On the Theory and Practice of Invariant-based Verification of Quantum Programs

Shih-Han Hung Yuxiang Peng Xin Wang Shaopeng Zhu Xiaodi Wu
University of Maryland, College Park, USA

Abstract

We investigate program verification, a fundamental and challenging task in quantum programming, based on quantum Hoare logic and quantum invariants. Ying et al. (POPL'17) have developed a framework based on semidefinite programs (SDPs) for computing quantum invariants for partial correctness, which is however expensive to use due to its exponential scaling in terms of the number of qubits. In this paper, we develop two approaches that can effectively reuse the information generated from quantum invariants so as to avoid repetitive costs in solving exponentially large SDPs. Our first contribution is the formulation of the *master* Hoare triple that encodes the complete information of quantum programs inspired by the Choi-Jamiołkowski isomorphism. We demonstrate how to obtain the master Hoare triple by a single use of SDP, and once it is generated, how to obtain any other Hoare triple from the master one by a simple logic rule, and how master Hoare triples of sub-programs can be reused to simplify the computation of quantum invariants of the main program. Our second contribution is the introduction of *approximate quantum invariants* and a framework to reuse invariants of any given quantum program to efficiently analyze its nearby (e.g., noisy, erroneous) programs without using SDPs. Finally, we demonstrate the use of our techniques on realistic quantum programs, such as quantum Hamiltonian simulation and Repeat-Until-Success, with a huge numerical advantage over the previous SDP approach.

Keywords quantum programming languages, quantum Hoare logic, quantum invariants, invariant generation and analysis, program verification, semidefinite programs.

1 Introduction

Program analysis and verification is a central topic in programming languages. It is an important and challenging task especially for quantum programming, because standard approaches to software assurance are likely to break down in the quantum setting. For example, unit testing is infeasible due to both the indeterminacy of quantum algorithms and the substantial expense involved in executing or simulating them. The rapid development of quantum computing, in particular, the introduction of several quantum programming languages and platforms such as Quipper [16], Scaffold [3], QWIRE [29] Microsoft's Q# [34], IBM's Qiskit [4], Google's Cirq [1], Rigetti's Forest [2], and so on, has also imposed practical need of verification of quantum programs.

Indeed, quantum program verification has been a central topic ever since the seminal work on semantics and language design [15, 26, 30, 31, 33]. There have been many attempts of developing Hoare-like logic [18] for verification of quantum programs [5, 8, 9, 12, 21, 41, 44]. In particular, D'Hondt and Panangaden [11] proposed the notion of quantum weakest precondition, and Ying [41] established quantum Hoare logic for reasoning about partial and total correctness of quantum programs for a quantum extension of the **while**-language. Based these developments, an semidefinite program (SDP) algorithm for quantum invariant generation and termination analysis of quantum programs were developed in [44] and [23] respectively. For detailed surveys, see [14, 32, 42, 43].

The profound influence of classical Hoare logic in the practice of software development [17] has inspired quantum researchers to build a rich tool-box of quantum Hoare logic, e.g., [35, 36, 45], for various quantum applications.

A strong motivation of this paper is to contribute to the tool-box of quantum Hoare logic. Specifically, we focus on the SDP framework proposed by Ying et al. [44] for quantum invariant generation and partial correctness, which has a major drawback because of the exponential scaling of SDP size in terms of the number of involved qubits. Although this complexity is consistent with the one of classical simulation of quantum systems, it becomes very costly in the practical verification of quantum programs with SDPs.

We ask the question *whether such expensive use of SDPs can be mitigated in practical scenarios of verification of quantum programs*. For example, we would hope to avoid repeating the entire SDP-based derivation of Hoare triples if we only want to change the post-condition of any already studied program. Or, if we have already analyzed certain quantum programs, we would hope these analyses would simplify the analysis if we reuse these programs as subroutines.

Another common scenario is to leverage existing program analysis to study the behavior of its "nearby" programs. For example, the nearby programs could be a noisy/erroneous variant of the existing one. Or, they could come from the real-vs-ideal framework in algorithm analysis, where the analysis of a *real* program is conducted by studying a nearby easy-to-analyze *ideal* program and bounding their differences.

