp}(\textstyle\sum [\![S]\!]_\epsilon(\rho)) \supseteq \mathrm{supp}(\rho') \dashv Q \quad \text{(Lemma 2.18)}\\
&\Rightarrow \vDash [P]S[\epsilon : Q] \quad \text{(Def. 3.4)}
\end{aligned}$$

$\vDash [P]S[\epsilon : Q] \approx \vDash_w [P]S[\epsilon : Q]$ has been described by Lemma 3.14.

On the other hand, if S does not contain initialization statements, i.e. $[\![S]\!]_\epsilon(\rho) = \{(K_i \rho K_i^\dagger, \lambda_i) \mid i \in \mathbb{Z}^+\}$, we can get tighter relations as follows,

$$\begin{aligned}
\vDash_s [P]S[\epsilon : Q] &\Rightarrow \forall \rho \dashv P .\, \exists (\rho,\rho') \in [\![S]\!]_\epsilon .\, \rho' \dashv Q \quad \text{(Def. 3.15)}\\
&\Rightarrow \exists K_i, \rho = P/\mathrm{Tr}(P).\, \mathrm{supp}(\rho) = P.\, \rho' = K_i \rho K_i^\dagger \dashv Q\\
&\Rightarrow \exists K_i .\, \mathrm{supp}(K_i P K_i^\dagger) \supseteq Q\\
&\Rightarrow \exists K_i .\, \forall \sigma' \vDash Q .\, \sigma' \vDash \mathrm{supp}(K_i P K_i^\dagger)\\
&\Rightarrow \exists \sigma, K_i .\, \sigma \vDash P .\, \sigma' = K_i \sigma K_i^\dagger \quad \text{(Lemma 2.15)}\\
&\Rightarrow \forall \sigma' \vDash Q .\, \exists (\sigma,\sigma') \in [\![S]\!]_\epsilon .\, \sigma \vDash P\\
&\Leftrightarrow \vDash_c [P]S[\epsilon : Q] \quad \text{(Def. 3.10)}
\end{aligned}$$

■

Chapter 4

Approximate Quantum Relational Hoare logic

Quantum computation is inevitably subject to imperfections in its implementation. The imperfection of the implementation is inevitable and arises from various sources in quantum computation. From the hardware aspect, the imperfection can come from modeling the hardware level of environment noise. From the software level, quantum algorithm designers may introduce noisy implementation in the flavor of lower-depth computation and other features. The relational logic provides significant advantages in program reasoning and the importance of assessing the robustness of quantum programs between their ideal specifications and imperfect implementations. However, the exploration of approximate relational properties, vital for assessing the robustness of quantum programs between their ideal specifications and imperfect implementations, is rarely studied from the perspective of program logic. In this chapter, 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 (5.6). Furthermore, we validate the correctness of bit flip code (5.4) and the algorithm for the approximation of unitary (5.5). 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.

Organization of Chapter 4. This chapter is organized as follows. Sec. 4.1 outlines the syntax and semantics of the quantum while language under investigation, and presents the concepts of quantum approximate couplings and lifting. We formally define of our aqRHL judgments and introduce the concept of approximate measurement conditions in Sec. 4.2. Sec. 4.3 presents the corresponding proof system based on the specification of aqRHL judgments. In Sec. 4.4, we prove that our proof system is sound. We give some discussions on aqRHL judgments and separability problem in Sec. 4.5. The case studies of low-depth approximation quantum Fourier transform, bit flip code, and repeat until success can be found in Chapter 5.

4.1 Quantum Approximate Coupling and Lifting

This section begins with a review of trace and diamond norms, followed by our proposal of the approximate quantum couplings and liftings used in our aqRHL.

Trace distance & Diamond Norm

In this subsection, we review the quantum generalization of the classical trace distance, a commonly used metric for quantifying the difference between two quantum states. The definition is presented below, where we extend it directly to partial density operators.

Definition 4.1 (trace distance). The trace distance of any two partial density operators ρ and σ is defined as follows:

$$D(\rho, \sigma) \equiv \frac{1}{2} \operatorname{Tr} |\rho - \sigma|$$

where $|A| = \sqrt{A^\dagger A}$ for any operator A, i.e. the positive square root of $A^\dagger A$.

Here are some important properties of trace distance that will aid in understanding the proposed approximate quantum couplings later on.

Lemma 4.2. *Properties of trace distance.*

(1) *Trace distance is a metric on the space of partial density operators. To be specific, $D(\rho,\sigma) = 0$ if and only if $\rho = \sigma$; $D(\rho,\sigma) = D(\sigma,\rho)$; $D(\rho_1,\rho_3) \le D(\rho_1,\rho_2) + D(\rho_2,\rho_3)$. Also, we have $0 \le D(\rho,\sigma) \le 1$.*

(2) *Let $\mathcal{E}$ be a superoperator, ρ and σ be two partial density operators, then $D(\mathcal{E}(\rho), \mathcal{E}(\sigma)) \le D(\rho,\sigma)$.*

(3) (Convexity) *Let $\{p_i\}$ and $\{q_i\}$ be two arbitrary probability distributions, and ρ_i and σ_i be partial density operators over the same index set. Then, $D(\sum_i p_i\rho_i, \sum_i q_i\sigma_i) \le D(p_i,q_i) + \sum_i p_i D(\rho_i,\sigma_i)$. A special case is $D(\sum_i p_i\rho_i, \sum_i p_i\sigma_i) \le \sum_i p_i D(\rho_i,\sigma_i)$ when $\{p_i\}$ and $\{q_i\}$ are of the same distribution.*

(4) *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be two superoperators. If $D(\rho_1,\rho_2) \le \delta$ and $D(\mathcal{E}_1(\rho),\mathcal{E}_2(\rho)) \le \delta'$ hold for any ρ, then we have $D(\mathcal{E}_1(\rho_1),\mathcal{E}_2(\rho_2)) \le \delta + \delta'$.*

The trace distance is a useful *measurable* metric to distinguish two quantum states directly, but more is needed for comparing two superoperators $\mathcal{E}_1$ and $\mathcal{E}_2$. An intuitive but inadequate way is to find an optimal input state ρ that maximizes the trace distance between $\mathcal{E}_1(\rho)$ and $\mathcal{E}_2(\rho)$ because this does not take entanglement into account. It is known that quantum entanglement is useful for channel discrimination [PW09]. To address this issue, Kitaev proposed the *diamond norm* [AKN98a] to distinguish between two superoperators sufficiently with the help of the power of quantum entanglement by introducing auxiliary qubits.

Definition 4.3 (Diamond Norm). Let A be a linear operator over Hilbert space $\mathcal{H}$, the diamond norm of A is defined as

$$\|A\|_\diamond \;\equiv\; \max_{\rho\in\mathcal{D}(\mathcal{H}\otimes\mathcal{H}')} \frac{1}{2}\operatorname{Tr}|(A\otimes I_{\mathcal{H}'})\rho| \tag{4.1}$$

where $\mathcal{H}'$ denotes any auxiliary subspace that can be assumed to be a copy of $\mathcal{H}$ without loss of generality. The factor $1/2$ is added to keep consistent with trace distance.

Consequently, the distance between superoperators $\mathcal{E}_1$ and $\mathcal{E}_2$ over $\mathcal{H}$ is given as

$$\|\mathcal{E}_1-\mathcal{E}_2\|_\diamond \;\equiv\; \max_{\rho\in\mathcal{D}(\mathcal{H}\otimes\mathcal{H}')} D((\mathcal{E}_1\otimes I_{\mathcal{H}'})(\rho),(\mathcal{E}_2\otimes I_{\mathcal{H}'})(\rho)) \tag{4.2}$$

It is straightforward to verify that $D(\mathcal{E}_1(\sigma),\mathcal{E}_2(\sigma)) \le \|\mathcal{E}_1-\mathcal{E}_2\|_\diamond$ for any $\sigma\in\mathcal{D}(\mathcal{H})$ by choosing $\rho=\sigma\otimes I_{\mathcal{H}'}$. Therefore, we can use the diamond norm as an upper bound for the trace distance between quantum states. Compared with directly calculating $D(\mathcal{E}_1(\sigma),\mathcal{E}_2(\sigma))$, the diamond norm allows us to take advantage of quantum entanglement in $\rho\in\mathcal{D}(\mathcal{H}\otimes I_{\mathcal{H}'})$ to better distinguish between $\mathcal{E}_1$ and $\mathcal{E}_2$.

The distance between two superoperators can be computed efficiently by converting it into a semidefinite programming problem [Wat13]. In the case where two superoperators $\mathcal{E}_1=U_1\cdot U_1^\dagger$ and $\mathcal{E}_2=U_2\cdot U_2^\dagger$ are unitaries over $\mathcal{H}$, there is no need to introduce any auxiliary system for optimal discrimination [Wat18]. The *numerical range* of an operator A over $\mathcal{H}$ is defined as the set $\mathcal{N}(A)=\{\langle\psi|A|\psi\rangle : |\psi\rangle\in\mathcal{H}, |\psi\rangle\neq 0\}$ which is compact and convex. Since $U_1^\dagger U_2$ is normal, $\mathcal{N}(U_1^\dagger U_2)$ is directly equal to the convex hull of the eigenvalues of $U_1^\dagger U_2$. Consequently, we can express the distance between U_1 and U_2 as

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

where α is the smallest arc containing the spectrum of $U_1^\dagger U_2$ [NPPŻ18]. To be specific, let $\{\lambda_i\}(|\lambda_i|=1)$ be the eigenvalues of the unitary $U_1^\dagger U_2$. We arrange these eigenvalues along the unit circle in the complex plane, and α is the smallest arc on the unit circle that contains all of these eigenvalues.[1] We present some important properties of the diamond norm below.

Lemma 4.4. *Properties of the diamond norm.*

(1) Diamond norm is a metric over superoperators.

(2) For any superoperators $\mathcal{E}$ and partial density operator ρ, we have

$$\|\mathcal{E}(\rho)\|_\diamond \leq \|\mathcal{E}\|_\diamond \cdot \mathrm{Tr}(\rho)$$

(3) For any superoperators $\mathcal{E}_1$, $\mathcal{E}_1'$, $\mathcal{E}_2$ and $\mathcal{E}_2'$ over $\mathcal{H}$, we have

$$\|\mathcal{E}_2 \circ \mathcal{E}_1 - \mathcal{E}_2' \circ \mathcal{E}_1'\|_\diamond \leq \|\mathcal{E}_1 - \mathcal{E}_1'\|_\diamond + \|\mathcal{E}_2 - \mathcal{E}_2'\|_\diamond$$

(4) For any superoperators $\mathcal{E}_1$ and $\mathcal{E}_1'$ over $\mathcal{H}_1$, $\mathcal{E}_2$ and $\mathcal{E}_2'$ over $\mathcal{H}_2$, we have

$$\|\mathcal{E}_1 \otimes \mathcal{E}_2 - \mathcal{E}_1' \otimes \mathcal{E}_2'\|_\diamond \leq \|\mathcal{E}_1 - \mathcal{E}_1'\|_\diamond + \|\mathcal{E}_2 - \mathcal{E}_2'\|_\diamond$$

Approximate Quantum Couplings

The exact quantum coupling between two quantum states, defined in the paper [BHY+19], is a natural generalization of the classical coupling between two discrete distributions. 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. In the quantum world, we have a natural association between the density matrix and probability distribution, where the partial trace over a quantum state corresponds to the marginalizing over a probability distribution. We review the definition of exact quantum coupling in [BHY+19] as follows.

[1]Readers may refer to [NPPŻ18] for a graphical representation of α.

Definition 4.5 (Exact 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 *coupling* for $\langle \rho_1, \rho_2 \rangle$ if $\mathrm{Tr}_1(\rho) = \rho_2$ and $\mathrm{Tr}_2(\rho) = \rho_1$.

The partial trace operation is not a one-to-one function, so the exact quantum coupling for a given pair of density matrices ρ_1 and ρ_2 may not be unique. Therefore, the exact quantum coupling can be seen as a weak reverse process of the partial trace operation. For any pair of density matrices ρ_1 and ρ_2 with equal trace, we can always find a trivial coupling where the two subsystems are independent of each other, which is $\rho_1 \otimes \rho_2/\mathrm{Tr}(\rho_1)$. A key feature of the exact quantum coupling ρ for a given pair $\langle \rho_1, \rho_2 \rangle$ is that its trace is equal to the trace of both ρ_1 and ρ_2, i.e., $\mathrm{Tr}(\rho) = \mathrm{Tr}(\rho_1) = \mathrm{Tr}(\rho_2)$. However, this constraint can be relaxed in the approximate version.

In the paper [BKOZB13], Barthe et al. proposed the approximate classical coupling for reasoning about differential privacy. They designed a parameterized α-distance between sub-distributions to serve as an upper bound for the approximation. Inspired by their work, we use trace distance to measure the approximation without loss of generality. The approximate quantum coupling degenerates into its counterpart in [BHY+19] if $\delta = 0$. The deviation δ is introduced to quantify the distance between the exact and approximate coupling of ρ_1 and ρ_2, rather than a direct comparison between ρ_1 and ρ_2.

Definition 4.6 (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 an *δ-coupling* for $\langle \rho_1, \rho_2 \rangle$ if

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

Approximate Quantum Liftings

To formulate our judgments, we introduce projections as relations between partial density operators. Similar to the classical case, a valid approximate quantum lifting implies the existence of an approximate quantum coupling that satisfies a quantum predicate.

Definition 4.7 (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 an 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 $\delta = 0$. We usually select a "closest" witness with a smallest deviation δ for the approximate quantum coupling $\rho_1 \sim \rho_2$. However, if we want to compare them under a specific condition P, that's where approximate liftings come into play. For example, give two distinct states $\rho_1 = |+\rangle\langle+|$ and $\rho_2 = |-\rangle\langle-|$, an exact witness $|+-\rangle\langle+-|$ is suitable to exactly describe the pair $\langle \rho_1, \rho_2 \rangle$ with $\delta = 0$. How about the witness of these two states under the computational basis measurement $\{P_{ij} = |ij\rangle\langle ij|\}(i, j \in \{0, 1\})$? For any P_{ij}, the closeset witness for the lifting $\rho_1 \sim_{P_{ij}} \rho_2$ is always the trivial state $\mathbf{0}$ with $\delta = 1/2$. The trivial wintess with the same non-zero δ implies that the computational basis measurement cannot distinguish the pair $\langle \rho_1, \rho_2 \rangle$ at all.

We list the basic properties of approximate quantum listings below.

Lemma 4.8. *In the following, ρ_1, ρ_2, $\alpha\rho_1$, $\alpha\rho_2$, $\rho_1 + \rho_3$ and $\rho_2 + \rho_4$ are partial density matrices.*

(1) (Scalability) *If $\rho_1 \sim_P^\delta \rho_2$ and $0 \leq \alpha$, then we have $(\alpha\rho_1) \sim_P^{\alpha\delta} (\alpha\rho_2)$.*

(2) (Linearity) *If $\rho_1 \sim_P^{\delta_1} \rho_2$ and $\rho_3 \sim_P^{\delta_2} \rho_4$, then we have $(\rho_1 + \rho_3) \sim_P^{\delta_1+\delta_2} (\rho_2 + \rho_4)$.*

(3) (Monotonicity) *If $\delta \leq \delta'$ and $P \subseteq P'$, we have $\rho_1 \sim_P^\delta \rho_2 \implies \rho_1 \sim_{P'}^{\delta'} \rho_2$.*

Proof. **(1)** It is direct from the fact $D(\alpha\rho, \alpha\sigma) = \alpha D(\rho, \sigma)$ for any partial density matrices ρ, $\alpha\rho$, σ, $\alpha\sigma$. Therefore, if ρ is a witness of $\rho_1 \sim_P^\delta \rho_2$, then $\alpha\rho$ is the witness of $(\alpha\rho_1) \sim_P^{\alpha\delta} (\alpha\rho_2)$.

(2) Let σ_1 and σ_2 be the witness of the lifting $\rho_1 \sim_P^{\delta_1} \rho_2$ and $\rho_3 \sim_P^{\delta_1} \rho_4$ respectively, then $\sigma_1 + \sigma_2$ is also the witness of the lifting $(\rho_1 + \rho_3) \sim_P^{\delta_1+\delta_2} (\rho_2 + \rho_4)$ by the convexity of trace distance. That is, $\sigma_1 + \sigma_2 \vDash P$, and we also have

$$D(\mathrm{Tr}_2(\sigma_1 + \sigma_2), \rho_1 + \rho_3) \le D(\mathrm{Tr}_2(\sigma_1), \rho_1) + D(\mathrm{Tr}_2(\sigma_2), \rho_3) \le \delta_1 + \delta_2$$

and the same goes for $D(\mathrm{Tr}_1(\sigma_1 + \sigma_2), \rho_2 + \rho_4) \le \delta_1 + \delta_2$.

(3) Let ρ be the witness of the lifting $\rho_1 \sim_P^\delta \rho_2$, then it is direct to check $\mathrm{supp}(\rho) \subseteq P \subseteq P'$, $D(\mathrm{Tr}_2(\rho), \rho_1) \le \delta \le \delta'$ and $D(\mathrm{Tr}_1(\rho), \rho_2) \le \delta \le \delta'$, thus ρ is also the witness of the lifting $\rho_1 \sim_P^\delta \rho_2 \implies \rho_1 \sim_{P'}^{\delta'} \rho_2$.

■

Quantum Equivalence

One of the most important quantum predicates is the equivalence relation between two registers. Let $\rho \in \mathcal{D}(\mathcal{H})$ be a partial density operator with the spectral decomposition $\rho = \sum_i \lambda_i |i\rangle\langle i|$, where $\{|i\rangle\}$ is an orthonormal basis of $\mathcal{H}$. It is straightforward to verify that $\sum_i \lambda_i |ii\rangle\langle ii| \in \mathcal{H} \otimes \mathcal{H}$ is a witness of the exact lifting $\rho \sim_\equiv \rho$, where $\equiv$ represents the quantum equivalence(or identity relation), as formally defined below [BHY+19].

Definition 4.9. 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 Def 4.9 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 need to use the projection $(I + SWAP)/2$ to represent the corresponding symmetric space. By doing so, we have $(I + SWAP)(|+\rangle \otimes |+\rangle)/2 = |+\rangle \otimes |+\rangle$,that is, $\operatorname{supp}(|+\rangle \otimes |+\rangle) \subseteq (I + SWAP)/2$.

Here we have an approximate version of the lemma for the identity relation in [BHY+19], where the approximate lifting for the identity relation has the same expressiveness as trace distance. This lemma implies that we can use approximate quantum liftings for the identity relation to reason about the approximate equivalence of quantum states.

Lemma 4.10. *For any ρ_1, ρ_2, we have*

$$\rho_1 \sim^{\delta}_{\equiv} \rho_2 \ \Leftrightarrow\ D(\rho_1, \rho_2) \leq 2\delta$$

Particularly, if $\delta = 0$, we have

$$\rho_1 \sim_{\equiv} \rho_2 \ \Leftrightarrow\ \rho_1 = \rho_2$$

Proof. From the left to right side, let ρ be the witness of the lifting $\rho_1 \sim^{\delta}_{\equiv} \rho_2$, we have

$$\operatorname{supp}(\rho) \subseteq \equiv \qquad D(\rho_1, \operatorname{Tr}_2(\rho)) \leq \delta \qquad D(\operatorname{Tr}_1(\rho), \rho_2) \leq \delta$$

Notice that $\operatorname{Tr}_1(\rho) = \operatorname{Tr}_2(\rho)$ from $\operatorname{supp}(\rho) \subseteq \equiv$. Then we have

$$\begin{aligned} D(\rho_1, \rho_2) &\leq D(\rho_1, \operatorname{Tr}_1(\rho)) + D(\operatorname{Tr}_1(\rho), \rho_2) \\ &= D(\rho_1, \operatorname{Tr}_2(\rho)) + D(\operatorname{Tr}_1(\rho), \rho_2) = \delta + \delta = 2\delta \end{aligned}$$

From the right to left side, let $\sigma = (\rho_1 + \rho_2)/2$, the eigen-decomposition of σ is $\sigma = \sum_i \lambda_i |\psi\rangle\langle\psi|$. We need to check $\rho = \sum_i \lambda_i |\psi\rangle\langle\psi| \otimes |\psi\rangle\langle\psi|$ is a witness of the lifting

$\rho_1 \sim^{\delta}_{\equiv} \rho_2$. It is direct to check $\mathrm{supp}(\rho) \subseteq \equiv$. Furthermore, we have

$$D(\mathrm{Tr}_1(\rho), \rho_2) = D(\sigma, \rho_2) = D((\rho_1 + \rho_2)/2, \rho_2) = D(\rho_1, \rho_2)/2 \leq \delta,$$

where $\mathrm{Tr}_1(\rho) = \mathrm{Tr}_2(\rho) = \sigma$, and the same goes for $D(\mathrm{Tr}_2(\rho), \rho_1) \leq \delta$. Thus $\rho_1 \sim^{\delta}_{\equiv} \rho_2$ holds. The particular case is straightforward since $\rho_1 = \rho_2$ if and only if $D(\rho_1, \rho_2) = 0$. ■

Notice that the trace distance provides a natural explanation for state discrimination, as presented in the lemma 4.11. Together with lemma 4.10, the approximate lifting $\rho_1 \sim^{\delta}_{\equiv} \rho_2$ also demonstrates the extent to which two states can be effectively distinguished through POVM measurements.

Lemma 4.11. *[NC11] Let $\{E_m\}$ be a POVM, with $p_m = \mathrm{Tr}(E_m\rho_1)$ and $q_m = \mathrm{Tr}(E_m\rho_2)$ as the probabilities of obtaining a measurement outcome labeled by m. Then we have*

$$D(\rho_1, \rho_2) = \max_{\{E_m\}} D(p_m, q_m),$$

where the maximization is over all POVMs $\{E_m\}$. $D(p_m, q_m)$ denotes the L_1 distance between distributions, i.e., $D(p_m, q_m) = \frac{1}{2}\sum_m |p_m - q_m|$.

Upper Bound of Approximation

The approximation usually arises in a scenario 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

$$\forall \rho_1, \rho_2, \rho_1 \sim_A \rho_2 \implies \rho_1 \sim^{\delta}_B \rho_2 \tag{4.3}$$

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 $\rho_1 \sim^{\delta}_B \rho_2$? The optimal deviation δ in Eq. 4.3 is equivalent to the following quantity,

$$\delta = d(A, B) = \sup_{\rho \models A} \inf_{\sigma \models B} \max\{D(\mathrm{Tr}_1(\rho), \mathrm{Tr}_1(\sigma)), D(\mathrm{Tr}_2(\rho), \mathrm{Tr}_2(\sigma))\} \tag{4.4}$$

where $d(A, B)$ can be upper bounded by $\sup_{\rho \models A} \inf_{\sigma \models B} D(\rho, \sigma)$ introduced in [ZYY19].

Here we discuss a simple but important instance of Eq. 4.3 with $A = (U_1 \otimes U_2)B(U_1 \otimes U_2)^\dagger$ and B being the quantum equivalence predicate $\equiv$. Then Eq. 4.3 can be represented as follows,

$$\forall \rho_1, \rho_2, \rho_1 \sim_{\equiv} \rho_2 \;\Rightarrow\; U_1\rho_1U_1^\dagger \sim_A U_2\rho_2U_2^\dagger \;\Rightarrow\; U_1\rho_1U_1^\dagger \sim_{\equiv}^{\delta} U_2\rho_2U_2^\dagger$$

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 [AKN98a] can better distinguish between two superoperators with the help of the power of quantum entanglement by introducing auxiliary qubits.

4.2 Specification Formula

Judgment and Validity

Our logic, called aqRHL, "approximates" the quantum relational Hoare logic described in [BHY+19]. 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 4.12 (Validity). The judgement $S_1 \sim_\delta S_2 : A \Rightarrow B$ is valid, written as

$$\models S_1 \sim_\delta S_2 : A \Rightarrow B \tag{4.5}$$

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 Def 4.12, we choose projective predicates [BN36] over the joint system of two programs because such predicates are the quantum analog of binary relations, the predicates used in pRHL [BKOZB13]. Moreover, this definition will become a judgment of [BHY+19] 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 judgment 4.5, where δ characterizes the deviation of approximation. We choose the set of projective predicates because it is the quantum generalization of the binary relation in approximate relational Hoare logic.

Lemma 4.13 (Program Equivalence). *Program S_1 is equivalent to program S_2,*[2] *if and only if $\vDash S_1 \sim S_2 : \equiv \Rightarrow \equiv$.*

Proof. It is direct form Def. 4.12 and lemma 4.10. From the left to the right side, let $\sigma = [\![S_1]\!](\rho) = [\![S_2]\!](\rho)$ for any ρ. The eigen-decompositions of ρ and σ are $\rho = \sum_i \alpha_i |\psi\rangle\langle\psi|$ and $\sigma = \sum_i \beta_i |\varphi\rangle\langle\varphi|$. It is direct to see that $\sum_i \alpha_i |\psi\rangle\langle\psi| \otimes |\psi\rangle\langle\psi|$ and $\sum_i \beta_i |\varphi\rangle\langle\varphi| \otimes |\varphi\rangle\langle\varphi|$ are witnesses for liftings $\rho \sim_\equiv \rho$ and $[\![S_1]\!](\rho) \sim_\equiv [\![S_2]\!](\rho)$. Thus, we have $\vDash S_1 \sim S_2 : \equiv \Rightarrow \equiv$ by Def. 4.12. From the right to the left side, given $\vDash S_1 \sim S_2 : \equiv \Rightarrow \equiv$, we have $\rho_1 \sim_\equiv \rho_2 \;\Rightarrow\; [\![S_1]\!](\rho_1) \sim_\equiv [\![S_2]\!](\rho_2)$ for any ρ. Then we have $\rho_1 = \rho_2$ and $D([\![S_1]\!](\rho_1), [\![S_2]\!](\rho_2)) = 0$ by lemma 4.10, that is $[\![S_1]\!](\rho) = [\![S_2]\!](\rho)$. ■

The program equivalence can be expressed concisely by judgment 4.5 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 simple example shows that quantum program equivalence is more complex than its classical counterpart due to the superposition of quantum states.

Example 4.14. *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*

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

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

$$[\![S_1]\!](|0\rangle\langle 0|) = [\![S_2]\!](|0\rangle\langle 0|) \qquad [\![S_1]\!](|1\rangle\langle 1|) = [\![S_2]\!](|1\rangle\langle 1|) \tag{4.6}$$

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

$$S_1 ::= \textbf{skip} \qquad S_2 ::= q := Z[q]$$

it is clear that S_1 and S_2 are not equivalent[3] *although Eq. 4.6 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.*

Measurement Conditions

Additional constraints must be imposed on the programs to establish feasible relational proof rules for programs with complex structures, such as if and loop statements. In the classical pRHL approach in [BGZB09], 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 [BHY+19], 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,

$$\mathrm{Tr}(M_1^m \rho_1 M_1^{m\dagger}) = \mathrm{Tr}(M_2^m \rho_2 M_2^{m\dagger}).$$

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

To study more general programs, we propose the approximate measurement conditions, which establish appropriate upper bounds for the deviations in our judgments.

Definition 4.15 (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

$$\mathcal{M}_1 \approx_{\{\delta_m\}} \mathcal{M}_2 : A \Rightarrow \{(p_m, B_m)\} \tag{4.7}$$

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 $p_m = 1$.

Our approximate measurement condition extends its counterpart in [BHY$^+$19] to an approximate version that no longer requires two measurements $\mathcal{M}_1$ and $\mathcal{M}_2$ must produce the same distribution. Instead, it employs information to bound the probability of entering the branches of index m, making approximate reasoning about branches possible. It is worth noting that the deviations in Def. 4.15 slightly differ from those in Def. 4.12. In Def. 4.12, programs S_1 and S_2 are trace-preserving if they can terminate. However, it is generally *not* the case that $\mathrm{Tr}(M_i^m \rho_i M_i^{m\dagger}) = \mathrm{Tr}(\rho_i)$ for $i \in \{1, 2\}$. Only the whole process of quantum measurement is trace-preserving, i.e., $\sum_m \mathrm{Tr}(M_i^m \rho_i M_i^{m\dagger}) = \mathrm{Tr}(\rho_i)$. As a result, δ_m specified in Definition 4.15 needs to work with p_m to establish bounds on the approximate reasoning.

Let $Q_1 = \textbf{while } \mathcal{M}_1[\bar{q}] = 1 \textbf{ do } S_1 \textbf{ od}$ and $Q_2 = \textbf{while } \mathcal{M}_2[\bar{q}] = 1 \textbf{ do } S_2 \textbf{ od}$ be two loop statements. In [BHY$^+$19], the measurement condition requires that Q_1 and Q_2 should produce the same distribution over measurement outcomes during the iterations, that is,

$$\mathrm{Tr}((\llbracket S_1 \rrbracket \circ \mathcal{M}_1^1)^k(\rho_1)) = \mathrm{Tr}((\llbracket S_2 \rrbracket \circ \mathcal{M}_2^1)^k(\rho_2))$$

holds for any integer $0 \leq k$. Such a condition limits the applications since it is generally not true.

Instead, we use the approximate measurement condition in loop statements. As shown in the rule (LP) in the next subsection, we introduce an "approximate invariant" A related to the approximate measurement condition. The approximate measurement condition indicates that loop statements have the following property,

$$\forall 0 \leq k. \, \rho \sim_A \sigma \implies \exists \gamma_k \; s.t. \begin{cases} \gamma_k \text{ is a witness of } \rho_k \sim_A^{d_k} \sigma_k \\ \mathrm{Tr}(\gamma_k) \leq p_1^k \end{cases}$$

where $d_k = \frac{1-p_1^k}{1-p_1}\delta_1 + \frac{p_1 - p_1^{k+1}}{1-p_1}\delta$, $\rho_k = ([\![S_1]\!] \circ \mathcal{M}_1^1)^k(\rho_1)$, $\sigma_k = ([\![S_2]\!] \circ \mathcal{M}_2^1)^k(\rho_2)$, $i \in \{1, 2\}$. Since the measurement condition performs approximate liftings based on the approximate output of the last iteration, the deviation accumulated in each iteration is scaled down by p_1. Notice that the measurement condition in rule (LP) puts limitations on the witness γ_k rather than actual program states ρ_k and σ_k. It usually can not give valid upper bounds on $\mathrm{Tr}(\rho_k)$ and $\mathrm{Tr}(\sigma_k)$. An exceptional case is the exact reasoning, where we have $\mathrm{Tr}(\rho_k) = \mathrm{Tr}(\sigma_k) = \mathrm{Tr}(\gamma_k) \leq p_1^k$ with $\delta_1 = \delta = 0$.

How about the case when finding a proper "approximate invariant" is not easy? In practice, the approximate reasoning usually stops after unrolling the loops deep enough, and the minor remaining parts are ignored. The loop statement can be approximated as a finite-length sequence of quantum measurement and loop body, and proper approximate measurement conditions can be set to terminate the loop somewhere and provide bounds on the approximation of the truncation. As shown in the rule (LP*), we usually need a bunch of approximate measurement conditions parameterized by the number of iterations,

$$\mathcal{M}_1 \approx_{\{\alpha_k, \beta_k\}} \mathcal{M}_2 : A_k \Rightarrow \{(q_k, C), (p_k, B_k)\}$$

to work with the premises $\vdash S_1 \sim_{\delta_k} S_2 : B_k \implies A_{k+1}$ related to the loop bodies. If compared loop statements could terminate within N-iterations, then the approximate measurement condition

$$\mathcal{M}_1 \approx_{\{\alpha_N,0\}} \mathcal{M}_2 : A_N \Rightarrow \{(1,C),(0,I)\}$$

is unnecessary. Otherwise, an upper bound N on the unrolling of loops is needed, and the above approximate measurement condition is used to approximate the truncation.

4.3 Proof System

This section will present a set of proof rules for reasoning about the validity of aqRHL judgments. These rules include construct-specific rules (two-sided and one-sided) and structural ones, as is typical in classical relational Hoare logic.

Two-side Rules

(Skip) $\vdash \mathbf{skip} \sim \mathbf{skip} : A \Rightarrow A$

(Init) $\vdash \bar{q}_1 := |0\rangle \sim \bar{q}_2 := |0\rangle : A \Rightarrow |0\rangle_{\bar{q}_1}\langle 0| \otimes |0\rangle_{\bar{q}_2}\langle 0| \otimes \mathrm{proj}(\mathrm{Tr}_{(\bar{q}_1,\bar{q}_2)}(A))$

(Ut) $\vdash \bar{q}_1 := U_1[\bar{q}_1] \sim \bar{q}_2 := U_2[\bar{q}_2] : A \Rightarrow (U_1 \otimes U_2)A(U_1^\dagger \otimes U_2^\dagger)$

(Qo) $\vdash \bar{q}_1 := \mathcal{E}_1[\bar{q}_1] \sim \bar{q}_2 := \mathcal{E}_2[\bar{q}_2] : A \Rightarrow \mathrm{proj}((\mathcal{E}_1 \otimes \mathcal{E}_2)(A))$

(Seq) $$\frac{\vdash S_1 \sim_{\delta_1} S_2 : A \Rightarrow R \quad \vdash S_1' \sim_{\delta_2} S_2' : R \Rightarrow B}{\vdash S_1; S_1' \sim_{\delta_1+\delta_2} S_2; S_2' : A \Rightarrow B}$$

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

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

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

Figure 4.1: Two-sided aqRHL rules. The deviation of the LP* rule 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 $\lambda_n = \prod_{k=0}^{n} p_k$.

Fig. 4.1 includes the two-sided proof rules when comparing programs with the same structure. The basic rules, namely (Skip), (Init), (Ut) and (Qo) are similar to their counterparts in [BHY+19] 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 along with the sequences, which directly comes from the triangle inequality of trace distance.

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 [BKOZB13, BHY+19] or the speed bound at which loops converge [HHZ+19]. 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 1 at the beginning, we can unroll loop statements several times to make p_1 less than 1.

We derive rule (LP*) as an alternative of rule (LP) by incorporating more specific measurement conditions for the iterations of loops when we can not find a good preidcate A for rule (LP). When doing approximate reasoning about loops, setting an upper bound N on the number of iterations is typical. 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.

One-side Rules

One-sided relational proof rules are listed in Fig. 4.2. These rules are necessary when two programs do not share the same structure. We have only listed the one-side rules for the left side, and similar rules apply to the right side symmetrically. Since we can always make the **skip** statement share the same probability distribution with any **if** and **while** statement, thus the measurement conditions in rules (IF-L), (LP-L), and (LP*-L) are more straightforward.

$$\text{(INIT-L)} \quad \vdash \bar{q}_1 := |0\rangle \sim \textbf{skip} : A \Rightarrow |0\rangle_{\bar{q}_1}\langle 0| \otimes \text{proj}(\text{Tr}_{\mathcal{H}_{\bar{q}_1}}(A))$$

$$\text{(UT-L)} \quad \vdash \bar{q}_1 := U_1[\bar{q}_1] \sim \textbf{skip} : A \Rightarrow (U_1 \otimes I_2)A(U_1^\dagger \otimes I_2)$$

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

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

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

Figure 4.2: One-sided aqRHL rules.

Structural Rules

Unlike classical programs, the potential quantum entanglement between subsystems brings a unique challenge in constructing a general frame rule for quantum programs [BHY+19, Unr19, ZBH+21]. We derive a simple frame rule (FRAME) to specify a particular instance that the predicate C on additional independent system $(\bar{r}_1, \bar{r}_2)$ is one-dimensional. The subscripts related to registers are displayed explicitly to avoid confusion. Rule (ORDER) adds an order relation $\le$ over deviations. In addition, a side condition is introduced in rule (APPROX) to allow switching postconditions at the cost of bringing approximation.

Rules for Equivalence Relation

We address a scenario regarding the rules (UT), where the precondition and postcondition are equivalence relations. We use diamond norm to bound the deviation in rule

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

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

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

Figure 4.3: Structural aqRHL rules.

(Ut-id), where $U\cdot U^\dagger$ denotes the Kraus representation [NC11] of unitary U. The rule (Comp) permits reasoning equivalence between programs by introducing intermediate programs.

$$\text{(Ut-id)}\quad \vdash \bar{q}_1 := U_1[\bar{q}_1] \sim_{\|U_1\cdot U_1^\dagger - U_2\cdot U_2^\dagger\|_\diamond/2} \bar{q}_2 := U_2[\bar{q}_2] : \equiv \Rightarrow \equiv$$

$$\text{(Comp)}\quad \frac{\vdash S_1 \sim_{\delta_1} S_2 : \equiv \Rightarrow \equiv \qquad \vdash S_2 \sim_{\delta_2} S_3 : \equiv \Rightarrow \equiv}{\vdash S_1 \sim_{\delta_1+\delta_2} S_3 : \equiv \Rightarrow \equiv}$$

Figure 4.4: Rules for Equivalence Relation

4.4 Soundness Theorem

All these proof rules in Fig. 4.1, 4.2, 4.3 and 4.4 are proved to be sound concerning the validity of judgments defined in Def. 4.12. Our logic has no relative completeness due to its foundation in the quantum extension of probabilistic coupling, which is insufficient for demonstrating the convergence of Markov chains. Ancillary lemmas 4.17, 4.18 and 4.19 are introduced to prove theorem 4.16.

Theorem 4.16. [Soundness] *For any program S_1, S_2, projections A and B, deviation δ, we have,*

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

Proof. The soundness is proved by showing the validity of axioms and inference rules in Fig. 4.1, Fig. 4.2, Fig. 4.3 and Fig. 4.4 concerning Def. 4.12 by the induction on the proof structural of $\vdash S_1 \sim_\delta S_2 : A \Rightarrow B$. 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. O_1 and O_2 to represent operators over $\mathcal{H}_{\bar{q}_1}$ and $\mathcal{H}_{\bar{q}_2}$. $\mathrm{Tr}_2(\rho)$ and $\mathrm{Tr}_1(\rho)$ to denote the reduced density matrix over $\mathcal{H}_{q_1}$ and $\mathcal{H}_{q_2}$, respectively.

(1) (Skip). It is straight from Def. 4.12.

(2) (Ut). For any ρ_1 and ρ_2 such that there exist a witness ρ of the lifting $\rho_1 \sim_A \rho_2$, then we have

$$\mathrm{supp}(\rho) \subseteq A \quad \mathrm{Tr}_2(\rho) = \rho_1 \quad \mathrm{Tr}_1(\rho) = \rho_2$$

Let $\sigma = (U_1 \otimes U_2)\rho(U_1^\dagger \otimes U_2^\dagger)$ and $B = (U_1 \otimes U_2)A(U_1^\dagger \otimes U_2^\dagger)$, then it is direct to check that σ is a witness of the lifting $U_1\rho_1U_1^\dagger \sim_B U_2\rho_2U_2^\dagger$,

$$\mathrm{supp}(\sigma) = \mathrm{supp}((U_1 \otimes U_2)\rho(U_1^\dagger \otimes U_2^\dagger)) \subseteq (U_1 \otimes U_2)A(U_1^\dagger \otimes U_2^\dagger) = B$$
$$\mathrm{Tr}_2(\sigma) = U_1\rho_1U_1^\dagger \quad \mathrm{Tr}_1(\sigma) = U_2\rho_2U_2^\dagger$$

Thus the lifting $U_1\rho_1U_1 \sim_B U_2\rho_2U_2^\dagger$ holds, we have

$$\forall \rho_1, \rho_2.\, \rho_1 \sim_A \rho_2 \;\Rightarrow\; U_1\rho_1U_1^\dagger \sim_B U_2\rho_2U_2^\dagger$$

i.e. $\vDash \bar{q}_1 := U_1[\bar{q}_1] \sim \bar{q}_2 := U_2[\bar{q}_2] : A \Rightarrow (U_1 \otimes U_2)A(U_1^\dagger \otimes U_2^\dagger)$ by the semantics.

(3) (Qo). Here we first prove the rule for any two superoperators $\mathcal{E}_1$ and $\mathcal{E}_2$,

$$\vDash \bar{q}_1 := \mathcal{E}_1[\bar{q}_1] \sim \bar{q}_2 := \mathcal{E}_2[\bar{q}_2] : A \Rightarrow \mathrm{proj}((\mathcal{E}_1 \otimes \mathcal{E}_2)(A))$$

For any ρ_1 and ρ_2 such that there exist a witness ρ of the lifting $\rho_1 \sim_A \rho_2$, then we have

$$\mathrm{supp}(\rho) \subseteq A \quad \mathrm{Tr}_2(\rho) = \rho_1 \quad \mathrm{Tr}_1(\rho) = \rho_2$$

Let $\sigma = (\mathcal{E}_1 \otimes \mathcal{E}_2)(\rho)$ and $B = \text{proj}((\mathcal{E}_1 \otimes \mathcal{E}_2)(A))$, then it is direct to check that σ is a witness of lifting $\mathcal{E}_1(\rho_1) \sim_B \mathcal{E}_2(\rho_2)$,

$$\text{supp}(\sigma) = \text{supp}((\mathcal{E}_1 \otimes \mathcal{E}_2)(\rho)) \subseteq \text{proj}((\mathcal{E}_1 \otimes \mathcal{E}_2)(A)) = B$$
$$\text{Tr}_2(\sigma) = \mathcal{E}_1(\text{Tr}_2(\rho)) = \mathcal{E}_1(\rho_1) \quad \text{Tr}_1(\sigma) = \mathcal{E}_2(\text{Tr}_1(\rho)) = \mathcal{E}_2(\rho_2)$$

Thus the lifting $\mathcal{E}_1(\rho_1) \sim_B \mathcal{E}_2(\rho_2)$ holds, we have

$$\forall \rho_1, \rho_2 . \rho_1 \sim_A \rho_2 \implies \mathcal{E}_1(\rho_1) \sim_B \mathcal{E}_2(\rho_2)$$

i.e. $\vDash \bar{q}_1 := \mathcal{E}_1[\bar{q}_1] \sim \bar{q}_2 := \mathcal{E}_2[\bar{q}_2] : A \Rightarrow \text{proj}((\mathcal{E}_1 \otimes \mathcal{E}_2)(A))$.

(4) (Init). The Init rule is an instance of the above rule with $\mathcal{E}_1 = \sum_n |0\rangle_{\bar{q}_1}\langle n|\rho|n\rangle_{\bar{q}_1}\langle 0|$ and $\mathcal{E}_2 = \sum_n |0\rangle_{\bar{q}_2}\langle n|\rho|n\rangle_{\bar{q}_2}\langle 0|$.

(5) (Seq) By the first assumption $\vdash S_1 \sim_{\delta_1} S_2 : A \Rightarrow R$, we have

$$\forall \rho_1, \rho_2 . \rho_1 \sim_A \rho_2 \implies [\![S_1]\!](\rho_1) \sim_R^{\delta_1} [\![S_2]\!](\rho_2)$$

then there exits a witness ρ of the lifting $[\![S_1]\!](\rho_1) \sim_R^{\delta_1} [\![S_2]\!](\rho_2)$. Given the second assumption $\vdash S_1' \sim_{\delta_2} S_2' : R \Rightarrow B$, we apply Lemma 4.19 to have

$$[\![S_1]\!](\rho_1) \sim_R^{\delta_1} [\![S_2]\!](\rho_2) \quad \Rightarrow \quad [\![S_1']\!]([\![S_1]\!](\rho_1)) \sim_B^{\delta_1+\delta_2} [\![S_2']\!]([\![S_2]\!])(\rho_2)$$

Combine the first assumption and the semantics of sequence, we have

$$\forall \rho_1, \rho_2 . \rho_1 \sim_A \rho_2 \implies [\![S_1; S_1']\!](\rho_1) \sim_B^{\delta_1+\delta_2} [\![S_2; S_2']\!](\rho_2)$$

i.e. $\vDash S_1; S_1' \sim_{\delta_1+\delta_2} S_2; S_2' : A \Rightarrow B$.

(6) (IF). Let $Q_1 = \textbf{if } (\square m \cdot \mathcal{M}_1[\bar{q}] = m \rightarrow S_{1m}) \textbf{ fi}$, $Q_2 = \textbf{if } (\square m \cdot \mathcal{M}_2[\bar{q}] = m \rightarrow S_{2m}) \textbf{ fi}$. By the assumption $\mathcal{M}_1 \approx_{\{\delta_m\}} \mathcal{M}_2 : A \Rightarrow \{(p_m, B_m)\}$, we have

$$\forall \rho_1, \rho_2 . \rho_1 \sim_A \rho_2 \implies \sigma_{1m} \sim_{B_m}^{\delta_m} \sigma_{2m}$$

for all m, where $\sigma_{1m} = M_1^m \rho_1 M_1^{m\dagger}$, $\sigma_{2m} = M_2^m \rho_2 M_2^{m\dagger}$, $\mathrm{Tr}(\sigma_{1m}) \le p_m$ and $\mathrm{Tr}(\sigma_{2m}) \le p_m$. Together with the assumption $\vdash S_{1m} \sim_{\delta'_m} S_{2m} : B_m \Rightarrow C$, we have

$$\begin{aligned}
&\forall \rho_1, \rho_2 .\, \rho_1 \sim_A \rho_2 \\
&\Rightarrow\ \sigma_{1m} \sim_{B_m}^{\delta_m} \sigma_{2m} \\
&\Rightarrow\ \sigma_{1m}/p_m \sim_{B_m}^{\delta_m/p_m} \sigma_{2m}/p_m \qquad \text{(Scalability in 4.8)} \\
&\Rightarrow\ [\![S_{1m}]\!](\sigma_{1m}/p_m) \sim_C^{\delta_m/p_m+\delta'_m} [\![S_{2m}]\!](\sigma_{2m}/p_m) \qquad \text{(Lemma 4.19)} \\
&\Rightarrow\ [\![S_{1m}]\!](\sigma_{1m}) \sim_C^{\delta_m+p_m\delta'_m} [\![S_{2m}]\!](\sigma_{2m}) \qquad \text{(Scalability in 4.8)} \\
&\Rightarrow\ \textstyle\sum_m [\![S_{1m}]\!](\sigma_{1m}) \sim_C^{\sum_m \delta_m+p_m\delta'_m} \sum_m [\![S_{2m}]\!](\sigma_{2m}) \qquad \text{(Linearity in 4.8)} \\
&\Rightarrow\ [\![Q_1]\!](\rho_1) \sim_C^{\sum_m \delta_m+p_m\delta'_m} [\![Q_2]\!](\rho_2) \quad \text{(Fig. 2.2)}
\end{aligned}$$

for all m. Thus we have $\vDash Q_1 \sim_{\sum_m \delta_m+p_m\delta'_m} Q_2 : A \Rightarrow C$.