Unfortunately, the original SDP framework by Ying et al. [44] cannot leverage these scenarios to mitigate the cost.

Draft paper, November 22, 2019,
2020.

One still needs to solve an exponentially large SDP for every required Hoare triple.

Contributions. We provide an *affirmative* answer to our question by developing techniques that efficiently reuse analysis generated by SDPs. We will restrict our discussion to more relevant terminating programs. We also include preliminaries on quantum information (Section 2), quantum **while** programs (Section 3), and quantum invariants (Section 4).

In the first scenario, we introduce ancilla variables and Hoare triples on the extended space of both original and ancilla variables. Note that these ancilla variables are for analysis purpose only and won't appear in program execution, which distinguishes us from some existing work on *ghost* variables [13, 22]. Inspired by the Choi-Jamiołkowski isomorphism between quantum operations and states (e.g., [38]), we identify the *master* Hoare triples, to encode the entire information of any quantum program. We show such master Hoare triple can be computed by SDPs in a similar way to [44]. Once it is generated, one can derive the weakest precondition for any post-condition by using a newly developed Hoare logic rule, called the Collapse rule, on the master Hoare triple. It can also reduce the size of SDPs for the main program when used as subroutines. (Section 5.)

We introduce *approximate quantum invariants* to address the second scenario. Note that approximation was also introduced to quantum Hoare logic by Zhou et al. [45] to improve its usability by hand. However, it comes at the cost of sacrificing its expressive power (predicate being projection) and numerical infeasibility. Our notion poses a weaker requirement to keep its expressive power and allows its solution by SDPs at the same time. We also develop two approaches that convert quantum invariants of one program to approximate quantum invariants of a nearby one for any quantum program with (a, n)-bounded loops [19]. (Section 6.)

Finally, we demonstrate our techniques numerically on important and realistic quantum programs, such as quantum Hamiltonian simulation [24] and Repeat-Until-Success [7, 27, 40], in the aforementioned two scenarios. In particular, we make a comparison between our new approaches and the original SDP framework from [44] and observe a huge save in the actual running time. (Section 7.)

Related work. Ancilla variables have been introduced in the analysis of Hoare logic either classically (e.g., [13, 22]) or quantumly (e.g., [35]). The specific uses of ancilla variables in both cases are quite different ours. The classical examples [13, 22] actively update ancillas to simplify program reasoning or detect common program errors. The quantum example [35] introduced the ghost variables to express the behavior of quantum program, especially whether its variables share classical or quantum correlations, which is very useful information in cryptographic settings.

2 Preliminaries on quantum information

This section presents basic background on quantum information. Here, we focus on the notations majorly discussed in this paper, and a more self-contained introduction is included in Appendix A. A notation table is included in Appendix C.

2.1 Quantum states

For any finite integer n, an n-dimensional Hilbert space $\mathcal{H}$ is essentially the space $\mathbb{C}^n$ of complex vectors. We use Dirac's notation, $|\psi\rangle$, to denote a complex vector in $\mathbb{C}^n$.

Linear operators between n-dimensional Hilbert spaces are represented by $n \times n$ matrices. The Hermitian conjugate of operator A is denoted by $A^\dagger$. Operator A is *positive simidefinite* if for all vectors $|\psi\rangle \in \mathcal{H}$, $\langle\psi|A|\psi\rangle \geq 0$ This gives rise to the *Löwner order* $\sqsubseteq$ among operators: $A \sqsubseteq B$ if $B - A$ is positive semidefinite.

A *density operator* ρ is a positive semidefinite operator $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ where $\sum_i p_i = 1, p_i > 0$. A special case $\rho = |\psi\rangle\langle\psi|$ is conventionally denoted as $|\psi\rangle$. A positive semidefinite operator ρ on $\mathcal{H}$ is said to be a *partial* density operator if $\mathrm{tr}(\rho) \leq 1$. The set of partial density operators is denoted by $\mathcal{D}(\mathcal{H})$.