(7) (LP). Let $Q_i = \textbf{while}\ \mathcal{M}_i[\bar{q}] = 1\ \textbf{do}\ S_i\ \textbf{od}$, $\mathcal{E}_i(\rho) = [\![S_i]\!](M_i^1 \rho M_i^{1\dagger})$ for $i \in \{1,2\}$. We combine $\mathcal{M}_1 \approx_{\{\delta_0,\delta_1\}} \mathcal{M}_2 : A \Rightarrow \{(p_0,B_0),(p_1,B_1)\}$ and $\vdash S_1 \sim_\delta S_2 : B_1 \Rightarrow A$ to have

$$\forall \rho_1, \rho_2 .\, \rho_1 \sim_A \rho_2 \ \Rightarrow\ M_1^0 \rho_1 M_1^{0\dagger} \sim_{B_0}^{\delta_0} M_2^0 \rho_2 M_2^{0\dagger}$$
$$\forall \rho_1, \rho_2 .\, \rho_1 \sim_A \rho_2 \ \Rightarrow\ M_1^1 \rho_1 M_1^{1\dagger} \sim_{B_1}^{\delta_1} M_2^1 \rho_2 M_2^{1\dagger} \Rightarrow \mathcal{E}_1(\rho_1) \sim_A^{\delta_1+p_1\delta} \mathcal{E}_2(\rho_2)$$

with $\mathrm{Tr}(\mathcal{E}_i(\rho_i)) \le \mathrm{Tr}(M_i^1 \rho_i M_i^{1\dagger}) \le p_1$ for $i \in \{1,2\}$. Now we prove our rule by induction. Let Q_i^k be the loop Q_i must terminate at most k iterations, $[\![Q_i^0]\!](\rho) = M_i^0 \rho M_i^{0\dagger}$, then we have

$$[\![Q_i^{k+1}]\!](\rho) = M_i^0 \rho M_i^{0\dagger} + [\![Q_i^k]\!](\mathcal{E}_i(\rho))$$

by the semantics of the loop statement in Fig. 2.2. Assume we have $\vDash Q_1^k \sim_{d_k} Q_2^k : A \Rightarrow B_0$ for $k \in \mathbb{N}$. For $k = 0$, it is straightforward to check $d_0 = \delta_0$ by the measurement condition. Then we have

$$\forall \rho_1, \rho_2 .\, \rho_1 \sim_A \rho_2 \ \Rightarrow\ \mathcal{E}_1(\rho_1) \sim_A^{\delta_1+p_1\delta} \mathcal{E}_2(\rho_2)$$

$$\Rightarrow \llbracket Q_1^k \rrbracket(\mathcal{E}_1(\rho_1)) \sim_{B_0}^{\delta_1+p_1(\delta+d_k)} \llbracket Q_2^k \rrbracket(\mathcal{E}_2(\rho_2)) \quad (\vdash Q_1^k \sim_{d_k} Q_2^k : A \Rightarrow B_0)$$
$$\Rightarrow \llbracket Q_1^{k+1} \rrbracket(\rho_1) \sim_{B_0}^{\delta_0+\delta_1+p_1(\delta+d_k)} \llbracket Q_2^{k+1} \rrbracket(\rho_2) \quad \text{(Linearity in 4.8)}$$

Therefore, we have $d_{k+1} = \delta_0+\delta_1+p_1(\delta+d_k) = \sum_{i=0}^{k+1} p_1^i\delta_0 + \sum_{i=0}^{k} p_1^i\delta_1 + \sum_{i=1}^{k+1} p_1^i\delta$. Since $\lim_{k\to\infty} Q_i^k$ converges to Q_i, thus we have

$$\forall \rho_1, \rho_2.\, \rho_1 \sim_A \rho_2 \;\Rightarrow\Rightarrow\; \llbracket Q_1 \rrbracket(\rho_1) \sim_{B_0}^{\Delta} \llbracket Q_2 \rrbracket(\rho_2)$$

where $\lim_{k\to\infty} d_k \le \frac{1}{1-p_1}(\delta_0+\delta_1) + \frac{p_1}{1-p_1}\delta = \Delta$. Thus we have $\vDash Q_1 \sim_\Delta Q_2 : A \Rightarrow B_0$.

(8) (LP*). Let $Q_i = \textbf{while } \mathcal{M}_i[\bar{q}] = 1 \textbf{ do } S_i \textbf{ od}$, $\mathcal{E}_i(\rho) = \llbracket S_i \rrbracket(M_i^1 \rho M_i^{1\dagger})$, $\llbracket Q_i^k \rrbracket(\rho) = M_i^0 \mathcal{E}_i^k(\rho) M_i^{0\dagger}$ for $i \in \{1,2\}$, then we have $Q_i = \sum_{k=0}^{\infty} Q_i^k$. By the measurement condition

$$\mathcal{M}_1 \approx_{\{\alpha_k,\beta_k\}} \mathcal{M}_2 : A_k \Rightarrow \{(q_k, C), (p_k, B_k)\}$$

we have

$$\forall \rho_1,\ \rho_2.\, \rho_1 \sim_{A_k} \rho_2 \;\Rightarrow\; \begin{cases} M_1^0\rho_1 M_1^{0\dagger} \sim_C^{\alpha_k} M_2^0 \rho_1 M_2^{0\dagger} & M_1^1\rho_2 M_1^{1\dagger} \sim_{B_k}^{\beta_k} M_2^1 \rho_2 M_2^{1\dagger} \\ \mathrm{Tr}(M_i^0\rho_1 M_i^{0\dagger}) \le q_k & \mathrm{Tr}(M_i^1 \rho_2 M_i^{1\dagger}) \le p_k \end{cases}$$

for $0 \le k < N$. Together with the assumption $\vdash S_1 \sim_{\delta_k} S_2 : B_k \Rightarrow A_{k+1}$, we have

$$\begin{aligned} \forall \rho_1,\rho_2.\, \rho_1 \sim_{A_0} \rho_2 \;&\Rightarrow\; M_1^1\rho_1 M_1^{1\dagger} \sim_{B_0}^{\beta_0} M_2^1 \rho_2 M_2^{1\dagger} \\ &\Rightarrow\; M_1^1\rho_1 M_1^{1\dagger}/p_0 \sim_{B_1}^{\beta_0/p_0} M_2^1 \rho_2 M_2^{1\dagger}/p_0 \quad \text{(Scalability in 4.8)} \\ &\Rightarrow\; \mathcal{E}_1(\rho_1/p_1) \sim_{A_1}^{\beta_0/p_0+\delta_0} \mathcal{E}_2(\rho_2/p_1) \quad \text{(Lemma 4.19)} \\ &\Rightarrow\; \mathcal{E}_1(\rho_1) \sim_{A_1}^{\beta_0+p_0\delta_0} \mathcal{E}_2(\rho_2) \quad \text{(Scalability in 4.8)} \end{aligned}$$

where $\mathrm{Tr}(\mathcal{E}_i(\rho_i)) \le \mathrm{Tr}(\rho_i)\cdot p_0$. Similarly, we also have

$$\forall \rho_1,\rho_2.\, \rho_1 \sim_{A_k} \rho_2 \;\Rightarrow\; \mathcal{E}_1(\rho_1) \sim_{A_{k+1}}^{\beta_k+p_k\delta_k} \mathcal{E}_2(\rho_2)$$

where $\mathrm{Tr}(\mathcal{E}_i(\rho_i)) \le \mathrm{Tr}(\rho_i)\cdot p_k$. By Lemma 4.18, we combine the above equations to have

$$\forall \rho_1,\rho_2.\, \rho_1 \sim_{A_0} \rho_2 \;\Rightarrow\; \mathcal{E}_1^n(\rho_1) \sim_{A_n}^{(\beta_0+\sum_{i=0}^{n-2}\lambda_i\beta_{i+1})+(\sum_{i=0}^{n-1}\lambda_i\delta_i)} \mathcal{E}_2^n(\rho_2)$$

$$\Rightarrow\ [\![Q_1^n]\!](\rho_1) \sim_C^{d_n} [\![Q_2^n]\!](\rho_2)$$

for $2 \le n \le N$, where $\lambda_n = \prod_{k=0}^{n} p_k$, $d_n = \lambda_{n-1}\alpha_n + (\beta_0 + \sum_{i=0}^{n-2}\lambda_i\beta_{i+1}) + (\sum_{i=0}^{n-1}\lambda_i\delta_i)$. Particularly, we have $d_0 = \alpha_0$ and $d_1 = p_0\alpha_0 + \beta_0 + p_0\delta_0$. By another measurement condition $\mathcal{M}_1 \approx_{\{\alpha_N,0\}} \mathcal{M}_2 : A_N \Rightarrow \{(1,C),(0,I)\}$, the loops would terminate at most N iterations, that is, $Q_i = \sum_{k=0}^{N}[\![Q_i^k]\!]$. Next, we have

$$\begin{aligned}
\forall\rho_1,\rho_2.\ \rho_1 \sim_A \rho_2 \ &\Rightarrow\ [\![Q_1^n]\!](\rho_1) \sim_C^{d_n} [\![Q_2^n]\!](\rho_2)\\
&\Rightarrow\ \textstyle\sum_{n=0}^{N}[\![Q_1^n]\!](\rho_1) \sim_C^{\sum_{n=0}^{N} d_n} \sum_{n=0}^{N}[\![Q_2^n]\!](\rho_2) \qquad \text{(Linearity in 4.8)}\\
&\Rightarrow\ [\![Q_1]\!](\rho_1) \sim_C^{f(\alpha_k,\beta_k,p_k)} [\![Q_2]\!](\rho_2)
\end{aligned}$$

where we $f(\alpha_k,\beta_k,p_k) = \sum_{i=0}^{N} d_i = (\alpha_0 + \sum_{n=0}^{N-1}\lambda_n\alpha_{n+1}) + (N\beta_0 + \sum_{n=0}^{N-2}(N-n-1)\lambda_n\beta_{n+1}) + (\sum_{n=0}^{N-1}(N-n)\lambda_n\delta_n)$. Thus, we have $\vDash Q_1 \sim_{f(\alpha_k,\beta_k,p_k)} Q_2 : A_0 \Rightarrow C$.

(9) (Init-L) A instance of rule (Init) with $\sum_n |0\rangle_{q_2}\langle n|\rho|n\rangle_{q_2}\langle 0|$ on q_2 being I.

(10) (Ut-L) A instance of rule (Ut) with $U_2 = I$.

(11) (If-L) Let $Q = \textbf{if}\ (\square m \cdot \mathcal{M}_1[\bar{q}] = m \rightarrow S_{1m})\ \textbf{fi}$. By the assumption $\mathcal{M}_1 \approx I_2 : A \Rightarrow \{(p_m, B_m)\}$, we have

$$\forall\rho_1,\rho_2.\ \rho_1 \sim_A \rho_2 \ \Rightarrow\ \sigma_{1m} \sim_{B_m} \sigma_{2m}$$

for all m, where $\sigma_{1m} = M_1^m\rho_1 M_1^{m\dagger}$, $\sum_m \sigma_{2m} = \rho_2$, $\mathrm{Tr}(\sigma_{1m}) \le p_m$ and $\mathrm{Tr}(\sigma_{2m}) \le p_m$. Together with the assumption $\vdash S_{1m} \sim_{\delta_m} \textbf{skip} : B_m \Rightarrow C$, we have

$$\begin{aligned}
&\forall\rho_1,\rho_2.\ \rho_1 \sim_A \rho_2\\
&\Rightarrow\ \sigma_{1m} \sim_{B_m} \sigma_{2m}\\
&\Rightarrow\ \sigma_{1m}/p_m \sim_{B_m} \sigma_{2m}/p_m \qquad \text{(Scalability in 4.8)}\\
&\Rightarrow\ [\![S_{1m}]\!](\sigma_{1m}/p_m) \sim_C^{\delta_m} \sigma_{2m}/p_m \qquad \text{(Lemma 4.19)}\\
&\Rightarrow\ [\![S_{1m}]\!](\sigma_{1m}) \sim_C^{p_m\delta_m} \sigma_{2m} \qquad \text{(Scalability in 4.8)}\\
&\Rightarrow\ \textstyle\sum_m[\![S_{1m}]\!](\sigma_{1m}) \sim_C^{\sum_m p_m\delta_m} \rho_2 \qquad \text{(Linearity in 4.8)}
\end{aligned}$$

$$\Rightarrow \ [\![Q]\!](\rho_1) \sim_C^{\sum_m p_m\delta_m} [\![\mathbf{skip}]\!](\rho_2) \quad \text{(Fig. 2.2)}$$

for all m. Thus we have $\vDash Q \sim_{\sum_m p_m\delta_m} \mathbf{skip} : A \Rightarrow C$.

(12) (Lp-L) Let $Q = \mathbf{while}\ \mathcal{M}_1[\bar{q}] = 1\ \mathbf{do}\ S_1\ \mathbf{od}$, $\mathcal{E}(\rho) = [\![S_1]\!](M_1^1\rho M_1^{1\dagger})$. We combine $\mathcal{M}_1 \approx I_2 : A \Rightarrow \{(p_0, B_0), (p_1, B_1)\}$ and $\vdash S_1 \sim_\delta \mathbf{skip} : B_1 \Rightarrow A$ to have

$$\begin{aligned} &\forall \rho_1, \rho_2.\, \rho_1 \sim_A \rho_2 \ \Rightarrow\ M_1^0\rho_1 M_1^{0\dagger} \sim_{B_0} \sigma_0 \\ &\forall \rho_1, \rho_2.\, \rho_1 \sim_A \rho_2 \ \Rightarrow\ M_1^1\rho_1 M_1^{1\dagger} \sim_{B_1} \sigma_1 \ \Rightarrow\ \mathcal{E}(\rho_1) \sim_A^{p_1\delta} \sigma_1 \end{aligned}$$

with $\sigma_0 + \sigma_1 = \rho_2$, $\mathrm{Tr}(M_1^i\rho_1 M_1^{i\dagger}) \le p_i$, $\mathrm{Tr}(\sigma_i) \le p_i$ for $i \in \{0, 1\}$. Now we prove our rule by induction. Let Q^k be the loop Q must terminate at most k iterations, and P^k be the corresponding statement that shares the same probability distribution with program Q^k. That is,

$$\begin{aligned} [\![Q^{k+1}]\!](\rho_1) &= M_1^0\rho_1 M_1^{0\dagger} + [\![Q^k]\!](\mathcal{E}(\rho_1)) \\ [\![P^{k+1}]\!](\rho_2) &= \sigma_0 + [\![P^k]\!](\sigma_1) \end{aligned}$$

where $[\![Q^0]\!](\rho) = M_1^0\rho M_1^{0\dagger}$, $[\![P^0]\!](\rho_2) = \sigma_0$, $\lim_{k\to\infty} Q^k$ converges to Q, Since $\lim_{k\to\infty} P^k$ converges to **skip**.

Assume we have $\vDash Q^k \sim_{d_k} P^k : A \Rightarrow B_0$ for $k \in \mathbb{N}$. For $k = 0$, it is straightforward to check $d_0 = 0$ by the measurement condition. Then we have

$$\begin{aligned} &\forall \rho_1, \rho_2.\, \rho_1 \sim_A \rho_2 \ \Rightarrow\ \mathcal{E}(\rho_1) \sim_A^{p_1\delta} \sigma_1 \\ &\Rightarrow\ [\![Q^k]\!](\mathcal{E}(\rho_1)) \sim_{B_0}^{p_1(\delta+d_k)} [\![P^k]\!](\sigma_1) \quad (\vdash Q^k \sim_{d_k} P^k : A \Rightarrow B_0) \\ &\Rightarrow\ [\![Q^{k+1}]\!](\rho_1) \sim_{B_0}^{p_1(\delta+d_k)} [\![P^{k+1}]\!](\rho_2) \quad \text{(Linearity in 4.8)} \end{aligned}$$

Therefore, we have $d_{k+1} = p_1(\delta + d_k) = \sum_{i=1}^{k+1} p_1^i\delta$. Thus we have

$$\forall \rho_1, \rho_2.\, \rho_1 \sim_A \rho_2 \Rightarrow [\![Q]\!](\rho_1) \sim_{B_0}^{\Delta} [\![\mathbf{skip}]\!](\rho_2)$$

where $\lim_{k\to\infty} d_k \le \frac{p_1}{1-p_1}\delta = \Delta$. Thus we have $\vDash Q \sim_\Delta \mathbf{skip} : A \Rightarrow B_0$.

(13) (Lp*-L) Let $Q = \textbf{while } \mathcal{M}_1[\bar{q}] = 1 \textbf{ do } S_1 \textbf{ od}$, $\mathcal{E}(\rho) = [\![S_1]\!](M_1^1 \rho M_1^{1\dagger})$, $[\![Q^k]\!](\rho) = M_1^0 \mathcal{E}^k(\rho) M_1^{0\dagger}$, then we have $Q = \sum_{k=0}^{\infty} Q^k$. By the measurement condition

$$\mathcal{M}_1 \approx I_2 : A_k \Rightarrow \{(q_k, C), (p_k, B_k)\}$$

we have

$$\forall \rho_1,\ \rho_2 .\, \rho_1 \sim_{A_k} \rho_2 \ \Rightarrow \begin{cases} M_1^0 \rho_1 M_1^{0\dagger} \sim_C \sigma_{0,k} & M_1^1 \rho_1 M_1^{1\dagger} \sim_{B_k} \sigma_{1,k} \\ \mathrm{Tr}(M_1^0 \rho_1 M_1^{0\dagger}) \le q_k & \mathrm{Tr}(M_1^1 \rho_2 M_1^{1\dagger}) \le p_k \end{cases}$$

for $0 \le k < N$, $\sigma_{0,k} + \sigma_{1,k} = \rho_2$. Together with the assumption $\vdash S_1 \sim_{\delta_k} \textbf{skip} : B_k \Rightarrow A_{k+1}$, we have

$$\forall \rho_1, \rho_2 .\, \rho_1 \sim_{A_k} \rho_2 \ \Rightarrow \ \mathcal{E}(\rho_1) \sim_{A_{k+1}}^{p_k \delta_k} \sigma_{1,k}$$

where $\mathrm{Tr}(\mathcal{E}(\rho_1)) \le \mathrm{Tr}(\rho_1) \cdot p_k$. By Lemma 4.18, we combine the above equations to have

$$\forall \rho_1, \rho_2 .\, \rho_1 \sim_{A_0} \rho_2 \ \Rightarrow \ \mathcal{E}^n(\rho_1) \sim_{A_n}^{d_n} \sigma_{1,n-1} \ \Rightarrow \ [\![Q_1^n]\!](\rho_1) \sim_C^{d_n} \sigma_{0,n}$$

for $1 \le n \le N-1$, where $\sigma_{1,n-1} = \sigma_{0,n} + \sigma_{1,n}$, $\lambda_n = \prod_{k=0}^{n} p_k$, $d_n = \sum_{i=0}^{n-1} \lambda_i \delta_i$. Particularly, we have $d_0 = 0$. By another measurement condition $\mathcal{M}_1 \approx_{\{\alpha_N, 0\}} \mathcal{M}_2 : A_N \Rightarrow \{(1, C), (0, I)\}$, the loops would terminate at most N iterations, that is, $Q_i = \sum_{k=0}^{N} [\![Q_i^k]\!]$. Next, we have

$$\begin{aligned} \forall \rho_1, \rho_2 .\, \rho_1 \sim_{A_0} \rho_2 \ &\Rightarrow \ [\![Q_1^n]\!](\rho_1) \sim_C^{d_n} \sigma_{0,n} \\ &\Rightarrow \ \textstyle\sum_{n=0}^{N} [\![Q_1^n]\!](\rho_1) \sim_C^{\Delta} \sum_{n=0}^{N} \sigma_{0,n} \quad \text{(Linearity in 4.8)} \\ &\Rightarrow \ [\![Q_1]\!](\rho_1) \sim_C^{\Delta} \textbf{skip} \end{aligned}$$

where $\sigma_{1,N-1} = \sigma_{0,N}$, $\sum_{n=0}^{N} \sigma_{0,n} = \rho_2$, $\Delta = \lambda_{N-1} \alpha_N + \sum_{n=0}^{N} d_n$. Thus, we have $\vDash Q_1 \sim_{\Delta} \textbf{skip} : A_0 \Rightarrow C$.

(14) (Order). By the assumptions $\vdash S_1 \sim_{\delta'} S_2 : A' \Rightarrow B'$, $A \le A'$, $B' \le B$, $\delta' \le \delta$, we have

$$\forall \rho .\, \rho \vDash A \Rightarrow \ \rho \vDash A' \quad \text{((Monotonicity in 4.8))}$$

$$\Rightarrow \llbracket S_1 \rrbracket(\mathrm{Tr}_2(\rho)) \sim_{B'}^{\delta'} \llbracket S_2 \rrbracket(\mathrm{Tr}_1(\rho)) \quad (\vdash S_1 \sim_{\delta'} S_2 : A' \Rightarrow B')$$

$$\Rightarrow \llbracket S_1 \rrbracket(\mathrm{Tr}_2(\rho)) \sim_{B}^{\delta} \llbracket S_2 \rrbracket(\mathrm{Tr}_1(\rho)) \quad ((\text{Monotonicity in } 4.8))$$

i.e. $\Gamma \vDash S_1 \sim_\delta S_2 : A \Rightarrow B$.

(15) (Approx). It is direct from Def. 4.12. By the assumptions $\vdash S_1 \sim S_2 : A \Rightarrow B$,

$$\forall \rho_1, \rho_2.\, \rho_1 \sim_A \rho_2 \;\Rightarrow\; \llbracket S_1 \rrbracket(\rho_1) \sim_B \llbracket S_2 \rrbracket(\rho_2) \Rightarrow \llbracket S_1 \rrbracket(\rho_1) \sim_C^{\delta} \llbracket S_2 \rrbracket(\rho_2)$$

Thus we have $\vdash S_1 \sim_\delta S_2 : A \Rightarrow C$.

(16) (Frame). Let $C_{(\bar r_1, \bar r_2)} = |\psi\rangle\langle\psi|$. Next, we have

$$\begin{aligned}
&\forall \rho_1', \rho_2'.\, \rho_1' \sim_{A \otimes C_{(\bar r_1, \bar r_2)}} \rho_2' \\
&\Rightarrow \rho' \vDash A \otimes C_{(\bar r_1, \bar r_2)}, \quad \rho_1' = \mathrm{Tr}_{\bar r_1, \bar q_2, \bar r_2}\, \rho' \quad \rho_2' = \mathrm{Tr}_{\bar r_1, \bar q_1, \bar r_2}\, \rho' \\
&\Rightarrow \rho' = \rho \otimes |\psi\rangle\langle\psi|\,, \rho \vDash A \\
&\Rightarrow \rho_1' = \mathrm{Tr}_{\bar q_2}\, \rho \otimes \mathrm{Tr}_{\bar r_2} |\psi\rangle\langle\psi| \quad \rho_2' = \mathrm{Tr}_{\bar q_1}\, \rho \otimes \mathrm{Tr}_{\bar r_1} |\psi\rangle\langle\psi| \\
&\Rightarrow \mathrm{Tr}_{\bar q_2}\, \rho \sim_A \mathrm{Tr}_{\bar q_1}\, \rho, \quad \mathrm{Tr}_{\bar r_2} |\psi\rangle\langle\psi| \sim_{|\psi\rangle\langle\psi|} \mathrm{Tr}_{\bar r_1} |\psi\rangle\langle\psi| \\
&\Rightarrow \llbracket S_1 \rrbracket(\mathrm{Tr}_{\bar q_2}\, \rho) \sim_B^{\delta} \llbracket S_2 \rrbracket(\mathrm{Tr}_{\bar q_1}\, \rho) \\
&\Rightarrow \llbracket S_1 \rrbracket(\mathrm{Tr}_{\bar q_2}\, \rho \otimes \mathrm{Tr}_{\bar r_2} |\psi\rangle\langle\psi|) \sim_{B \otimes C_{(\bar r_1, \bar r_2)}}^{\delta} \llbracket S_2 \rrbracket(\mathrm{Tr}_{\bar q_1}\, \rho \otimes \mathrm{Tr}_{\bar r_1} |\psi\rangle\langle\psi|) \\
&\Rightarrow \llbracket S_1 \rrbracket(\rho_1') \sim_{B \otimes C_{(\bar r_1, \bar r_2)}}^{\delta} \llbracket S_2 \rrbracket(\rho_2')
\end{aligned}$$

where in the fifth $\Rightarrow$, we use the assumption $\vdash_{(\bar q_1, \bar q_2)} S_1 \sim_\delta S_2 : A \Rightarrow B$, we have

$$\forall \rho_1', \rho_2'.\, \rho_1 \sim_{A \otimes C_{(\bar r_1, \bar r_2)}} \rho_2 \;\Rightarrow\; \llbracket S_1 \rrbracket(\rho_1') \sim_{B \otimes C_{(\bar r_1, \bar r_2)}}^{\delta} \llbracket S_2 \rrbracket(\rho_2')$$

Thus, we have $\vDash_{(\bar q_1, \bar r_1), (\bar q_2, \bar r_2)} S_1 \sim_\delta S_2 : A \otimes C_{(\bar r_1, \bar r_2)} \Rightarrow B \otimes C_{(\bar r_1, \bar r_2)}$.

(17) (Ut-id). For any ρ_1, ρ_2 such that $\rho_1 \sim_\equiv \rho_2$, we have $\rho_1 = \rho_2 = \rho$ by lemma 4.10. Let $\sigma = (U_1 \rho U_1^\dagger + U_2 \rho U_2^\dagger)/2$, the eigen-decomposition of σ is $\sigma = \sum_i \lambda_i |\psi\rangle\langle\psi|$. We need to check $\gamma = \sum_i \lambda_i |\psi\rangle\langle\psi| \otimes |\psi\rangle\langle\psi|$ is a witness of the lifting $U_1 \rho U_1^\dagger \sim_\equiv^{\|U_1 \cdot U_1^\dagger - U_2 \cdot U_2^\dagger\|_\diamond / 2} U_2 \rho U_2^\dagger$. It is direct to check $\mathrm{supp}(\gamma) \subseteq \equiv$. Besides,

$$\mathrm{Tr}(\gamma) = \mathrm{Tr}(\sigma) = (\mathrm{Tr}(U_1 \rho U_1^\dagger) + \mathrm{Tr}(U_2 \rho U_2^\dagger))/2 = \mathrm{Tr}(\rho) = \mathrm{Tr}(U_1 \rho U_1^\dagger) = \mathrm{Tr}(U_2 \rho U_2^\dagger)$$

And we also have

$$D(U_1\rho U_1^\dagger, \mathrm{Tr}_2(\gamma)) = D(U_1\rho U_1^\dagger, \sigma) = \frac{1}{2}D(U_1\rho U_1^\dagger, U_2\rho U_2^\dagger)$$

By Def. 4.3 of diamond norm we have

$$D(U_1\rho U_1^\dagger, U_2\rho U_2^\dagger) \leq \|U_1 \cdot U_1^\dagger - U_2 \cdot U_2^\dagger\|_\diamond$$

for any ρ. Thus, we have $D(U_1\rho U_1^\dagger, \mathrm{Tr}_2(\gamma)) \leq \|U_1 \cdot U_1^\dagger - U_2 \cdot U_2^\dagger\|_\diamond/2$, and the same goes for $D(U_2\rho U_2^\dagger, \mathrm{Tr}_1(\gamma)) \leq \|U_1 \cdot U_1^\dagger - U_2 \cdot U_2^\dagger\|_\diamond/2$. Thus we have $\vDash \bar{q}_1 := U_1[\bar{q}_1] \sim_{\|U_1\cdot U_1^\dagger - U_2\cdot U_2^\dagger\|_\diamond/2} \bar{q}_2 := U_2[\bar{q}_2] : \equiv \Rightarrow \equiv$.

(18) (Comp). This rule is direct from lemma 4.10 and the triangle inequality of trace distance. Given $\vdash S_1 \sim_{\delta_1} S_2 : \equiv \Rightarrow \equiv$ and $\vdash S_2 \sim_{\delta_2} S_3 : \equiv \Rightarrow \equiv$, we have

$$\forall \rho_1, \rho_2 .\, \rho_1 \equiv \rho_2 \;\Rightarrow\; [\![S_1]\!](\rho_1) \sim_{\equiv}^{\delta_1} [\![S_2]\!](\rho_2)$$
$$\forall \rho_2, \rho_3 .\, \rho_2 \equiv \rho_3 \;\Rightarrow\; [\![S_2]\!](\rho_2) \sim_{\equiv}^{\delta_2} [\![S_3]\!](\rho_3)$$

By lemma 4.10, we have

$$\rho_1 = \rho_2 = \rho_3 = \rho$$
$$D([\![S_1]\!](\rho_1), [\![S_2]\!](\rho_2)) \leq 2\delta_1$$
$$D([\![S_2]\!](\rho_2), [\![S_3]\!](\rho_3)) \leq 2\delta_2$$

Thus we have

$$\begin{aligned} D([\![S_1]\!](\rho_1), [\![S_3]\!](\rho_3)) &\leq D([\![S_1]\!](\rho_1), [\![S_2]\!](\rho_2)) + D([\![S_2]\!](\rho_2), [\![S_3]\!](\rho_3)) \\ &= 2(\delta_1 + \delta_2) \end{aligned}$$

Then we apply lemma 4.10 again to have

$$\forall \rho_1, \rho_3 .\, \rho_1 \equiv \rho_3 \;\Rightarrow\; [\![S_1]\!](\rho_1) \sim_{\equiv}^{\delta_1+\delta_2} [\![S_3]\!](\rho_3)$$

Thus we have $\vdash S_1 \sim_{\delta_1+\delta_2} S_3 : \equiv \Rightarrow \equiv$.

■

Lemma 4.17. *Let S be a quantum program defined in Fig. 2.2, ρ and σ are arbitrary partial density matrices.*

(1) $\mathrm{Tr}(\llbracket S\rrbracket(\rho)) \le \mathrm{Tr}(\rho)$. *The equality holds when S could terminate.*

(2) $D(\llbracket S\rrbracket(\rho), \llbracket S\rrbracket(\sigma)) \le D(\rho, \sigma)$.

Proof. Notice that S can always be represented by a non-trace-increasing superoperator $\mathcal{E}$, and these proprieties directly hold for $\mathcal{E}$. ■

Lemma 4.18. *If $\vDash S_1 \sim_\delta S_2 : A \Rightarrow B$, then we have*

$$\forall \rho \vDash A.\, \mathrm{Tr}(\rho) \le \alpha \;\Rightarrow\; \llbracket S_1\rrbracket(\mathrm{Tr}_2(\rho)) \sim_B^{\alpha\delta} \llbracket S_2\rrbracket(\mathrm{Tr}_1(\rho))$$

for any $0 \le \alpha \le 1$.

Proof. It is direct from the scalability 4.8 of quantum approximate liftings. ■

Lemma 4.19. *If $\vDash S_1 \sim_\delta S_2 : A \Rightarrow B$, then we have*

$$\forall \rho_1, \rho_2.\, \rho_1 \sim_A^{\delta'} \rho_2 \;\Rightarrow\; \llbracket S_1\rrbracket(\rho_1) \sim_B^{\delta+\delta'} \llbracket S_2\rrbracket(\rho_2)$$

for any $0 \le \delta'$.

Proof. Let ρ be a witness of $\rho_1 \sim_A^{\delta'} \rho_2$ for any ρ_1 and ρ_2, $\sigma_1 = \mathrm{Tr}_2(\rho)$ and $\sigma_2 = \mathrm{Tr}_1(\rho)$, then we have

$$\mathrm{supp}(\rho) \subseteq A \quad D(\sigma_1, \rho_1) \le \delta' \quad D(\sigma_2, \rho_2) \le \delta'$$

By the prerequisite $\vdash S_1 \sim_\delta S_2 : A \Rightarrow B$, we have

$$\sigma_1 \sim_A \sigma_2 \quad \Rightarrow \quad \llbracket S_1\rrbracket(\sigma_1) \sim_B^{\delta} \llbracket S_2\rrbracket(\sigma_2)$$

then there exits a witness σ of the lifting $\llbracket S_1\rrbracket(\sigma_1) \sim_B^{\delta} \llbracket S_2\rrbracket(\sigma_2)$ such that

$$\mathrm{supp}(\sigma) \subseteq B \quad D(\mathrm{Tr}_2(\sigma), \llbracket S_1\rrbracket(\sigma_1)) \le \delta \quad D(\mathrm{Tr}_1(\sigma), \llbracket S_2\rrbracket(\sigma_2)) \le \delta$$

Now we just need to check that σ is also a witness of the lifting $[\![S_1]\!](\rho_1) \sim_B^{\delta+\delta'} [\![S_2]\!](\rho_2)$. $\mathrm{supp}(\sigma) \subseteq B$ already holds. Besides, we also have

$$\begin{aligned} D(\mathrm{Tr}_2(\sigma), [\![S_1]\!](\rho_1)) &\leq D(\mathrm{Tr}_2(\sigma), [\![S_1]\!](\sigma_1)) + D([\![S_1]\!](\sigma_1), [\![S_1]\!](\rho_1)) \\ &\leq \delta + D(\sigma_1, \rho_1) \leq \delta + \delta' \end{aligned}$$

by property(3) in 4.17. Similarly, we also have $D(\mathrm{Tr}_1(\sigma), [\![S_2]\!](\rho_2)) \leq \delta + \delta'$. Thus $[\![S_1]\!](\rho_1) \sim_B^{\delta+\delta'} [\![S_2]\!](\rho_2)$ holds. ■

4.5 Discussions

In this section, we give another characterization of the validity of relational judgments based on approximate predicate. Besides, we add some comments about separation conditions and structural rules. All these definitions, notations, and rules in this section are independent of the main text, and readers can safely skip if they are uninterested.

Approximate Predicate

We first give some necessary definitions related to "approximate predicates", which is a binary relation over the Cartesian product of sets of quantum states.

Definition 4.20. For any projection $P \in \mathcal{H}_{S_1} \otimes \mathcal{H}_{S_2}$ and a real deviation $0 \leq \delta$, we define

(1) The approximate predicate P_δ:

$$P_\delta = \{(\rho_1, \rho_2) \mid \rho_1 \sim_P^\delta \rho_2 \text{ for } \rho_1 \in \mathcal{H}_{S_1}, \rho_2 \in \mathcal{H}_{S_2}\}$$

(2) The distance $\Delta(a, b)$ between elements $a = (a[1], a[2]), b = (b[1], b[2])$ in approximate predicates:

$$\Delta(a, b) = \max\{\mathcal{D}(a[1], b[1]), \mathcal{D}(a[2], b[2])\}$$

(3) The post-image of program pair $\langle S_1, S_2\rangle$ with respect to approximate predicate P_δ :

$$post(S_1, S_2)P_\delta = \{(\sigma_1, \sigma_2) \mid \exists(\rho_1, \rho_2) \in P_\delta.\, \sigma_1 = [\![S_1]\!](\rho_1), \sigma_2 = [\![S_2]\!](\rho_2)\}$$

The approximate predicates can also form partial orders $\leq$ concerning projections and deviations. That is, we have $A_\delta \subseteq B_\delta$ if $A \leq B$ or $A_{\delta_1} \subseteq A_{\delta_2}$ if $\delta_1 \leq \delta_2$. It is straightforward to see that

$$\forall a \in A_\delta, \exists b \in A_0 \text{ s.t. } \Delta(a, b) \leq \delta$$

from the definition of approximate predicate. In other words, the deviation δ characterizes the distance from the approximate predicate A_δ to the exact predicate A_0. Naturally, we can use the approximate predicate to describe the validity of approximate judgment in Eq. 4.5 as follows

$$\vDash S_1 \sim_\delta S_2 : A \Rightarrow B \quad \Leftrightarrow \quad post(S_1, S_2)A \subseteq B_\delta$$

This characterization works like a Hoare-type triple. That is, the post-assertion B over-approximates the expected assertion $post(S_1, S_2)A$.

We do not adopt it in our main text because the approximate predicate is not a tractable structure. A major problem is the logical operations on approximate predicates are not well-defined. Review the definition of fundamental logical operations $\neg$, $\wedge$, and $\vee$ on projections in Def. 2.8. These basic operations over two approximate predicates usually can not generate another approximate predicate. For example, we can only have the following weak set inclusion relations rather than equivalence relations

$$A_\delta \cap B_\delta \ \supseteq \ (A \wedge B)_\delta \qquad A_\delta \cup B_\delta \ \subseteq \ (A \vee B)_\delta \qquad \neg(A_\delta) \supseteq (\neg A)_\delta \tag{4.8}$$

As discussed later, these weak relations lead to the failure of structural rules in Fig. 4.6. In addition, it is not direct to check the inclusion relation between two approximate predicates. It is also difficult to characterize how quantum operations affect approximate predicates.

Separability Condition

Unlike classical programs, quantum programs exhibit quantum entanglement between subsystems, which brings unique challenges in constructing a valid quantum frame rule. In Sec. 4.3, we only introduce the simple frame rule (FRAME) since the separability problem is not our major concern. We clarify the separability conditions from [BHY⁺19] to address this problem.

Definition 4.21 (Separability Condition). Let S_1 and S_2 be two quantum programs, and $\Gamma = [\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_t]$ is a separability condition which is a partition of registers such that

$$var(\bar{q}_i) \cap var(\bar{q}_j) = \emptyset (i \neq j) \qquad var(S_1) \cup var(S_2) \subseteq \cup_{i=1}^{t} var(\bar{q}_i)$$

where $i, j \in \{1, 2, \ldots, t\}$. Then we say that a state $\rho \in \mathcal{D}(\mathcal{H}_{S_1} \otimes \mathcal{H}_{S_2})$ satisfies the separability condition Γ, written $\rho \models \Gamma$, if ρ is separable between subspaces $\mathcal{H}_{\bar{q}_i}$.

In Def .4.21, a partial density operator $\rho \in \otimes_{i=1}^{n} \mathcal{H}_{\bar{q}_i}$ is separable between subspaces $\mathcal{H}_{\bar{q}_i}$ if there exist partial density operators $\rho_{ij} \in \mathcal{D}(\mathcal{H}_{\bar{q}_i})$ such that $\rho = \sum_j (\otimes_{i=1}^{n} \rho_{ij})$. The separability condition $\Gamma = [\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_t]$ indicates that there is no quantum entanglement between all registers $\bar{q}_i$, that is, any quantum operation on register q_i would not affect another register q_j $(i \neq j)$. Besides, we use the general logic symbols to denote the relations between separability conditions based on Def. 4.21. For example, we have $[\bar{q}_1, \bar{q}_2] \wedge [\bar{q}_2, \bar{q}_3] \equiv [\bar{q}_1, \bar{q}_2, \bar{q}_3]$, $[\bar{q}_1, \bar{q}_2, \bar{q}_3] \Rightarrow \Gamma \Rightarrow \emptyset$.

Now we incorporate the separability condition into our judgments in Def. 4.12, formally defined as follows. Statically verifying the non-existence of quantum entanglement between subsystems is usually challenging, which makes quantum structural rules more subtle and restricted.

Definition 4.22 (General Validity). The judgement $\Gamma \vdash S_1 \sim_\delta S_2 : A \Rightarrow B$ is valid, written as

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

if and only if for all $\rho \in \mathcal{D}(\mathcal{H}_{S_1} \otimes \mathcal{H}_{S_2})$,

$$\rho \models \Gamma \wedge \rho \models A \quad \Rightarrow \quad [\![S_1]\!](\mathrm{Tr}_2(\rho)) \sim_B^{\delta} [\![S_2]\!](\mathrm{Tr}_1(\rho))$$

where A and B are projections, Γ is a separability condition. In particular, if $\Gamma = \emptyset$, then $\Gamma \models S_1 \sim_\delta S_2 : A \Rightarrow B$ collapsed into $\models S_1 \sim_\delta S_2 : A \Rightarrow B$ naturally.

$$(\textsc{Frame}^*)\quad \frac{\Gamma \vdash S_1 \sim_\delta S_2 : A \Rightarrow B}{\Gamma \wedge [\bar{q}, var(S_1, S_2)] \vdash S_1 \sim_\delta S_2 : A \otimes C_{\bar{q}} \Rightarrow B \otimes C_{\bar{q}}}$$

$$(\textsc{Order}^*)\quad \frac{\Gamma' \vdash S_1 \sim_{\delta'} S_2 : A' \Rightarrow B' \quad A \subseteq A' \quad B' \subseteq B \quad \delta' \le \delta \quad \Gamma \Rightarrow \Gamma'}{\Gamma \vdash S_1 \sim_\delta S_2 : A \Rightarrow B}$$

$$(\textsc{Par}^*)\quad \frac{\Gamma \vdash S_1 \sim_\delta S_2 : A \Rightarrow B \quad \Gamma' \vdash S_1' \sim_{\delta'} S_2' : C \Rightarrow D \quad var(S_1) \cap var(S_2) = \emptyset}{\Gamma \wedge \Gamma' \wedge [var(S_1), var(S_2)] \vdash S_1; S_1' \sim_{\delta+\delta'} S_2; S_2' : A \otimes C \Rightarrow B \otimes D}$$

$$(\textsc{Seq}^*)\quad \frac{\Gamma \vdash S_1 \sim_\delta S_2 : A \Rightarrow B \quad \Delta \vdash S_1' \sim_{\delta'} S_2' : B \Rightarrow C \quad \Gamma \Rightarrow^{\delta}_{(S_1,S_2)} \Delta}{\Gamma \vdash S_1; S_1' \sim_{\delta+\delta'} S_2; S_2' : A \Rightarrow C}$$

Figure 4.5: Structural aqRHL rules with separation condition.

The additional structural rules are presented in Fig. 4.5, which are the approximate versions of their counterpart in [BHY$^+$19]. The rule (Frame*) not only requires that two programs S_1 and S_2 cannot modify the free variables in $\bar{q}$ but also there is no entanglement between register $\bar{q}$ and register $var(S_1, S_2)$. The separability condition $[\bar{q}, var(S_1, S_2)]$ can degenerate into $var(\bar{q}) \cap var(S_1, S_2) = \emptyset$ if subspace $C_{\bar{q}}$ is one-dimensional. Rule (Order*) is almost the same as its counterpart in [BHY$^+$19], where order relation over deviations is included. Rule (Par*) demonstrates how to combine two independent judgments via separability condition, which is necessary for parallel computing. Rule (Seq*) is the extension of rule (Seq) in Fig. 4.1 which shows how to combine separability conditions in sequential composition. The entailment between separability conditions used in Rule (Seq*) is a direct extension from its counterpart in [BHY$^+$19], which is formally defined below.

Definition 4.23. Let Γ and Δ be two sets of measurement or separability conditions, S_1 and S_2 be two quantum programs. Δ is *couple-entailed* by Γ with respect to (S_1, S_2), written as

$$\Gamma \Rightarrow^{\delta}_{(S_1,S_2)} \Delta$$

if for any $\rho \vDash \Gamma$, whenever σ is an δ-coupling for $\langle [\![S_1]\!](\mathrm{Tr}_2(\rho)), [\![S_2]\!](\mathrm{Tr}_1(\rho))\rangle$, then $\sigma \vDash \Delta$.

Invalid structural rules

$$\text{(Dis)}\quad \frac{\vdash S_1 \sim_\delta S_2 : A \Rightarrow B \qquad \vdash S_1 \sim_\delta S_2 : A' \Rightarrow B'}{\nvdash S_1 \sim_\delta S_2 : A \vee A' \Rightarrow B \vee B'}$$

$$\text{(Con)}\quad \frac{\vdash S_1 \sim_\delta S_2 : A \Rightarrow B \qquad \vdash S_1 \sim_\delta S_2 : A' \Rightarrow B'}{\nvdash S_1 \sim_\delta S_2 : A \wedge A' \Rightarrow B \wedge B'}$$

$$\text{(Case)}\quad \frac{\vdash S_1 \sim_\delta S_2 : A \wedge C \Rightarrow B \qquad \vdash S_1 \sim_\delta S_2 : A \wedge C^{\perp} \Rightarrow B}{\nvdash S_1 \sim_\delta S_2 : A \Rightarrow B}$$

$$\text{(Compo)}\quad \frac{\vdash S_1 \sim_\delta S_2 : A \Rightarrow B \qquad \vdash S_2 \sim_{\delta'} S_3 : C \Rightarrow D}{\nvdash S_1 \sim_{\delta+\delta'} S_3 : A \circ C \Rightarrow B \circ D}$$

Figure 4.6: Invalid structural aqRHL rules.