2.2 Evolution of a quantum system

The evolution of a quantum system can be characterized by a *completely-positive* and *trace-non-increasing* linear map $\mathcal{E}$ from $\mathcal{D}(\mathcal{H})$ to $\mathcal{D}(\mathcal{H}')$ for Hilbert spaces $\mathcal{H}, \mathcal{H}'$.

For every superoperator $\mathcal{E} : \mathcal{D}(\mathcal{H}) \to \mathcal{D}(\mathcal{H}')$, there exists a set of Kraus operators $\{E_k\}_k$ such that $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$ for any input $\rho \in \mathcal{D}(\mathcal{H})$. The Schrödinger-Heisenberg *dual* of a superoperator $\mathcal{E} = \sum_k E_k \circ E_k^\dagger$, denoted by $\mathcal{E}^* = \sum_k E_k^\dagger \circ E_k$, is another superoperator such that for every state $\rho \in \mathcal{D}(\mathcal{H})$ and any operator A, $\mathrm{tr}(A\mathcal{E}(\rho)) = \mathrm{tr}(\mathcal{E}^*(A)\rho)$.

A quantum *measurement* on a system over Hilbert space $\mathcal{H}$ can be described by a set of linear operators $\{M_m\}_m$ where $\sum_m M_m^\dagger M_m = I_{\mathcal{H}}$. The outcome m when measuring ρ is observed with probability $p_m = \mathrm{tr}(M_m \rho M_m^\dagger)$ for each m. A major difference between classical and quantum computation is that a quantum state collapses after measurement. The collapsion with result m is captured by an superoperator $\mathcal{E}_m = \frac{1}{p_m} M_m \circ M_m^\dagger$.

3 Quantum programs

Our work builds on top of the quantum **while**-language developed by Ying [41, 42]. Here we review the syntax of this language, and present several typical programs. A more detailed introduction to the syntax and the semantics is provided in Appendix B.

3.1 Syntax

The syntax of a *quantum **while** program*, also called a program, P is defined as follows.

$$\begin{aligned} P ::= \quad & \textbf{skip} \mid q := |0\rangle \mid \overline{q} := U[\overline{q}] \mid P_1; P_2 \mid \\ & \textbf{case } M[\overline{q}] = \overline{m \to P_m} \textbf{ end} \mid \\ & \textbf{while } M[\overline{q}] = 1 \textbf{ do } P_1 \textbf{ done}. \end{aligned} \tag{3.1.1}$$

This syntax is analogue to the classical while language, but substituting their quantum operations for their counterparts. The denotational semantics of a quantum **while** program P is a superoperator, denoted by $[\![P]\!]$.

3.2 Examples

We introduce several basic programs written in quantum **while** language here.

Example 3.1 (Teleportation)**.** *Quantum teleportation is a protocol that can send one qubit from the sender to the receiver only using classical communication and shared entanglement. More specifically, Alice communicates a state $|\psi\rangle$ from register a to Bob on register b.*

$$\begin{aligned} \text{QT}[a] \equiv\ & b := |0\rangle;\ c := |0\rangle;\ c := H[c];\ c, b := CNOT[c, b]; \\ & a, c := CNOT[a, c];\ a := H[a]; \\ & \textbf{case } M[a, c] = 00 \to \textbf{skip}; 01 \to b = X[b]; \\ & \qquad 10 \to b = Z[b]; 11 \to b = X[b]; b := Z[b]; \\ & \textbf{end}. \end{aligned} \tag{3.2.1}$$

Example 3.2 (Repeat-Until-Success (RUS) circuit decomposition)**.** *The RUS circuit decomposition technique proposed in Bocharov et al. [7], Paetznick and Svore [27], Wiebe and Roetteler [40] is an alternative approach to implement a unitary with lower T gate complexity by invoking ancillas. The RUS process for implementing a unitary U on n qubits state $|\psi\rangle$ can be outlined as follows.*