Fig. 4.6 presents some invalid structural rules that hold in classical RHL and pRHL. These rules also do not hold in the exact quantum relational Hoare logic [BHY+19]. The reason for invalidity lies in the specialty of quantum states and the selection of subspaces as quantum predicates. In addition, there is no quantum counterpart to the classical composition rule. Unlike the natural composition of binary relations, the composition $A \circ B$ of two projections A and B cannot be well defined in the quantum context. The invalidity of these structural rules highlights that the relational reasoning of quantum programs differs significantly from that of classical probabilistic programs and poses unique challenges due to superposition and entanglement. Readers can find more discussion in Sec. 4.5 if they are interested in the failure of rules in Fig. 4.6.

We use approximate predicates and some concrete examples to explain these difficulties straightforwardly. Assume $S = S_1 = S_2 = \textbf{if } (\square m \cdot \mathcal{M}[\bar{q}] = m \rightarrow \textbf{skip}) \textbf{ fi}$ with $\mathcal{M} = \{|0\rangle\langle 0|, |1\rangle\langle 1|\}$, then we have $[\![S]\!](|+\rangle\langle +|) = I/2$. Let $|\beta_0\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$,

$|\beta_1\rangle = (|00\rangle - |11\rangle)/\sqrt{2}$, $A = |{+}{+}\rangle\langle{+}{+}|$, $B = |\beta_0\rangle\langle\beta_0|$, $B' = |\beta_1\rangle\langle\beta_1|$, We can observe that

$$\vDash S_1 \sim S_2 : A \Rightarrow B \qquad\qquad \vDash S_1 \sim S_2 : A \Rightarrow B'$$

However, there is no coupling $I/2 \sim_{\mathbf{0}} I/2$. Hence, we can not infer $\vdash S_1 \sim S_2 : A \Rightarrow B \wedge B'$ from the invalid rule (Con), where $B \wedge B' = \mathbf{0}$. We only have

$$\frac{\vdash S_1 \sim_\delta S_2 : A \Rightarrow B \qquad \vdash S_1 \sim_\delta S_2 : A' \Rightarrow B'}{post(S_1, S_2)A \cap post(S_1, S_2)A \Rightarrow B_\delta \cap B'_\delta \nRightarrow (B \wedge B')_\delta}$$

where $post(S_1, S_2)A \cap post(S_1, S_2)A$ and $B_\delta \cap B_\delta$ may not even be approximate predicates. The same problem also arises for rule (Compo), that is, $A_\delta \circ B_\delta$ generally do not form an approximate predicate C'_δ such that $C'_\delta = A_\delta \circ B_\delta$.

Here is another example for rule (Dis). Assume $S_1 = \textbf{skip}$ and $S_2 = \textbf{if}\ (\square m \cdot \mathcal{M}[\bar{q}] = m \longrightarrow \textbf{skip})\ \textbf{fi}$ with $\mathcal{M} = \{|0\rangle\langle 0|, |1\rangle\langle 1|\}$, then we have $[\![S_1]\!](\rho) = \rho$ and $[\![S_2]\!](|+\rangle\langle +|) = [\![S_2]\!](|-\rangle\langle -|) = I/2$. Let $A = |0+\rangle\langle 0+|$, $A' = |0-\rangle\langle 0-|$, $B = |00\rangle\langle 00|$, we can observe that

$$\vDash S_1 \sim_{1/4} S_2 : A \Rightarrow B \qquad\qquad \vDash S_1 \sim_{1/4} S_2 : A' \Rightarrow B$$

Once again, the invalidity of rule (Dis) means that we cannot infer $\vdash S_1 \sim_{1/4} S_2 : A \vee A' \Rightarrow B$ since $|0\rangle\langle 0| \sim_{A \vee A'} |1\rangle\langle 1| \nRightarrow [\![S_1]\!](|0\rangle\langle 0|) \sim_B^{1/4} [\![S_2]\!](|1\rangle\langle 1|)$. Similarly, the direct quantum analog (Case) of classical case rule is also invalid.

Chapter 5

Applications

In this chapter, we demonstrate how to use our logic to reason about the correctness of quantum programs. To be specific, we use QIL to verify the correctness and existence of bugs in examples of quantum teleportation 5.1, Grover's algorithm 5.2 and RUS algorithm 5.3. In addition, in examples of bit flip code 5.4, approximation of unitary gates 5.5 and approximate Quantum Fourier Transform 5.6, aqRHL is applied to reason about the approximate equivalence between quantum programs for verifying correctness indirectly.

5.1 Quantum Teleportation

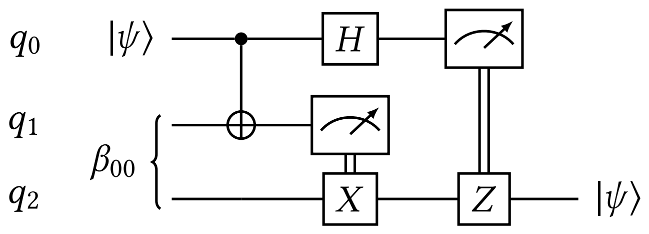

Figure 5.1: Circuit for teleportation.

Quantum teleportation is a technique that transports quantum states through classical communication. As shown in the Fig. 5.1, 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.

Figure. 5.2a 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. 5.2b, we introduce a bug of type "incorrect operations" according to [HM19b]. 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 avoiding reasoning about exponentially many execution paths.

5.2 Grover's Algorithm

Grover's algorithm [Gro96] 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. 5.3, where we put the aQHL and QIL predicates together to show their connection.

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 $f : \{0,1\}^d \to \{0,1\}$. 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

```
    $[R \otimes \beta_{00}\beta_{00}^\dagger]$
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

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

Figure 5.2: Reasoning the quantum teleportation program by QIL.

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. 5.3, it would be better to give a more intuitive explanation for Grover iteration from a geometric point of 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

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

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

Figure 5.3: Reasoning Grover's algorithm by QIL.

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. 5.3, 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 $post([\![S]\!]_\epsilon)$. 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 **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 does not affect the subspace $|\alpha\rangle\langle\alpha| + |\beta\rangle\langle\beta|$. [1] But we can still capture any kind of errors that makes state $G^i |\psi\rangle$ $(0 \leq i \leq R)$ end up not lying in the subspace $|\alpha\rangle\langle\alpha| + |\beta\rangle\langle\beta|$, e.g., an erroneous implementation of Ph.

5.3 Repeat Until Success Algorithm

We take a simple program implementing a repeat-until-success [PS14, BRS15] (RUS) algorithm as another example. RUS algorithms offer exact, fault-tolerant implementations of a large set of single-qubit unitary gates that can improve the approximate decomposition of single-qubit unitaries significantly. Implementing the algorithm requires wrapping RUS circuits into while-loops, which can be easily erroneous. Recall the smallest circuit for the loop body found in [PS14] in Fig. 2.3.

In this subsection, we demonstrate how to use QIL to reason about the correctness and existence of bugs in the implementation of the RUS algorithm. Fig. 5.4a and 5.4b

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

```
        [|0⟩⟨0| ⊗ R]
0 :   q1 := Xq1;
      [ok: |1⟩⟨1| ⊗ R]
1 :   while M[q1] = 1 do
        [ok: |1⟩⟨1| ⊗ R]
2 :     q1 := Xq1;
        [ok: |0⟩⟨0| ⊗ R]
3 :     q1 := Hq1;
4 :     q1 := Tq1;
5 :     (q1, q2) := CNOT(q1, q2);
6 :     q1 := Hq1;
7 :     (q1, q2) := CNOT(q1, q2);
8 :     q1 := Tq1;
9 :     q1 := Hq1
        [ok: U(|0⟩⟨0| ⊗ R)U†]
10 :  od;
      [ok: |0⟩⟨0| ⊗ VRV†]
11 :  assert(q2, VRV†)
      [er: 0]
```

(a) RUS program

```
        [|0⟩⟨0| ⊗ R]
0 :   q1 := Xq1;
      [ok: |1⟩⟨1| ⊗ R]
1 :   while M[q1] = 1 do
        [ok: |1⟩⟨1| ⊗ Q_n]
2 :     ~~q1 := Xq1;~~
        [ok: |1⟩⟨1| ⊗ Q_n]
3 :     q1 := Hq1;
4 :     q1 := Tq1;
5 :     (q1, q2) := CNOT(q1, q2);
6 :     q1 := Hq1;
7 :     (q1, q2) := CNOT(q1, q2);
8 :     q1 := Tq1;
9 :     q1 := Hq1
        [ok: P_{n+1}]
10 :  od;
      [ok: |0⟩⟨0| ⊗ Q_N]
11 :  assert(q2, VRV†)
      [er: |0⟩⟨0| ⊗ (I − VRV†)Q_N(I − VRV†)]
```

(b) Erroneous program

$M = \{M_0, M_1\}$, where $M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|$ $\quad V = (I + i\sqrt{2}X)/\sqrt{3}$

$U = (X \otimes I - i\sqrt{3}I \otimes V)/2$ $\quad W = (X \otimes I - i\sqrt{3}I \otimes V^\dagger)/2$

$Q_n = V^{\dagger^n} R V^n$ $\quad P_0 = |1\rangle\langle 1| \otimes R$ $\quad P_{n+1} = W(|1\rangle\langle 1| \otimes Q_n)W^\dagger$

Figure 5.4: Reasoning an RUS program by QIL

are the correct and erroneous implementations of the RUS procedure corresponding to Fig. 5.4 along with the proof sketch, respectively. Assume q_2 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_2. The erroneous implementation contains a mistake at line 2: forgetting to restore the auxiliary qubit q_1 before entering the RUS circuit (lines 3-9). The seemingly artificial implementation error corresponds to a bug type “incorrect quantum initial values" as reported in [HM19b]: the initial value of the auxiliary qubit q_1 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 $[ok\colon |1\rangle\langle 1| \otimes R]$ by 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

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

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

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

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. 5.4, 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**$(q_1, |0\rangle\langle 0|)$ after line 2 would immediately capture the bug in Fig. 5.4b with postcondition $[er\colon |1\rangle\langle 1| \otimes Q_0]$, indicating an incorrect initial value of the RUS circuit.

5.4 Bit Flip Code

Quantum error correction is crucial for reliable information processing in the presence of noise, as large quantum logical gates and circuits remain challenging to construct with high reliability. In this example, we employ our logic to demonstrate how the *bit flip code* can resist the noisy *bit flip channel*. The complete quantum circuit for single-bit flip error correction is shown in Fig. 5.5.

The entire process can be divided into four parts by dashed lines. Assuming we have a single-qubit state $|\psi\rangle = a|0\rangle + b|1\rangle$ in qubit q_3, we introduce auxiliary qubits q_1 and q_2 to construct the three-qubit bit flip code of $|\psi\rangle$ by employing two *CNOT* gates. All qubits pass through a noisy channel described by the gate $\mathcal{E}$. The subsequent part involves error detection and correction achieved via quantum measurement $\mathcal{M} = \{M_0, M_1, M_2, M_3\}$,

$$M_0 = |000\rangle\langle 000| + |111\rangle\langle 111| \quad M_1 = |100\rangle\langle 100| + |011\rangle\langle 011|$$
$$M_2 = |010\rangle\langle 010| + |101\rangle\langle 101| \quad M_3 = |001\rangle\langle 001| + |110\rangle\langle 110|$$

and controlled-X gates based on measurement outcomes. The last part is the inverse of the encoder that decodes the recovered state. The qubit q_3 would output the desired state $|\psi\rangle$ while auxiliary qubits q_1 and q_2 would be reset to the default state $|0\rangle$.

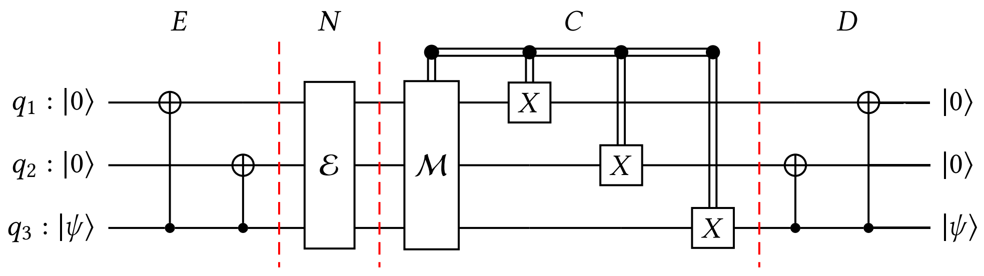

Figure 5.5: Circuit of Bit Flip Code.

The circuit in Fig. 5.5 can be represented as the program $S_{\text{BFC}} \equiv E; \mathcal{E}; C; D$, defined as follows,

$E \equiv (q_3, q_1) := CNOT(q_3, q_1); (q_3, q_2) := CNOT(q_3, q_2);$

$C \equiv \textbf{if } \mathcal{M}[q_1, q_2, q_3] = 0 \rightarrow \textbf{skip } \square 1 \rightarrow q_1 := X[q_1] \ \square 2 \rightarrow q_2 := X[q_2] \ \square 3 \rightarrow q_3 := X[q_3] \textbf{ fi};$

$D \equiv (q_3, q_2) := CNOT(q_3, q_2); (q_3, q_1) := CNOT(q_3, q_1);$

Correctness of BFC Let the gate $\mathcal{E}$ denote a theoretical noise model such that a one-bit flip can occur at most once,

$$\mathcal{E}(\rho) = p\rho + (1-p)/3 \sum_i X_i \rho X_i \qquad (i \in \{1,2,3\})$$

for any ρ over register (q_1, q_2, q_3), where $p \in [0,1]$ denotes the probability that no error occurs, X_i denotes X gate on qubit q_i. We want to show that program S_{BFC} can resist errors caused by noise $\mathcal{E}$, which can be described by the following judgment,

$$\vdash S_{\text{BFC}} \sim \textbf{skip} : |0\rangle\langle 0|_{(q_1,q_2)} \otimes \equiv_{(q_3,q_4)} \;\Rightarrow\; |0\rangle\langle 0|_{(q_1,q_2)} \otimes \equiv_{(q_3,q_4)} \tag{5.1}$$

where **skip** statement is defined on qubit q_4. Now we show how to use our proof system to derive Eq. 5.1. Let $P_0 = |0\rangle\langle 0|_{(q_1,q_2)} \otimes \equiv_{(q_3,q_4)}$, then we apply rule (Ut-L) to have

$$\begin{aligned} &\vdash E \sim \textbf{skip} : P_0 \;\Rightarrow\; P_1 \\ &\vdash D \sim \textbf{skip} : P_1 \;\Rightarrow\; P_0 \\ &\vdash \mathcal{E} \sim \textbf{skip} : P_1 \;\Rightarrow\; P_2 \end{aligned} \tag{5.2}$$

where $U(q_1,q_2,q_3) = CNOT(q_3,q_2)CNOT(q_3,q_1)$, $P_1 = (U \otimes I_{q_4})P_0(U^\dagger \otimes I_{q_4})$, $P_2 = P_1 + \sum_{i=0}^{3} X_i P_1 X_i$. We can verify the validity of the following measurement condition,

$$\mathcal{M} \approx I : P_2 \;\Rightarrow\; \{(q_i, X_i P_1 X_i)\} \qquad (i \in \{0,1,2,3\}) \tag{5.3}$$

such that $\sum q_i = 1$, where X_0 denotes I particularly. For the bodies of C, we have

$$\vdash q_i := X[q_i] \sim \textbf{skip} : \; X_i P_1 X_i \;\Rightarrow\; P_1 \qquad (i \in \{0,1,2,3\}) \tag{5.4}$$

by rule (Ut-L). We use (IF-L) to combine Eq. 5.3 and 5.4 to obtain

$$\vdash C \sim \textbf{skip} : P_2 \;\Rightarrow\; P_1 \tag{5.5}$$

Finally, we can apply rule (Seq) to combine Eq. 5.2 and 5.5 to get Eq. 5.1.

Multiple bit flip errors What would happen to the circuit in Fig. 5.5 if the bit flip error in gate $\mathcal{E}$ can occur on any qubit independently? Such a noise model can be characterized as

$$\mathcal{E}'(\rho) = p\rho + (1-\sqrt[3]{p})\sqrt[3]{p}^2(X_1\rho X_1 + X_2\rho X_2 + X_3\rho X_3) + (1-\sqrt[3]{p})^2\sqrt[3]{p}(X_1X_2\rho X_1X_2 + X_1X_3\rho X_1X_3 + X_2X_3\rho X_2X_3) + (1-\sqrt[3]{p})^3 X_1X_2X_3\rho X_1X_2X_3$$

for any ρ over register (q_1', q_2', q_3'), where $p \in [0,1]$ still denotes the probability that no error occurs. Let $S_{\text{BFC}}' \equiv E; \mathcal{E}'; C; D$ be the corresponding program defined on register (q_1', q_2', q_3'). The difference between programs S_{BFC} and S_{BFC}' can be characterized by the judgement

$$\vdash S_{\text{BFC}} \sim_\delta S_{\text{BFC}}' : |0\rangle\langle 0|_{(q_1,q_2,q_1',q_2')} \otimes \equiv_{(q_3,q_3')} \Rightarrow |0\rangle\langle 0|_{(q_1,q_2,q_1',q_2')} \otimes \equiv_{(q_3,q_3')} \tag{5.6}$$

with $\delta = (1-p-3(1-\sqrt[3]{p})\sqrt[3]{p}^2)/2$. It is direct to see that $\delta = 0$ when $p = 1$, and $\delta = 1/2$ when $p = 0$, which indicates that bit flip code can not deal with multiple errors.

The proof of the above judgment is similar to Eq. 5.1. Let $Q_0 = |0\rangle\langle 0|_{(q_1,q_2,q_1',q_2')} \otimes \equiv_{(q_3,q_3')}$, then we apply rule (Ut-L) to have

$$\begin{aligned} &\vdash E \sim E : Q_0 \Rightarrow Q_1 \\ &\vdash D \sim D : Q_1 \Rightarrow Q_0 \end{aligned} \tag{5.7}$$

where $R(q_1, q_2, q_3, q_1', q_2', q_3') = CNOT(q_3, q_2)CNOT(q_3, q_1)CNOT(q_3', q_2')CNOT(q_3', q_1')$, $Q_1 = RQ_0R^\dagger$. Besides, we also have

$$\vdash \mathcal{E} \sim_\delta \mathcal{E}' : Q_1 \Rightarrow Q_2 \tag{5.8}$$

where $\delta = \|\mathcal{E} - \mathcal{E}'\|_\diamond/2 \le (1-p-3(1-\sqrt[3]{p})\sqrt[3]{p}^2)/2$, $Q_2 = Q_1 + \sum_i X_iX_i'Q_1X_iX_i'$. Here X_i and X_i' denote X gates on qubits q_i and q_i' respectively. We can verify the validity of the following measurement condition,

$$\mathcal{M} \approx \mathcal{M} : Q_2 \Rightarrow \{(q_i, X_iX_i'Q_1X_iX_i')\} \qquad (i \in \{0,1,2,3\}) \tag{5.9}$$

such that $\sum_{i=0}^{3} q_i = 1$, where X_0 and X'_0 denotes I particularly. For the bodies of C, we have

$$\vdash q_i := X[q_i] \sim q'_i := X[q'_i] : X_i X'_i Q_1 X_i X'_i \Rightarrow Q_1 \qquad (i \in \{0,1,2,3\}) \tag{5.10}$$

by rule (Ut-L), where $i \in \{0,1,2,3\}$. We use (IF-L) to combine Eq. 5.9 and 5.10 to obtain

$$\vdash C \sim C : Q_2 \Rightarrow Q_1 \tag{5.11}$$

Finally, we can apply rule (Seq) to combine Eq. 5.7, Eq. 5.8 and 5.11 to get Eq. 5.6.

5.5 Approximation of Unitary Gates

It is well-known that we can decompose arbitrary unitary operators into gates from a universal gate set [NC11]. One of the most important universal gate sets is the Clifford+T gates. For instance, we want to use the set of Clifford+T gates to approximate the following unitary $V = (I + i\sqrt{2}X)/\sqrt{3}$ defined on qubit q_0,

$$S_1 \equiv q_0 := V[q_0];$$

We will reason about the approximate equivalence between the following two approaches of approximating V. In subsection 5.5, unitary V is decomposed as $V = HR_z(\theta)H$, where the single-qubit $R_z(\theta)$ gate can be directly approximated by ancilla-free Clifford+T gates using the algorithm in [RS16]. In subsection 5.5, we discuss the repeat until success [PS14] circuit to stimulate V exactly or approximately.

Approximation of z-rotation by direct decomposition

The approximation of an arbitrary single-qubit z-rotation gate

$$R_z(\theta) = e^{-i\theta Z/2} = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$$

by Clifford+T circuits is an important problem since it lays the foundation for the approximate decomposition of arbitrary multi-qubit gates. Assume the basic z-rotation gate $R_z(\theta)$ is approximated by a single-qubit gate U that can be decomposed as a sequence of Clifford+T gates, the corresponding program S_2 is defined as

$$S_2 \equiv q_0' := H[q_0']; q_0' = U[q_0']; q_0' := H[q_0']; \tag{5.12}$$

We adopt the algorithm from [RS16] to find the appropriate decomposition of U. Given $\theta = -\arctan(\sqrt{2})$ and $\epsilon = 10^{-2}$, we have

$$\begin{aligned} U \ &= HTSHTHTSHTSHTSHTSHTHTSHTHTS \\ &\quad HTHTSHTSHTSHTHTSHTHTHSSSWWW \end{aligned}$$

where W denotes the scalar $e^{i\pi/4}$. The precision ϵ of the approximation is defined as $\|R_z(\theta) - U\| \leq \epsilon$, where $\|A\|$ denotes the operator norm of the matrix A, i.e. the largest eigenvalue of $\sqrt{A^\dagger A}$. The following judgment characterizes the approximate equivalence between S_1 and S_2.

$$\vDash S_1 \sim_{0.002} S_2 : \equiv_{(q_0,q_0')} \Rightarrow \equiv_{(q_0,q_0')} \tag{5.13}$$

We prove this by employing rule (Ut-id), where the deviation $\delta = \|U \cdot U^\dagger - R_z(\theta) \cdot R_z(\theta)^\dagger\|_\diamond/2 \approx 0.002$.

Approximation of unitary gate by RUS algorithm

As we mentioned before, the repeat until success (RUS) circuit proposed by Paetznick et al. in [BRS15, PS14] is a powerful decomposition technique to approximate single-qubit unitaries. The RUS circuit introduces ancilla qubits with a small number of non-Clifford gates to manipulate the target qubits based on the measurement results from the ancilla qubits. Recall the aim of Fig. 2.3 is to apply the unitary operator $V = (I + i\sqrt{2}X)/\sqrt{3}$ to the target qubit q_2. The program terminates when the measurement of q_1 returns 0. The entire process can be viewed as a quantum loop, with qubit q_1 simultaneously serving as the loop body's control qubit and the auxiliary qubit for the RUS circuit. The whole process can be denoted by program S_3,

$$S_3 \equiv \ q_1 := |0\rangle ; q_1 := X[q_1]; R; \textbf{while } \mathcal{M}[q_1] = 1 \textbf{ do } R \textbf{ od}; \tag{5.14}$$

where we add some statements before the while-loop to ensure the circuit in Fig. 2.3 at least runs once. The loop body R,

$$\begin{aligned} R \equiv\ & q_1 := X[q_1];\ q_1 := H[q_1];\ q_1 := T[q_1];\ (q_1, q_2) := CNOT[(q_1, q_2)]; \\ & q_1 := H[q_1];\ (q_1, q_2) := CNOT[(q_1, q_2)];\ q_1 := T[q_1];\ q_1 := H[q_1]; \end{aligned}$$

can be characterized as a composite unitary, that is, $(q_1, q_2) := U_R[(q_1, q_2)]$, where $U_R = (I \otimes I - i\sqrt{3}X \otimes V)/2$. To align with the measurement $\mathcal{M} = \{M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|\}$ in the loop, we add a X gate in qubit q_1 at the beginning of the loop body R.

In the following, we first reason the exact correctness of the RUS program and then use that fact to prove the approximate correctness of the bounded RUS program.

Correctness of RUS. The following judgment describes the correctness of the RUS algorithm, that is, the equivalence between programs S_1 and S_3 on the working qubit.

$$\vDash S_1 \sim S_3 : I_{q_1} \otimes \equiv_{(q_2, q_0)} \Rightarrow |0\rangle\langle 0|_{q_1} \otimes \equiv_{(q_2, q_0)} \tag{5.15}$$

We explain using our proof rules to get the judgment 5.15. First, we can infer

$$\begin{gathered} \vdash q_1 := |0\rangle \sim \textbf{skip} : I_{q_1} \otimes \equiv_{(q_2, q_0)} \Rightarrow |0\rangle\langle 0|_{q_1} \otimes \equiv_{(q_2, q_0)} \\ \vdash q_1 := X[q_1] \sim \textbf{skip} : |0\rangle\langle 0|_{q_1} \otimes \equiv_{(q_2, q_0)} \Rightarrow A \\ \vdash R \sim \textbf{skip} : A \Rightarrow B \end{gathered} \tag{5.16}$$

by rules (Init-L) and (Ut-L) , where we define $A = |1\rangle\langle 1|_{q_1} \otimes \equiv_{(q_2, q_0)}$ and $B = (U_R \otimes I_{q_0}) A (U_R^\dagger \otimes I_{q_0})$. Let $|\psi\rangle$ be an arbitrary single-qubit vector state; then we can find that $|\varphi\rangle\langle\varphi|$ is exactly the invariant of the loop statement in S_3, i.e.,

$$|\varphi\rangle\langle\varphi| \xrightarrow{\mathcal{M}} \begin{cases} \frac{3}{4} |\varphi_0\rangle\langle\varphi_0| & (m = 0) \\ \frac{1}{4} |\varphi_1\rangle\langle\varphi_1| \xrightarrow{U_R} \frac{1}{4} |\varphi\rangle\langle\varphi| & (m = 1) \end{cases} \tag{5.17}$$

where $|\varphi_0\rangle = |0\rangle \otimes V|\psi\rangle$, $|\varphi_1\rangle = |1\rangle \otimes |\psi\rangle$, $|\varphi\rangle = U_R|\varphi_1\rangle$. Thus the measurement condition

$$\mathcal{M} \approx I : B \Rightarrow \{(3/4, C), (1/4, A)\} \tag{5.18}$$

holds from Eq. 5.17, where $C = |0\rangle\langle 0|_{q_1} \otimes ((I_{q_2} \otimes V^\dagger) \equiv_{(q_2,q_0)} (I_{q_2} \otimes V))$. Then we apply rule (LP-L) to have

$$\vdash \textbf{while } \mathcal{M}[q_1] = 1 \textbf{ do } R \textbf{ od} \sim \textbf{skip} : B \Rightarrow C \tag{5.19}$$

from the last equation in 5.16 and Eq. 5.18. Next, we apply the right version of rule (UT-L) to have

$$\vdash \textbf{skip} \sim q_0 := V[q_0] : C \Rightarrow |0\rangle\langle 0|_{q_1} \otimes \equiv_{(q_2,q_0)} \tag{5.20}$$

Finally, we apply rule (SEQ) to combine Eq. 5.16, 5.19 and 5.20 to get Eq. 5.15, together with the soundness of our logic.

Approximate RUS by a bounded loop. In practice, we usually set bounds for loop statements to avoid infinite unrolling. The approximate RUS can be implemented directly via a bounded loop. We add register $\bar{r}$ as a counter for the while-loop. The counter is increased during each iteration by applying operator U_+ which performs $U_+ |n\rangle = |n + 1 \text{ mod } 2^{|\bar{r}|}\rangle$. The yes-no measurement $\mathcal{M}' = \{M_0', M_1'\}$

$$M_1' = \textstyle\sum_{i=0}^{N-2} |i\rangle\langle i| \qquad M_0' = I - M_1'$$

on register $\bar{r}$ checks whether the number of iterations exceeds the upper bound N, where $N \le 2^{|\bar{r}|} - 1$. If measurement $\mathcal{M}'$ gives a no answer, an initiation statement would be applied to qubit q_1 to ensure the loop's termination. The corresponding bounded program S_3' is defined on register $(q_1, q_2, \bar{r})$ as follows,

$$S_3' \equiv \; q_1 := |0\rangle \,; q_1 := X[q_1]; R; \textbf{while } \mathcal{M}[q_1] = 1 \textbf{ do } R' \textbf{ od}; \tag{5.21}$$

where the loop body R' becomes

$$R' \equiv \; R; \textbf{if } \mathcal{M}'[\bar{r}] = 0 \to q_1 := |0\rangle \;\square 1 \to \textbf{skip fi}; \bar{r} := U_+\bar{r};$$

The approximate equivalence between program S_1 and S_3' can be characterized by the following judgment.

$$\vDash S_1 \sim_{1/(2\cdot 4^N)} S_3' : |0\rangle\langle 0|_{\bar{r}} \otimes I_{q_1} \otimes \equiv_{(q_2,q_0)} \Rightarrow M_1' \otimes |0\rangle\langle 0|_{q_1} \otimes \equiv_{(q_2,q_0)} \tag{5.22}$$

As expected, the deviation should decrease exponentially with respect to N.

The proof of Eq. 5.22 works as follows. First, we apply rules (Seq) and (Frame) to Eq. 5.16 to have

$$\vdash q_1 := |0\rangle\,; q_1 := X[q_1]; R; \sim \textbf{skip} : |0\rangle\langle 0|_{\bar{r}} \otimes I_{q_1} \otimes \equiv_{(q_2,q_0)} \Rightarrow\ |0\rangle\langle 0|_{\bar{r}} \otimes B \tag{5.23}$$

where register $\bar{r}$ is not used in the left program, and the dimension of predicate $|0\rangle\langle 0|_{\bar{r}}$ is 1. For any $0 \le k < N$, let $P_k = |k\rangle\langle k|_{\bar{r}} \otimes B$, $Q_k = |k\rangle\langle k|_{\bar{r}} \otimes A$, $P_N = |N\rangle\langle N|_{\bar{r}} \otimes |0\rangle\langle 0|_{q_1} \otimes \text{proj}(\text{Tr}_{q_1}(B))$. Then we have the following judgments

$$\vdash R \sim \textbf{skip} : Q_k \ \Rightarrow\ P_k$$

$$\vdash \textbf{if } \mathcal{M}'[\bar{r}] = 0 \longrightarrow q_1 := |0\rangle \ \square 1 \longrightarrow \textbf{skip fi} \sim \textbf{skip} : P_k \ \Rightarrow\ P_k$$

$$\vdash \bar{r} := U_+\bar{r} \sim \textbf{skip} : P_k \ \Rightarrow\ P_{k+1}$$

by rule (Frame) on the last equation in 5.16, rules (IF-L) and (Ut-L), respectively. We combine above judgments by rule (Seq) to obtain

$$\vdash R' \sim \textbf{skip} : Q_k \ \Rightarrow\ P_{k+1} \tag{5.24}$$

for the loop body R'. Similarly, we obtain the following measurement conditions

$$\begin{aligned} \mathcal{M} &\approx I : P_k \Rightarrow \{(3/4, M_1' \otimes C), (1/4, Q_k)\} \\ \mathcal{M} &\approx_{\{1/2,0\}} I : P_N \Rightarrow \{(1, M_1' \otimes C), (0, I)\} \end{aligned} \tag{5.25}$$

by Eq. 5.17. The above last measurement condition is trivial since the deviation for $M_1' \otimes C$ is set to the maximum value $1/2$. Then we apply rule (LP*-L) to combine Eq. 5.24 and 5.25 to have

$$\vdash \textbf{while } \mathcal{M}[q_1] = 1 \textbf{ do } R' \textbf{ od} \sim_\delta \textbf{skip} : |0\rangle\langle 0|_{\bar{r}} \otimes B \ \Rightarrow\ M_1' \otimes C \tag{5.26}$$

where $q_k = 3/4$, $p_k = 1/4$, $\delta_k = 0$, $\delta_N = 1/2$ and $\delta = \delta_N \cdot \prod_{k=0}^{N-1} p_k = 1/(2 \cdot 4^N)$. Next, we apply rule (Ut-L) to have

$$\vdash \textbf{skip} \sim q_0 := V[q_0] : M_1' \otimes C \ \Rightarrow\ M_1' \otimes |0\rangle\langle 0|_{q_1} \otimes \equiv_{(q_2,q_0)} \tag{5.27}$$

Finally, we apply rule (Seq) to combine Eq. 5.23, 5.26 and 5.27 to obtain Eq. 5.22, together with the soundness of our logic.

Conclusion

We use rule (Comp) to combine judgments 5.13 and 5.22 to have

$$S_2 \sim_{0.002+1/(2\cdot 4^N)} S_3' : \ |0\rangle\langle 0|_{\bar{r}} \otimes I_{q_1} \otimes \equiv_{(q_2,q_0')} \Rightarrow \ M_1' \otimes |0\rangle\langle 0|_{q_1} \otimes \equiv_{(q_2,q_0')} \tag{5.28}$$

Therefore, we have proved the approximate equivalence between the approximation of unitary V via the RUS algorithm or the approximate $R_z(\theta)$ gate.

Approximate RUS by case statement We can also unroll the loop statement in program S_3 into a N-depth branching structure composed of case statements. The corresponding approximate program S_3'' defined on qubits (q_1, q_2) can be given as follows,

$$S_3'' \equiv \ q_1 := |0\rangle\,; q_1 := X[q_1]; R; F_N; \tag{5.29}$$

where F_N is a finite recursion of **if** statements

$$F_N \equiv \ \textbf{if}\ \mathcal{M}[q_1] = 0 \rightarrow \textbf{skip}\ \square 1 \rightarrow R; F_{N-1}; \textbf{fi};$$

with $F_0 \equiv$ **skip**. Similarly, the approximate equivalence between program S_1 and S_3'' can be characterized by the following judgment

$$\vDash S_1 \sim_{1/(2\cdot 4^N)} S_3'' : I_{q_1} \otimes \equiv_{(q_2,q_0)} \Rightarrow \ |0\rangle\langle 0|_{q_1} \otimes \equiv_{(q_2,q_0)} \tag{5.30}$$

with the same deviation. We first start with reasoning the difference between F_N with **skip**. We apply rule (Skip*) to program F_0 to have

$$\vdash F_0 \sim_{1/2} \textbf{skip} : B \Rightarrow C$$

where the deviation is trivially set to its maximum value $1/2$. Similarly, we apply rule (Seq) and (IF-L) to obtain

$$\vdash F_1 \sim_{1/8} \textbf{skip} : B \Rightarrow C$$

from the last equation in 5.16 and 5.18. We can repeat the above reasoning N times to the case statement F_i ($i \in [1 .. N]]$) in sequence, thus we have

$$\vdash F_N \sim_{1/(2\cdot 4^N)} \textbf{skip} : B \Rightarrow C \tag{5.31}$$

Finally, we apply (Seq) to combine Eq. 5.16, 5.31 and 5.20 to have Eq. 5.30, together with the soundness of our logic.

5.6 Approximate Quantum Fourier Transform

Objective. As a quantum analog of the classical discrete Fourier transform, *quantum Fourier transform* (QFT) [Cop02] 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. [Cop02] proposes a direct AQFT based on ignoring gates related to high-order terms. Cleve and Watrous [CW00] parallelized the phase estimation procedure to perform AQFT with lower circuit depth.

This section uses aqRHL to reason about how well AQFT approximates QFT. Formally, let S_{QFT} and S_{AQFT} be the corresponding quantum programs for QFT and AQFT, respectively. We study the following judgment

$$\vDash S_{\text{QFT}} \sim_\delta S_{\text{AQFT}} : \equiv \Rightarrow \equiv \tag{5.32}$$

that characterize the approximate equivalence between S_{QFT} and S_{AQFT}. We discuss [Cop02] and [CW00] in turn.

Notations. For an n qubit system, QFT on a *computational basis state* $|x\rangle = |x_1x_2\dots x_n\rangle$ is defined as the linear operation U such that

$$U|x\rangle = |\psi_x\rangle = \tfrac{1}{\sqrt{2^n}}\textstyle\sum_{y=0}^{N-1}(e^{2\pi i/N})^{x\cdot y}|y\rangle \tag{5.33}$$