1. *Prepare an m-qubit ancilla register $|0^m\rangle$.*
2. *Apply a specially designed unitary W on $|0^m\rangle|\psi\rangle$.*
3. *Measure the ancilla; if the outcome is 0, then we are done; otherwise we apply a recovery process (including applying $W^\dagger$ and a reflection $R := (2M_0 - I) \otimes I$ about M_0) and go back to step (2) again.*

The algorithm can be described with the following program:

$$\begin{aligned} \text{RUS}[q] \equiv\ & p := |0^m\rangle;\ p, q := W[p, q]; \\ & \textbf{while } M[p] \neq 0^m \textbf{ do} \\ & \quad p, q := W^\dagger[p, q];\ p, q := R[p, q];\ p, q := W[p, q] \\ & \textbf{done} \end{aligned} \tag{3.2.2}$$

provided that for every state $|\psi\rangle$,

$$W|0^m\rangle_p|\psi\rangle_q = \sqrt{p}|\Psi\rangle_{pq} + \sqrt{1-p}|\Psi^\perp\rangle_{pq} \tag{3.2.3}$$

where $|\Psi\rangle_{pq} := |0^m\rangle_p U|\psi\rangle_q$ and $|\Psi^\perp\rangle_{pq}$ is some unwanted state orthogonal to $|\Psi\rangle$.

3.3 Quantum Predicates and Hoare Logic

A *quantum predicate* is a Hermitian operator M with $0 \sqsubseteq M \sqsubseteq I$ [11]. For any predicate M and a state ρ, $\text{tr}(M\rho)$ is the expectation of the *truth value* of predicate M in state ρ.

Quantum predicates can be used to express pre- and post-conditions of quantum programs and hence to reason about the semantics of quantum programs in quantum Hoare logic Ying [41, 42]. Specifically, we define

Definition 3.1 (Partial and Total Correctness [41, 42])**.**

1. *The Hoare triple $\{A\}S\{B\}$ is true in the sense of* total *correctness, written $\models_{tot} \{A\}S\{B\}$, if for all $\rho \in \mathcal{D}(\mathcal{H}_S)$,*
$$\text{tr}(A\rho) \leq \text{tr}(B[\![S]\!](\rho)). \tag{3.3.1}$$
2. *The Hoare triple $\{A\}S\{B\}$ is true in the sense of* partial *correctness, written $\models_{par} \{A\}S\{B\}$, if for all $\rho \in \mathcal{D}(\mathcal{H}_S)$,*
$$\text{tr}(A\rho) \leq \text{tr}(B[\![S]\!](\rho)) + [\text{tr}(\rho) - \text{tr}([\![S]\!](\rho))]. \tag{3.3.2}$$

This total correctness inequality (3.3.1) can be seen as the quantum version of the following statement: if state ρ satisfies predicate A of degree $d = \text{tr}(A\rho)$, then after applying the program S the resulting state satisfies predicate B of degree at least d. The partial correctness inequality (3.3.2) also takes the possibility of divergence $\text{tr}(\rho) - \text{tr}([\![S]\!](\rho))$ into consideration. We are concerned with the partial correctness throughout the paper and all of our example programs terminates *almost surely*. In this case, partial correctness implies total correctness.

Definition 3.2 (Weakest (Liberal) Preconditions [41])**.** *For quantum program S and predicate Q, the weakest precondition of Q with respect to S, denoted by $wp.S.Q$, satisfies the following properties.*

1. *It is true that $\models_{tot} \{wp.S.Q\}S\{Q\}$.*
2. *For every quantum predicate R satisfying $\models_{tot} \{R\}S\{Q\}$, it holds that $R \sqsubseteq wp.S.Q$.*

Similarly, for quantum program S and predicate Q, the weakest liberal precondition of Q with respect to S, denoted by $wlp.S.Q$, satisfies the following properties.