where $N = 2^n$, $|\psi_x\rangle$ is called a *Fourier basis state* with respect to computational basis state $|x\rangle$, $x \cdot y$ denotes the multiplication between the binary representation of x and y. $|\psi_x\rangle$ can be given in the following useful tensor product representation

$$|\psi_x\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + e^{2\pi i(0.x_n)}|1\rangle)(|0\rangle + e^{2\pi i(0.x_{n-1}x_n)}|1\rangle)\cdots(|0\rangle + e^{2\pi i(0.x_1\dots x_n)}|1\rangle) \tag{5.34}$$

where $0.x_i \dots x_j$ denotes the binary fraction $x_i/2 + x_{i+1}/4 + \cdots + x_j/2^{j-i+1}$. For convenience, we define $|\mu_\theta\rangle = (|0\rangle + e^{2\pi i\theta}|1\rangle)/\sqrt{2}$ to denote the terms in Eq. 5.34. State $|\mu_\theta\rangle$ can be obtained by applying the phase shift gate $P(2\pi\theta)$ (mentioned in Chapter 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

$$R_m = P(2\pi/2^m) = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^m} \end{pmatrix}$$

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 with $\theta = 2\pi/2^m$.

AQFT Circuit proposed by CopperSmith

An direct circuit implementation of QFT from [NC11, p. 219] is shown in Fig. 5.6. Given a computational basis $|x\rangle$ on qubits $\{q_1, \dots, q_n\}$, the output of QFT circuit in Fig. 5.6 is the Fourier basis state $|\psi_x\rangle = \otimes_{i=1}^n |\mu_{0.x_{n-i+1}\dots x_n}\rangle$. Let k be the number of significant phase shift gates remaining in AQFT [Cop02]. As shown in Fig. 5.6, the AQFT circuit is the remaining circuit where all CR_m $(k < m)$ gates in the shadow boxes are excluded. Consequently, the output of the AQFT circuit would be

$$|\psi_x'\rangle = (\otimes_{i=1}^k |\mu_{0.x_{n-i+1}\dots x_n}\rangle) \otimes (\otimes_{i=k+1}^n |\mu_{0.x_{n-i+1}\dots x_{n-i+k}}\rangle)$$

where $|\mu_{0.x_{n-i+}\dots x_n}\rangle$ is approximated by $|\mu_{0.x_{n-i+1}\dots x_{n-i+k}}\rangle$ when $k < i$.

The programs for the circuits of QFT and AQFT are shown in Fig. 5.7, where $[a, b]$ denotes the closed interval of integers between a and b. Subprogram S_i denotes the block labeled by i in Fig. 5.6, T_i denotes the shadow boxes in the block S_i, and S_i' denotes the remaining program that excludes T_i from S_i. We show how to use our logic to obtain the judgment 5.32.

For the shared program S_i, we have

$$\vdash S_i \sim S_i' : \equiv \Rightarrow \equiv \quad i \in [n-k+1, n] \tag{5.35}$$

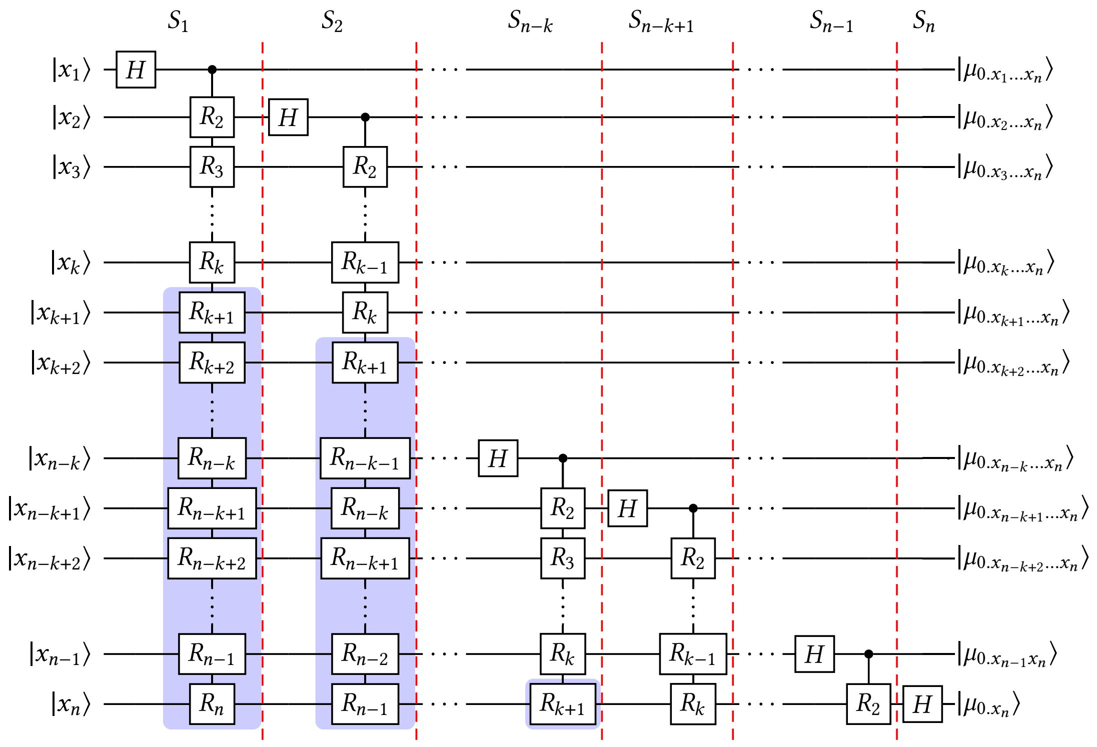

Figure 5.6: QFT circuit in [NC11]. To save space, we put multiple controlled gates CR_m that share the same control qubit in one column.

$$
\begin{aligned}
S_{\mathrm{QFT}} &:= S_1; S_2; \ldots; S_{n-k}; S_{n-k+1}; \ldots; S_n; \\
S_{\mathrm{AQFT}} &:= S_1'; S_2'; \ldots; S_{n-k}'; S_{n-k+1}; \ldots; S_n; \\
S_i' &:= \begin{cases} q_i = H[q_i]; (q_i, q_{i+1}) = CR_2[(q_i, q_{i+1})]; \ldots; (q_i, q_{i+k-1}) = CR_k[(q_i, q_{i+k-1})]; & i \in [1, n-k] \\ q_i = H[q_i]; (q_i, q_{i+1}) = CR_2[(q_i, q_{i+1})]; \ldots; (q_i, q_n) = CR_{n-i+1}[(q_i, q_n)]; & i \in [n-k+1, n] \end{cases} \\
S_i &:= \begin{cases} S_i'; T_i & i \in [1, n-k] \\ S_i' & i \in [n-k+1, n] \end{cases} \\
T_i &:= (q_i, q_{i+k}) = CR_{k+1}[(q_i, q_{i+k})]; \ldots; (q_i, q_n) = CR_{n-i+1}[(q_i, q_n)]; \quad i \in [1, n-k]
\end{aligned}
$$

Figure 5.7: Quantum Programs S_{QFT} and S_{AQFT}

according to our rule (UT) and the fact that $\equiv$ equals $(U \otimes U) \equiv (U \otimes U)^\dagger$ for any unitary U.

For the approximate subprogram T_i, we use rule (UT-ID) to have

$$\vdash T_i \sim_{\delta_i} \textbf{skip} : \equiv \Rightarrow \equiv \quad i \in [1, n-k] \tag{5.36}$$

where $\delta_i = \|I - CR_{k+1}[(q_i, q_{i+k})] \dots CR_{n-i+1}[(q_i, q_n)]\|_\diamond/2 = \frac{1}{2}\sin\pi(2^{-k} - 2^{i-n-1})$. Then we use rule (Seq) to combine Eq. 5.36 to have

$$\vdash S_i \sim_{\delta_i} S'_i : \equiv \Rightarrow \equiv \quad (1 \leq i \leq n-k) \tag{5.37}$$

Finally, we combine Eq. 5.35 and Eq. 5.37 to have Eq. 5.32, where $\delta = \sum_{i=1}^{n-k} \delta_i \leq \frac{\pi}{2}(\frac{n-k-1}{2^k} + \frac{1}{2^n}) \leq n\pi/2^{k+1}$. If there is no approximation ($k = n$) in program S_{AQFT}, then we have $\delta = 0$ and the equivalence relation holds exactly.

We have $D([\![S_{\text{QFT}}]\!](\rho), [\![S_{\text{AQFT}}]\!](\rho)) \leq 2\delta \leq n\pi/2^k$ for any ρ by lemma 4.10. To achieve the precision δ, we need to set parameter $k = \log(n\pi/\delta) - 1$, which is consistent with [CW00]. Notice that the deviation δ is an exponential decay with respect to k, and the number of CR_m gates reduces from $(n^2-n)/2$ to $(2n-k)(k-1)/2$, as shown in Fig. 5.8. It is well-known that QFT plays a key role in factoring big numbers [Sho94]. For example, factoring n-bit RSA integers [GE21] needs at least $3n + 0.002n\lg n$ logical qubits; a reliable AQFT would significantly improve the cracking of RSA.

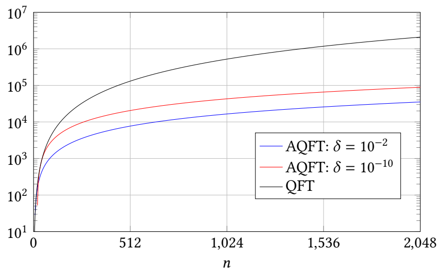

Figure 5.8: Number of CR_m gates.

AQFT Circuit proposed by Watrous

A more advanced approach to approximate QFT was proposed by Cleve and Watrous [CW00] that parallelized the phase estimation procedure [Kit96] to achieve a lower AQFT circuit depth $O(\log n)$. The alternative method of implementing QFT is shown in Fig. 5.9. The unitary V generates the Fourier basis state $|\psi_x\rangle$ defined in Eq. 5.34 without erasing the input computational basis $|x\rangle$. The unitary Add introduces additional $(k-1)n$ qubits to create $k-1$ replicas of Fourier basis state $|\psi_x\rangle$. The unitary oracle T introduces additional n qubits to compute the corresponding phase parameter $|x\rangle$ of the Fourier basis state $|\psi_x\rangle$ without erasing $|\psi_x\rangle$. Note that these oracles require additional auxiliary qubits to enable parallel execution. Still, these auxiliary qubits are not depicted in Fig. 5.9 since they are reset back to $|0\rangle$ after the computation. We encode the circuit in Fig. 5.9 as program S_{QFT}.

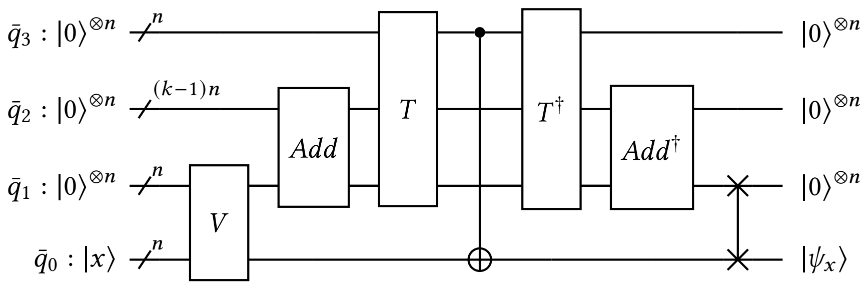

Figure 5.9: QFT circuit in [CW00]. Given a computational basis state $|x\rangle$ and corresponding Fourier basis state $|\psi_x\rangle$, unitary V performs mapping $|x\rangle|0\rangle^{\otimes n} \mapsto |x\rangle|\psi_x\rangle$, unitary Add performs mapping $|\psi_x\rangle|0\rangle^{\otimes n}\cdots|0\rangle^{\otimes n} \mapsto |\psi_x\rangle|\psi_x\rangle\cdots|\psi_x\rangle$, and unitary T performs mapping $|\psi_x\rangle\cdots|\psi_x\rangle|0\rangle^{\otimes n} \mapsto |\psi_x\rangle\cdots|\psi_x\rangle|x\rangle$.

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 [KSV02]. 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}. The Eq. 5.32 now becomes

$$S_{\mathrm{QFT}} \sim_{\delta_1+2\delta_2} S_{\mathrm{AQFT}} : \equiv_{(\bar{q}_0,\bar{q}_0')} \otimes |0\rangle\langle 0|_{aux} \Rightarrow \equiv_{(\bar{q}_0,\bar{q}_0')} \otimes |0\rangle\langle 0|_{aux} \tag{5.38}$$

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 steps are shown in Fig. 5.10, and detailed explanations follow.

Create the Fourier basis state

The computation of unitary oracle U in Eq. 5.33 can be parallelized by individually preparing every $|\mu_\theta\rangle$ in Eq. 5.34 by the following unitary

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

$$\left\{ \equiv_{(\bar{q}_0,\bar{q}'_0)} \otimes |0\rangle\langle 0|_{(\bar{q}_1,\bar{q}_2,\bar{q}_3,\bar{q}'_1,\bar{q}'_2,\bar{q}'_3,\bar{r})} \right\} // P_0$$
$$(\bar{q}_0,\bar{q}_1) := V[(\bar{q}_0,\bar{q}_1)]; \quad \sim_{\delta_1} \quad (\bar{q}'_0,\bar{q}'_1) := V'[(\bar{q}'_0,\bar{q}'_1)];$$
$$\{ (V \otimes V) P_0 (V^\dagger \otimes V^\dagger) \} // P_1$$
$$(\bar{q}_1,\bar{q}_2) := Add[(\bar{q}_1,\bar{q}_2)]; \quad \sim \quad (\bar{q}'_1,\bar{q}'_2) := Add[(\bar{q}'_1,\bar{q}'_2)];$$
$$\{ (Add \otimes Add) P_1 (Add^\dagger \otimes Add^\dagger) \} // P_2$$
$$(\bar{q}_1,\bar{q}_2,\bar{q}_3) = T[(\bar{q}_1,\bar{q}_2,\bar{q}_3)]; \quad \sim_{\delta_2} \quad (\bar{q}'_1,\bar{q}'_2,\bar{q}'_3) = T'[(\bar{q}'_1,\bar{q}'_2,\bar{q}'_3)];$$
$$\{ (T \otimes T) P_2 (T^\dagger \otimes T^\dagger) \} // P_3$$
$$(\bar{q}_3,\bar{q}_0) := CNOT(\bar{q}_3,\bar{q}_0); \quad \sim \quad (\bar{q}_3,\bar{q}_0) := CNOT(\bar{q}'_3,\bar{q}'_0);$$
$$\{ (CNOT \otimes CNOT) P_3 (CNOT \otimes CNOT)^\dagger \} // P_4$$
$$(\bar{q}_1,\bar{q}_2,\bar{q}_3) := T^\dagger[(\bar{q}_1,\bar{q}_2,\bar{q}_3)]; \quad \sim_{\delta_2} \quad (\bar{q}'_1,\bar{q}'_2,\bar{q}'_3,\bar{r}) := T^\dagger[(\bar{q}'_1,\bar{q}'_2,\bar{q}'_3,\bar{r})];$$
$$\{ (T^\dagger \otimes T^\dagger) P_4 (T \otimes T) \} // P_5$$
$$(\bar{q}_1,\bar{q}_2) := Add^\dagger[(\bar{q}_1,\bar{q}_2)]; \quad \sim \quad (\bar{q}'_1,\bar{q}'_2) := Add^\dagger[(\bar{q}'_1,\bar{q}'_2)];$$
$$\{ (Add^\dagger \otimes Add^\dagger) P_5 (Add \otimes Add) \} // P_6$$
$$(\bar{q}_0,\bar{q}_1) := SWAP(\bar{q}_0,\bar{q}_1); \quad \sim \quad (\bar{q}'_0,\bar{q}'_1) := SWAP(\bar{q}'_0,\bar{q}'_1);$$
$$\{ (SWAP \otimes SWAP) P_6 (SWAP^\dagger \otimes SWAP^\dagger) \} // P_7 = P_0$$

Figure 5.10: 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.

in [CW00], 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 by a sequence of unitaries,

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

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. 5.11 in [CW00] represents the circuit of unitary $Q_{4,i}$ on $|x\rangle$.

Similar to the approximation in [Cop02], 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

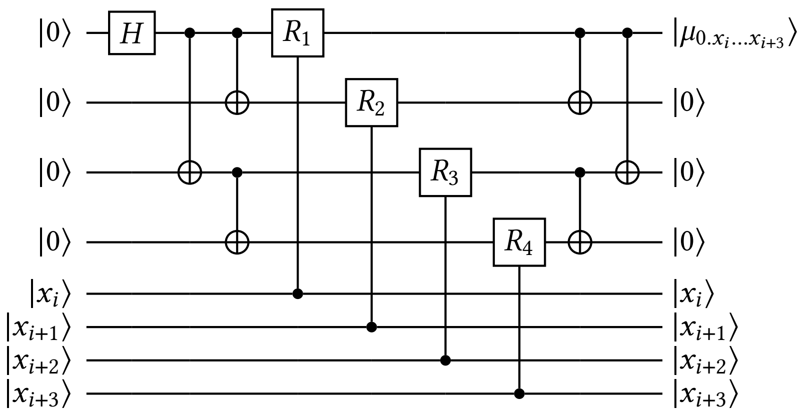

Figure 5.11: Circuit for oracle $Q_{4,i}$ on state $|x_1 \dots x_n\rangle$. Qubits $x_1 \dots x_{i-1}$ and $x_{i+4} \dots x_n$ are not used and ignored.

$1 \le t < t' \le n$. The approximation can be modeled by the following judgment

$$\begin{gathered}\vdash (\bar{q}, \bar{p}) = Q_{t,i}[(\bar{q}, \bar{p})] \sim_{\delta(t,t')} (\bar{q}', \bar{p}') = Q_{t',i}[(\bar{q}', \bar{p}')] : \\ |0\rangle\langle 0|_{(\bar{q},\bar{q}')} \otimes \equiv_{(\bar{p},\bar{p}')} \Rightarrow \equiv_{(\bar{q}[1],\bar{q}'[1])} \otimes |0\rangle\langle 0|_{(\bar{q}[2,n],\bar{q}'[2,n])} \otimes \equiv_{(\bar{p},\bar{p}')}\end{gathered} \tag{5.39}$$

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 $\bar{q}$.

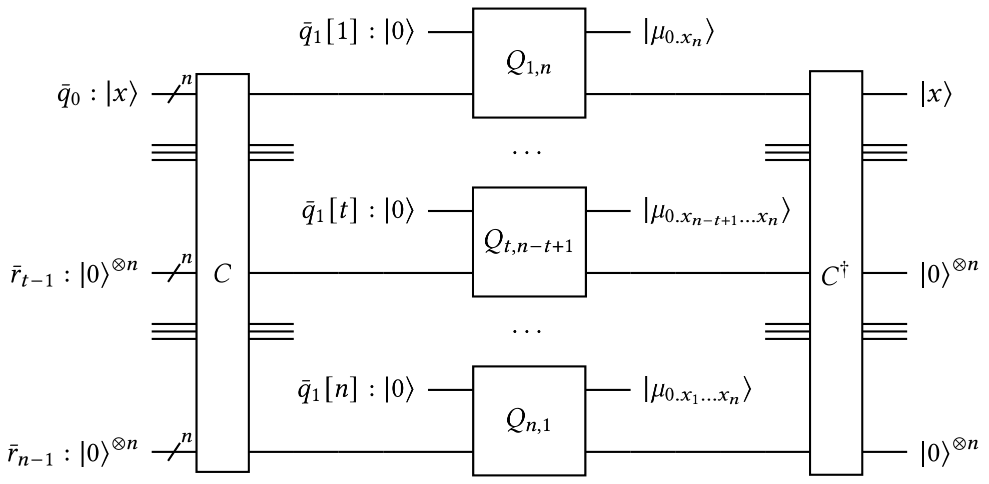
$\bar{q}_1[1] : |0\rangle$
$Q_{1,n}$
$|\mu_{0.x_n}\rangle$
$\bar{q}_0 : |x\rangle$
n
$|x\rangle$
$\bar{q}_1[t] : |0\rangle$
$Q_{t,n-t+1}$
$|\mu_{0.x_{n-t+1} \dots x_n}\rangle$
$\bar{r}_{t-1} : |0\rangle^{\otimes n}$
C
$C^\dagger$
$|0\rangle^{\otimes n}$
$\bar{q}_1[n] : |0\rangle$
$Q_{n,1}$
$|\mu_{0.x_1 \dots x_n}\rangle$
$\bar{r}_{n-1} : |0\rangle^{\otimes n}$
$|0\rangle^{\otimes n}$

Figure 5.12: Circuit for oracle V. Given a computational basis state $|x\rangle = |x_1 \dots x_n\rangle$, 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 $|0\rangle\, |x\rangle \mapsto |\mu_{0.x_i \dots x_{i+t-1}}\rangle\, |x\rangle$.

Fig. 5.12 illustrates the circuit of the oracle V. 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. 5.12 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. 5.12 except that $Q_{t,i}$ is approximated by $Q_{k,i}$, where k $(0 < k < t \le n)$ still denotes the number of significant phase shift gates mentioned in Fig. 5.6. Specifically, oracles V and V' can be represented as the following sequence of gates,

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

where auxiliary register $\bar{r} = \{\bar{r}_1,\dots,\bar{r}_{n-1}\}$ contains $n-1$ registers $\bar{r}_i$ that are initialized with $|0\rangle^{\otimes n}$.

Based on judgement 5.39, we have the following judgment

$$\vdash (\bar{q}_0,\bar{q}_1) = V[(\bar{q}_0,\bar{q}_1)] \sim_{\delta_1} (\bar{q}'_0,\bar{q}'_1) = V'[(\bar{q}'_0,\bar{q}'_1] : P_0 \Rightarrow P_1 \tag{5.40}$$

where $\delta_1 = \sum_{i=k+1}^{n} \delta(k,i) = \frac{1}{2}\sum_{i=k+1}^{n} \sin\pi(2^{-k}-2^{-i}) \le n\pi 2^{-k-1}$. Notice that every register $\bar{r}_i$ in Fig. 5.12 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 the Fourier basis state

We provide a brief overview of the functionality of the oracle *Add* as described in [CW00]. 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

$$|x_1\rangle|x_2\rangle\cdots|x_k\rangle \longrightarrow |x_1\rangle|x_2-x_1\rangle\cdots|x_k-x_{k-1}\rangle$$

to obtain $|\psi_x\rangle|\psi_x\rangle\dots|\psi_x\rangle$. Reversely, we can use prefix addition

$$|x_1\rangle|x_2\rangle\cdots|x_k\rangle \longrightarrow |x_1\rangle|x_1+x_2\rangle\cdots|x_1+x_2+\cdots+x_k\rangle$$

to eliminate the duplicates of the Fourier basis state. A $\log(k)$-depth tree of 3-2 adders can 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*† respectively. Then we have the following judgment

$$\vdash (\bar{q}_1, \bar{q}_2) := Add[(\bar{q}_1, \bar{q}_2)] \sim (\bar{q}'_1, \bar{q}'_2) := Add[(\bar{q}'_1, \bar{q}'_2)] : P_1 \Rightarrow P_2 \tag{5.41}$$

$$\vdash (\bar{q}_1, \bar{q}_2) := Add^\dagger[(\bar{q}_1, \bar{q}_2)] \sim (\bar{q}'_1, \bar{q}'_2) := Add^\dagger[(\bar{q}'_1, \bar{q}'_2)] : P_5 \Rightarrow P_6 \tag{5.42}$$

by rule (Ut).

Estimate the phase of a Fourier state

The key to this step is based on the idea [AKN98b] that quantum measurement can be simulated by unitaries with the help of ancillary qubits.

As shown in Fig 5.9, 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 follows,

$$\begin{aligned} |x\rangle\,|\psi_x\rangle \cdots |\psi_x\rangle\,|0\rangle^{\otimes n} &\xrightarrow{T} |x\rangle\,|\psi_x\rangle \cdots |\psi_x\rangle\,|x\rangle \xrightarrow{CNOT} |0\rangle^{\otimes n}\,|\psi_x\rangle \cdots |\psi_x\rangle\,|x\rangle \\ &\xrightarrow{T^\dagger} |0\rangle^{\otimes n}\,|\psi_x\rangle \cdots |\psi_x\rangle\,|0\rangle^{\otimes n} \end{aligned}$$

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, [CW00] parallelized the phase estimation procedure proposed by [Kit96]. 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 $|m_{(x,i)}\rangle$,

$$|m_{(x,i)}\rangle\,|0\rangle \xrightarrow{f} |m_{(x,i)}\rangle\,|x'_i\rangle$$

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 5.1. [CW00] *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 $|0\rangle$. Unitaries 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 5.1, we would have

$$\begin{aligned}&\vdash(\bar{q}_1,\bar{q}_2,\bar{q}_3)=T[(\bar{q}_1,\bar{q}_2,\bar{q}_3)]\sim_{\delta_2}(\bar{q}_1',\bar{q}_2',\bar{q}_3',\bar{r})=T'[(\bar{q}_1',\bar{q}_2',\bar{q}_3',\bar{r})]:P_2\;\Rightarrow\;P_3\\&\vdash(\bar{q}_1,\bar{q}_2,\bar{q}_3)=T^{\dagger}[(\bar{q}_1,\bar{q}_2,\bar{q}_3)]\sim_{\delta_2}(\bar{q}_1',\bar{q}_2',\bar{q}_3',\bar{r})=T'^{\dagger}[(\bar{q}_1',\bar{q}_2',\bar{q}_3',\bar{r})]:P_4\Rightarrow P_5\end{aligned}\tag{5.43}$$

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

Conclusion

We can use rule (Seq) to sum up all judgments in Fig. 5.10 to get Eq. 5.38,

$$S_{\text{QFT}}\sim_{\delta_1+2\delta_2}S_{\text{AQFT}}:\equiv_{(\bar{q}_0,\bar{q}_0')}\otimes|0\rangle\langle 0|_{aux}\Rightarrow\equiv_{(\bar{q}_0,\bar{q}_0')}\otimes|0\rangle\langle 0|_{aux}\tag{5.44}$$

where $\delta=\delta_1+\delta_2=n\pi 2^{-k-1}+2ne^{-k/8}$.

Chapter 6

Summary

Numerous works have delved into the formal verification of quantum programs since the emergence of quantum programming languages. This thesis addresses the correctness of quantum programs from two distinct perspectives. The first approach concentrates on bug detection as a means to enhance correctness. It involves an extension of the classical incorrectness logic [O'H19], marking an initial step towards formulating an incorrectness logic designed for quantum programs. The incorrectness triple for quantum programs is constructed by introducing novel concepts of underapproximation and reachability analysis. We establish a sound and complete proof system for bug detection in quantum programs. In another approach, relational reasoning is employed to improve the correctness of quantum programs. We introduce approximate quantum coupling as a fundamental tool for exploring approximate relational properties between quantum programs, addressing the open question from [BHY$^+$19] for robust reasoning. The efficacy of both approaches is demonstrated through a series of case studies in Chapter 5. To conclude this thesis, we review pertinent related work and outline prospective directions for future studies.

Related Work

Quantum 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 [ZYY19, LZY$^+$20, YP21], and quantum relational Hoare logics (qRHL) for reasoning about equivalence between two quantum programs [BHY$^+$19, Unr19]. Compared with other quantum predicates such as observables [DP06b, Yin12], 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'H19] "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'H19] for classical programs mainly inspires our work. We integrate the spirit of classical incorrectness logic of [O'H19] and generalize to the quantum settings. This result is partly inspired by the use of projections as assertions for testing and debugging quantum programs [LZY$^+$20]. There are other practical debugging tools for quantum programs, such as the one employing assertions based on statistical tests on classical observations [HM19b]. 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.

Approximate Quantum Relational Hoare logic

Quantitative robustness reasoning One work [HHZ+19] develops semantics for erroneous quantum while programs and logic to prove quantum robustness between an ideal program and a noisy one. They define that a noisy program S' is ϵ-robust with respect to (Q, λ) if any input state ρ satisfies the quantum predicate Q to degree λ, the distance between the ideal program S and the noisy program S' is bounded by ϵ. They extend the diamond norm to the so-called (Q, λ)-diamond norm to precisely describe the distance between program S and S' when input states satisfy a quantum predicate Q to degree ϵ. Another work [ZYY19] derives applied quantum Hoare logic by employing projection as predicates and reasons about the robustness of quantum programs, i.e., error bounds of outputs. The approximate satisfaction in [ZYY19] is based on introducing a convex set (P, ϵ) of any state ρ such that there exits state $\sigma \models P$ satisfying $\mathrm{Tr}(\rho) = \mathrm{Tr}(\sigma)$ and $D(\rho, \sigma) \leq \epsilon$, which shares similar characteristics with our approxiamte predicate discussed in Sec. 4.5.

These two works consider 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 [HHZ+19, ZYY19], 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 of [HHZ+19, ZYY19] focus on studying the robustness of quantum programs, i.e., equivalence or closeness. In particular, the Hilbert spaces of the compared programs in these two works must have the same dimension. Our choice of the 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 [HHZ+19, ZYY19] discuss programs with the same syntax statement. Our one-side rules can track relational properties for different statements, for instance, a unitary statement and a while statement, in repeat until success.

Quantum relational Hoare logics. Our work is most inspired by the quantum relational Hoare logics recently proposed by [Unr19, BHY$^+$19, LU21]. In particular, [BHY$^+$19] suggests that casting approximate reasoning into the general framework of relational quantum Hoare logic remains open. Generally, two quantum programs do not share the same probabilities for taking different paths or outcomes during execution. Under those circumstances, one can not even find exact quantum couplings since quantum coupling exists only for partial density operators with the same 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 [BKOZB13], even from a technical point of view: aqRHL employs projective predicates [BN36] over the joint systems of the programs, a natural quantum counterpart of binary relations, the predicates used in apRHL.

Future Work

Quantum Incorrectness Logic

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. Similar to classical incorrectness logic, when presented with a quantum incorrectness triple $[P]S[Q]$, the problem is whether the set P of input quantum states is included in the set Q of the output state of program S. It is interesting to explore the creation of a representation that succinctly characterizes sets of quantum states and the corresponding transformers for depicting quantum operations on this representation. For example, a very recent work [CCL+23] introduced a novel approach using tree automata to craft an assertion checker. Its algebraic representation of quantum states proves effective in sidestepping the inaccuracies associated with working with floating-point numbers.

Supporting quantum abstraction or local reasoning. Readers may notice that the Unitary rule has no advantage over the direct multiplication of density matrix, which is the strongest postcondition computation and requires matrices of size exponential in the number of qubits. To avoid direct full-blown quantum simulation, it is possible to apply quantum abstraction or local reasoning to improve the effectiveness of our logic. On the one hand, we may develop a similar abstract operation [YP21] that preserved the general Galois connection to shrink the size of matrix multiplication at the cost of losing some information. On the other hand, a promising idea is to combine the recently developed quantum separation logic [ZBH+21] and incorrectness separation logic [RBD+20] for classical programs, that is to determine the extent to which local reasoning is feasible for quantum programs from the incorrect point of view. Compared with classical local reasoning, 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.

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

Supporting quantum noise and quantum control We do not mention the quantum noise in our modified language (Def. 3.1). It would be more practical to incorporate noise estimation such as Gleipnir [TSY+21] and [HHZ+19] into the incorrectness logic. Another limitation is that we consider quantum programs with classical controls rather than quantum controls. 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.

Approximate Quantum Relational Hoare logic

Quantum predicates as observables. Projections have been initially utilized as predicates to establish quantum Hoare logic for reasoning about the correctness of quantum programs in [ZYY19]. They were also adopted in the quantum relational Hoare logics [BHY+19, Unr19] to justify equivalence between two quantum programs. In our robust relational reasoning, we also employ projective predicates instead of observables [DP06b]. Generally speaking, projective predicates are less expressive than observables, but they are more user-friendly for the analyzer to provide suitable propositions. Projective predicates simplify the verification of judgments by transforming them into boolean inferences. Our work needs to investigate whether observables may better characterize approximate reasoning. We may expand the judgment defined in 4.12 in the following manner.

Definition 6.1 (General Validity). The judgement $\vDash S_1 \sim_\delta S_2 : A \Rightarrow B$ holds if for any lifting $\rho_1 \sim_A \rho_2$ with ρ being the corresponding witness, then there exits an witness σ for lifting $[\![S_1]\!](\rho_1) \sim_B^\delta [\![S_2]\!](\rho_2)$ such that

$$\mathrm{Tr}(A\rho) \leq \mathrm{Tr}(B\sigma) + \mathrm{Tr}(\rho) - \mathrm{Tr}(\sigma)$$

where A and B are observables.

Quantum differential privacy. In our earlier discussions, we mentioned that our work is an extension of the relational Hoare logic [BHY+19] to an approximate version inspired by reasoning for differential privacy [DMNS06, BKOZB13]. To capture the exponential and Laplacian mechanisms of differential privacy, a skew-parameterized α-distance

$$\Delta_\alpha(\mu_1, \mu_2) = \max_{f:A\to[0,1]} \Delta_\alpha(\mu_1 f, \mu_2 f)$$

is introduced to measure the distance between two distributions μ_1 and μ_2 in $\mathcal{D}(A)$. In contrast, our work uses the standard trace distance to measure the difference between quantum states in a general setting. In [HRF22], Hirche et al. [SW13] introduced the quantum hockey-stick divergence $E_r(\rho, \sigma)$

$$E_r(\rho, \sigma) = \mathcal{D}(\rho, \alpha\sigma) + \frac{1}{2}(1 - \alpha)$$

to reason about quantum differential privacy introduced in [ZY17]. It might be possible to design a specific relational logic for quantum differential privacy by introducing a symmetric α distance

$$\Delta_\alpha(\rho, \sigma) = \max_{0 \le P \le I} \{\mathrm{Tr}(P(\rho - \alpha\sigma)), \mathrm{Tr}(P(\sigma - \alpha\rho)), 0\}$$

based on the hockey-stick divergence. Similarly, we can maintain the triangle inequality

$$\Delta_{\alpha\alpha'}(\rho_1, \rho_3) \le \Delta_{\alpha'}(\rho_2, \rho_3) + \alpha'\Delta_\alpha(\rho_1, \rho_2)$$
$$\Delta_{\alpha\alpha'}(\rho_1, \rho_3) \le \Delta_\alpha(\rho_1, \rho_2) + \alpha\Delta_{\alpha'}(\rho_2, \rho_3)$$

and contractility

$$\Delta_\alpha(\mathcal{E}(\rho_1), \mathcal{E}(\rho_2)) \le \Delta_\alpha(\rho_1, \rho_2)$$

for any trace-preserving quantum operations $\mathcal{E}$, which are essential for designing useful relational proof rules.

Applications for hybrid system. Our logic could be more useful if we extend our theory to the hybrid system, i.e., programs with quantum and classical variables. Quantum-classical hybrid systems allow for the exploitation of quantum advantages while leveraging the existing classical computing infrastructure. A unified language of both quantum and classical effects may bring advantages in hybrid program analysis [VLRH23]. 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. Last but not least, it is interesting to incorporate recently developed tools such as quantum abstract interpretation [YP21] and quantum separation logic [ZBH+21] to design over-approximation techniques [Yan07]. Another interesting technique is the Context-Free-Language Ordered Binary Decision Diagrams

[SCR23], which may serve as a backend representation and manipulation technique in studying quantum Hoare logics.

Bibliography

[AAB+19] 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. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. doi:10.1038/s41586-019-1666-5.

[AFD+12] Ali J Abhari, Arvin Faruque, Mohammad J Dousti, Lukas Svec, Oana Catu, Amlan Chakrabati, Chen-Fu Chiang, Seth Vanderwilt, John Black, and Fred Chong. Scaffold: Quantum programming language. Technical report, Princeton Univ NJ Dept of Computer Science, 2012.

[AKN98a] Dorit Aharonov, Alexei Kitaev, and Noam Nisan. Quantum circuits with mixed states, 1998. doi:10.48550/ARXIV.QUANT-PH/9806029.

[AKN98b] Dorit Aharonov, Alexei Kitaev, and Noam Nisan. Quantum circuits with mixed states. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 20–30, New York, NY, USA, 1998. Association for Computing Machinery. doi:10.1145/276698.276708.

[BALR20] Sahar Badihi, Faridah Akinotcho, Yi Li, and Julia Rubin. 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*, ESEC/FSE 2020, pages 13–

24, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3368089.3409757.

[Ben04] Nick Benton. Simple relational correctness proofs for static analyses and program transformations. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '04, pages 14–25, New York, NY, USA, 2004. Association for Computing Machinery. doi:10.1145/964001.964003.

[BGOS18] Sam Blackshear, Nikos Gorogiannis, Peter W. O'Hearn, and Ilya Sergey. Racerd: Compositional static race detection. *Proc. ACM Program. Lang.*, 2(OOPSLA), oct 2018. doi:10.1145/3276514.

[BGZB09] Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. Formal certification of code-based cryptographic proofs. In *Proceedings of the 36th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '09, pages 90–101, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1480881.1480894.

[BHY$^+$19] Gilles Barthe, Justin Hsu, Mingsheng Ying, Nengkun Yu, and Li Zhou. Relational proofs for quantum programs. *Proc. ACM Program. Lang.*, 4(POPL), Dec 2019. doi:10.1145/3371089.

[BJ04] Olivier Brunet and Philippe Jorrand. Dynamic quantum logic for quantum programs. *International Journal of Quantum Information*, 02(01):45–54, 2004. doi:10.1142/S0219749904000067.

[BKOZB13] Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella-Béguelin. Probabilistic relational reasoning for differential privacy. *ACM Trans. Program. Lang. Syst.*, 35(3), Nov 2013. doi:10.1145/2492061.

[BN36] Garrett Birkhoff and John Von Neumann. The logic of quantum mechanics. *Annals of Mathematics*, 37(4):823–843, 1936. URL `http://www.jstor.org/stable/1968621`.

[BRS15] Alex Bocharov, Martin Roetteler, and Krysta M. Svore. Efficient synthesis of universal repeat-until-success quantum circuits. *Phys. Rev. Lett.*, 114:080502, Feb 2015. doi:10.1103/PhysRevLett.114.080502.

[BS06] Alexandru Baltag and Sonja Smets. LQP: the dynamic logic of quantum information. *Mathematical Structures in Computer Science*, 16(3):491–525, 2006. doi:10.1017/S0960129506005299.

[BTT82] J.A. Bergstra, J. Tiuryn, and J.V. Tucker. Floyd's principle, correctness theories and program equivalence. *Theoretical Computer Science*, 17(2):113–149, 1982. doi:https://doi.org/10.1016/0304-3975(82)90001-9.

[CCL⁺23] Yu-Fang Chen, Kai-Min Chung, Ondřej Lengál, Jyun-Ao Lin, Wei-Lun Tsai, and Di-De Yen. An automata-based framework for verification and bug hunting in quantum circuits. *Proc. ACM Program. Lang.*, 7(PLDI), jun 2023. doi:10.1145/3591270.

[CE79] G. Cousineau and P. Enjalbert. Program equivalence and provability. In Jiří Bečvář, editor, *Mathematical Foundations of Computer Science*, pages 237–245, Berlin, Heidelberg, 1979. Springer Berlin Heidelberg. doi:10.1007/3-540-09526-8_20.

[CJAA⁺21] 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. Openqasm 3: A broader and deeper quantum assembly language. *ACM Transactions on Quantum Computing*, dec 2021. doi:10.1145/3505636. Just Accepted.

[CMS06] R. Chadha, P. Mateus, and A. Sernadas. Reasoning about imperative quantum programs. *Electronic Notes in Theoretical Computer Science*, 158:19 – 39, 2006. doi:https://doi.org/10.1016/j.entcs.2006.04.003. Proceedings of the 22nd Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXII).

[Cop02] Don Coppersmith. An approximate fourier transform useful in quantum factoring, 2002. doi:10.48550/arxiv.quant-ph/0201067.

[CPSA19] Berkeley Churchill, Oded Padon, Rahul Sharma, and Alex Aiken. Semantic program alignment for equivalence checking. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, page 1027–1040, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3314221.3314596.

[CW00] R. Cleve and J. Watrous. Fast parallel circuits for the quantum fourier transform. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 526–536, Redondo Beach, CA, USA, 02 2000. IEEE Computer Society. doi:10.1109/SFCS.2000.892140.

[Dev21] Cirq Developers. Cirq, August 2021. doi:10.5281/zenodo.5182845. See full list of authors on Github: https://github.com/ quantumlib/Cirq/graphs/-contributors.

[DFLO19] Dino Distefano, Manuel Fähndrich, Francesco Logozzo, and Peter W. O'Hearn. Scaling static analyses at facebook. *Commun. ACM*, 62(8):62–70, jul 2019. doi:10.1145/3338112.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, page 265–284, Berlin, Heidelberg, 2006. Springer-Verlag. doi:10.1007/11681878_14.

[DP06a] Ellie D'hondt and Prakash Panangaden. Quantum weakest preconditions. 16(3):429–451, June 2006. doi:10.1017/S0960129506005251.

[DP06b] Ellie D'hondt and Prakash Panangaden. Quantum weakest preconditions. *Mathematical. Structures in Comp. Sci.*, 16(3):429–451, jun 2006. doi:10.1017/S0960129506005251.

[dVK11] Edsko de Vries and Vasileios Koutavas. Reverse Hoare logic. In *Software Engineering and Formal Methods*, pages 155–171, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[EHK04] Mark Ettinger, Peter Høyer, and Emanuel Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *Information Processing Letters*, 91(1):43–48, 2004. doi:https://doi.org/10.1016/j.ipl.2004.01.024.

[Flo93] Robert W. Floyd. *Assigning Meanings to Programs*, pages 65–81. Springer Netherlands, Dordrecht, 1993. doi:10.1007/978-94-011-1793-7_4.

[GE21] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, April 2021. doi:10.22331/q-2021-04-15-433.

[GLR+13] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. Quipper: A scalable quantum programming language. *SIGPLAN Not.*, 48(6):333–342, jun 2013. doi:10.1145/2499370.2462177.

[GOS19] Nikos Gorogiannis, Peter W. O'Hearn, and Ilya Sergey. A true positives theorem for a static race detector. *Proc. ACM Program. Lang.*, 3(POPL), jan 2019. doi:10.1145/3290370.

[Gra05] Jonathan Grattage. A functional quantum programming language. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer*

Science, LICS '05, pages 249–258, USA, 2005. IEEE Computer Society. doi:10.1109/LICS.2005.1.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery. doi:10.1145/237814.237866.

[HHL09] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009. doi:10.1103/PhysRevLett.103.150502.

[HHZ$^+$19] Shih-Han Hung, Kesha Hietala, Shaopeng Zhu, Mingsheng Ying, Michael Hicks, and Xiaodi Wu. Quantitative robustness analysis of quantum programs. *Proc. ACM Program. Lang.*, 3(POPL):31:1–31:29, 2019. doi:10.1145/3290344.

[HM19a] Yipeng Huang and Margaret Martonosi. QDB: From Quantum Algorithms Towards Correct Quantum Programs. In Titus Barik, Joshua Sunshine, and Sarah Chasins, editors, *9th Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU 2018)*, volume 67 of *OpenAccess Series in Informatics (OASIcs)*, pages 4:1–4:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/OASIcs.PLATEAU.2018.4.

[HM19b] Yipeng Huang and Margaret Martonosi. Statistical assertions for validating patterns and finding bugs in quantum programs. In *Proceedings of the 46th International Symposium on Computer Architecture*, ISCA '19, page 541–553, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3307650.3322213.

[Hoa69] C. A. R. Hoare. An axiomatic basis for computer programming. 12(10):576–580, October 1969. doi:10.1145/363235.363259.

[HRF22] Christoph Hirche, Cambyse Rouzé, and Daniel Stilck França. Quantum differential privacy: An information theory perspective, 2022. doi:10.48550/ARXIV.2202.10717.

[HSM+20] Bettina Heim, Mathias Soeken, Sarah Marshall, Christopher E. Granade, Martin Roetteler, Alan Geller, Matthias Troyer, and Krysta Marie Svore. Quantum programming languages. *Nature Reviews Physics*, 2:709–722, Dec 2020. doi:10.1038/s42254-020-00245-7.

[Hsu17] Justin Hsu. Probabilistic couplings for probabilistic reasoning, 2017, arXiv:1710.09951.

[JPK+15] Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T. Chong, and Margaret Martonosi. Scaffcc: Scalable compilation and analysis of quantum programs. *Parallel Computing*, 45(C):2–17, jun 2015. doi:https://doi.org/10.1016/j.parco.2014.12.001.

[Kak09] Yoshihiko Kakutani. A logic for formal verification of quantum programs. In *Proceedings of the 13th Asian Conference on Advances in Computer Science: Information Security and Privacy*, ASIAN'09, pages 79–93, Berlin, Heidelberg, 2009. Springer-Verlag. doi:10.1007/978-3-642-10622-4_7.

[Kit96] Alexei Y. Kitaev. Quantum measurements and the abelian stabilizer problem. *Electron. Colloquium Comput. Complex.*, TR96-003, 1996. URL `https://eccc.weizmann.ac.il/eccc-reports/1996/TR96-003/index.html`.

[KSV02] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, USA, 2002. doi:https://doi.org/10.1090/gsm/047.

[KTL09] Sudipta Kundu, Zachary Tatlock, and Sorin Lerner. Proving optimizations correct using parameterized program equivalence. *SIGPLAN Not.*, 44(6):327–337, jun 2009. doi:10.1145/1543135.1542513.

[LBZ20] Ji Liu, Gregory T. Byrd, and Huiyang Zhou. *Quantum Circuits for Dynamic Runtime Assertions in Quantum Computation*, page 1017–1030. Association for Computing Machinery, New York, NY, USA, 2020. URL `https://doi.org/10.1145/3373376.3378488`.

[LMR14] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, jul 2014. doi:10.1038/nphys3029.

[LR15] Dorel Lucanu and Vlad Rusu. Program equivalence by circular reasoning. *Form. Asp. Comput.*, 27(4):701–726, jul 2015. doi:10.1007/s00165-014-0319-6.

[LSH15] Shuvendu K. Lahiri, Rohit Sinha, and Chris Hawblitzel. Automatic root-causing for program equivalence failures in binaries. In Daniel Kroening and Corina S. Păsăreanu, editors, *Computer Aided Verification*, pages 362–379, Cham, 2015. Springer International Publishing. doi:10.1007/978-3-319-21690-4_21.

[LU21] Yangjia Li and Dominique Unruh. Quantum Relational Hoare Logic with Expectations. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 136:1–136:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.136.

[LZM+22] J. Luo, P. Zhao, Z. Miao, S. Lan, and J. Zhao. A comprehensive study of bug fixes in quantum programs. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 1239–1246, Los Alamitos, CA, USA, mar 2022. IEEE Computer Society. doi:10.1109/SANER53432.2022.00147.

[LZY+20] Gushu Li, Li Zhou, Nengkun Yu, Yufei Ding, Mingsheng Ying, and Yuan Xie. Projection-based runtime assertions for testing and debugging quantum programs. *Proc. ACM Program. Lang.*, 4(OOPSLA), nov 2020. doi:10.1145/3428218.

[MZW+16] Jiang Ming, Fangfang Zhang, Dinghao Wu, Peng Liu, and Sencun Zhu. Deviation-based obfuscation-resilient program equivalence checking with application to software plagiarism detection. *IEEE Transactions on Reliability*, 65(4):1647–1664, 2016. doi:10.1109/TR.2016.2570554.

[NC11] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, New York, 10th edition, 2011. doi:10.1017/CBO9780511976667.

[Nec00] George C. Necula. Translation validation for an optimizing compiler. *SIGPLAN Not.*, 35(5):83–94, May 2000. doi:10.1145/358438.349314.

[NPPŻ18] Ion Nechita, Zbigniew Puchała, Łukasz Pawela, and Karol Życzkowski. Almost all quantum channels are equidistant. *Journal of Mathematical Physics*, 59(5):052201, 2018. doi:10.1063/1.5019322.

[O'H19] Peter W. O'Hearn. Incorrectness logic. *Proc. ACM Program. Lang.*, 4(POPL), dec 2019. doi:10.1145/3371078.

[Per08a] Simon Perdrix. A hierarchy of quantum semantics. *Electron. Notes Theor. Comput. Sci.*, 192(3):71–83, 2008. doi:10.1016/j.entcs.2008.10.028.

[Per08b] Simon Perdrix. Quantum entanglement analysis based on abstract interpretation. In *Proceedings of the 15th International Symposium on Static Analysis*, volume 5079 of *SAS '08*, page 270–282, Berlin, Heidelberg, 2008. Springer-Verlag. doi:10.1007/978-3-540-69166-2_18.

[Pit97] Andrew Pitts. *Operationally-Based Theories of Program Equivalence*, pages 241–298. Publications of the Newton Institute. Cambridge University Press, Cambridge, 1997. doi:10.1017/CBO9780511526619.007.

[PP22] Matteo Paltenghi and Michael Pradel. Bugs in quantum computing platforms: An empirical study. *Proc. ACM Program. Lang.*, 6(OOPSLA1), apr 2022. doi:10.1145/3527330.

[Pre18] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, Aug 2018. doi:10.22331/q-2018-08-06-79.

[PS14] Adam Paetznick and Krysta M. Svore. Repeat-until-success: Non-deterministic decomposition of single-qubit unitaries. *Quantum Info. Comput.*, 14(15–16):1277–1301, nov 2014. doi:10.26421/QIC14.15-16-2.

[PW09] Marco Piani and John Watrous. All entangled states are useful for channel discrimination. *Phys. Rev. Lett.*, 102:250501, Jun 2009. doi:10.1103/PhysRevLett.102.250501.

[RBD$^+$20] Azalea Raad, Josh Berdine, Hoang-Hai Dang, Derek Dreyer, Peter O'Hearn, and Jules Villard. 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*, page 225–252, Berlin, Heidelberg, 2020. Springer-Verlag. doi:10.1007/978-3-030-53291-8_14.

[RS16] Neil J. Ross and Peter Selinger. Optimal ancilla-free clifford+t approximation of z-rotations. *Quantum Info. Comput.*, 16(11-12):901–953, sep 2016. doi:10.26421/QIC16.11-12-1.

[SCR23] Meghana Sistla, Swarat Chaudhuri, and Thomas Reps. Cflobdds: Context-free-language ordered binary decision diagrams, 2023, arXiv:2211.06818.

[SCZ16] Robert S. Smith, Michael J. Curtis, and William J. Zeng. A practical quantum instruction set architecture, 2016, arXiv:1608.03355.

[Sel04a] Peter Selinger. Towards a quantum programming language. *Mathematical. Structures in Computer Science*, 14(4):527–586, August 2004. doi:10.1017/S0960129504004256.

[Sel04b] Peter Selinger. Towards a quantum programming language. *Mathematical. Structures in Comp. Sci.*, 14(4):527–586, aug 2004. doi:10.1017/S0960129504004256.

[SGT$^+$18] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. Q#: Enabling scalable quantum computing and development with a high-level dsl. In *Proceedings of the Real World Domain Specific Languages Workshop 2018*, RWDSL2018, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3183895.3183901.

[Sho94] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94, pages 124–134, USA, 1994. IEEE Computer Society. doi:10.1109/SFCS.1994.365700.

[SW13] Naresh Sharma and Naqueeb Ahmad Warsi. Fundamental bound on the reliability of quantum information transmission. *Phys. Rev. Lett.*, 110:080501, Feb 2013. doi:10.1103/PhysRevLett.110.080501.

[Tra17] Andreas Trabesinger. Quantum computing: towards reality. *Nature*, 543:S1, Mar 2017. doi:10.1038/543S1a.

[TSY+21] Runzhou Tao, Yunong Shi, Jianan Yao, John Hui, Frederic T. Chong, and Ronghui Gu. Gleipnir: Toward practical error analysis for quantum programs. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021, page 48–64, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3453483.3454029.

[Unr19] Dominique Unruh. Quantum relational Hoare logic. *Proc. ACM Program. Lang.*, 3(POPL), Jan 2019. doi:10.1145/3290346.

[VLRH23] Finn Voichick, Liyi Li, Robert Rand, and Michael Hicks. Qunity: A unified language for quantum and classical computing. *Proc. ACM Program. Lang.*, 7(POPL), jan 2023. doi:10.1145/3571225.

[Wat13] John Watrous. Simpler semidefinite programs for completely bounded norms. *Chicago Journal of Theoretical Computer Science*, 2013:8, 2013. URL `http://cjtcs.cs.uchicago.edu/articles/2013/8/contents.html`.

[Wat18] John Watrous. *The Theory of Quantum Information*. Cambridge University Press, Cambridge, 2018. doi:10.1017/9781316848142.

[WFF13] Westley Weimer, Zachary P. Fry, and Stephanie Forrest. Leveraging program equivalence for adaptive program repair: Models and first results. In *2013 28th IEEE/ACM International Conference on Automated Software*

Engineering (ASE), ASE '13, pages 356–366, Silicon Valley, CA, USA, 2013. IEEE Press. doi:10.1109/ASE.2013.6693094.

[WVMN19] Robert Wille, Rod Van Meter, and Yehuda Naveh. IBM's qiskit tool chain: Working with and developing for real quantum computers. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1234–1240, 2019. doi:10.23919/DATE.2019.8715261.

[Yan07] Hongseok Yang. Relational separation logic. *Theoretical Computer Science*, 375(1):308–334, 2007. doi:https://doi.org/10.1016/j.tcs.2006.12.036. Festschrift for John C. Reynolds's 70th birthday.

[Yin12] Mingsheng Ying. Floyd–Hoare logic for quantum programs. *ACM Trans. Program. Lang. Syst.*, 33(6), Jan 2012. doi:10.1145/2049706.2049708.

[YJY22] Peng Yan, Hanru Jiang, and Nengkun Yu. On incorrectness logic for quantum programs. *Proc. ACM Program. Lang.*, 6(OOPSLA1), apr 2022. doi:10.1145/3527316.

[YP21] Nengkun Yu and Jens Palsberg. Quantum abstract interpretation. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021, pages 542–558, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3453483.3454061.

[ZBH$^+$21] Li Zhou, Gilles Barthe, Justin Hsu, Mingsheng Ying, and Nengkun Yu. 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, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1109/LICS52264.2021.9470673.

[ZWD$^+$20] 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. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020. doi:10.1126/science.abe8770.

[ZY17] Li Zhou and Mingsheng Ying. Differential privacy in quantum computation. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 249–262, 2017. doi:10.1109/CSF.2017.23.

[ZYY19] Li Zhou, Nengkun Yu, and Mingsheng Ying. An applied quantum Hoare logic. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, pages 1149–1162, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3314221.3314584.