1. *It is true that $\models_{par} \{wlp.S.Q\}S\{Q\}$.*
2. *For every quantum predicate R satisfying $\models_{par} \{R\}S\{Q\}$, it holds that $R \sqsubseteq wlp.S.Q$.*

Proposition 3.3 (Ying [42])**.** *For any program S and predicate P, we have the following corresponding between the weakest (liberal) precondition and the denotational semantics $[\![S]\!]$:*

- *(Weakest precondition) $wp.S.P = [\![S]\!]^*(P)$;*
- *(Weakest liberal precondition) $wlp.S.P = [\![S]\!]^*(P) + I - [\![S]\!]^*(I)$.*

4 Invariants of quantum programs

We introduce the notion of superoperator-valued transition system as a framework to model the control flow of quantum

programs. Building on top of that, we will introduce the additive invariant and additively inductive assertion map from Ying et al. [44] and show how to generate the additive invariant using semidefinite programs (SDPs).

4.1 Superoperator-Valued transition systems (SVTS) and Control Flow Graphs

We adopt the definition of superoperator-valued transition systems from [44] with slight notation changes to better serve our use.

Definition 4.1 (Ying et al. [44]). *A superoperator-valued transition system (SVTS) is a 5-tuple $\mathcal{S} = \langle \mathcal{H}, \mathcal{L}, \mathcal{T}, l_{in}, l_{out} \rangle$ where*

1. *$\mathcal{H}$ is the state space, i.e., a Hilbert space.*
2. *$\mathcal{L}$ is a finite set of locations.*
3. *$l_{in}, l_{out} \in \mathcal{L}$ are the initial and terminal locations.*
4. *$\mathcal{T}$ is the set of transitions: each transition $\tau \in \mathcal{T}$ is a 3-tuple $\tau = \langle l, l', \mathcal{E} \rangle$, also denoted by $\tau = l \xrightarrow{\mathcal{E}} l'$, where $l, l' \in \mathcal{L}$, and $\mathcal{E}$ is a superoperator on $\mathcal{H}$. It is required that for each location $l \in \mathcal{L}$,*

$$\sum \{|\mathcal{E} : l \xrightarrow{\mathcal{E}} l' \in \mathcal{T}|\} \cong \mathcal{I}, \tag{4.1.1}$$

The symbol in (4.1.1) stands for a multi-set.

As the construction of an SVTS for a quantum program is not the focus of this paper, we omit the formal rules and refer the readers to [44]. For example, the SVTS of QT and RUS are given in Figure 1. The figures should be self-explantary given their simple structure.

A path (in SVTS $\mathcal{S}_P$) $\pi = l_1 \xrightarrow{\mathcal{E}_1} l_2 \xrightarrow{\mathcal{E}_2} \ldots \xrightarrow{\mathcal{E}_{n-1}} l_n$ is a sequence of vertices such that each pair of adjacent vertices are connected with a transition. We always assume that for each location $l \in \mathcal{L}$, the transition relation is countably branching, i.e., for each pair of locations l, l', the number of paths from l to l' is finite or countably infinite. A set Π of paths in SVTS is said to be *prime* if for each $\pi \in \Pi$, its proper initial segments $l_1 \xrightarrow{\mathcal{E}_1} \ldots \xrightarrow{\mathcal{E}_{k-1}} l_k \notin \Pi$ for all $k < n$. For any path π in the transition graph, we write $l_1 \overset{\pi}{\Rightarrow} l_n$ and denote by $\mathcal{E}_\pi$ the composition of the super-operators along the path, i.e. $\mathcal{E}_\pi = \mathcal{E}_{n-1} \circ \ldots \circ \mathcal{E}_2 \circ \mathcal{E}_1$.

4.2 Additive Invariants, Inductive Assertion Maps and Semi-definite Programs

Inspired by classical invariants of programs and continuous valued logics, Ying et al. [44] introduce additive and multiplicative invariants for quantum programs. We will focus on the *additive* invariant because it allows efficient generation by semidefinite programs [44].

Definition 4.2 (Additive Invariants). *Let $\mathcal{S}_P = \langle \mathcal{H}, \mathcal{L}, l_{in}, l_{out}, \mathcal{T} \rangle$ be an SVTS with a precondition Θ. An additive invariant at location $l \in \mathcal{L}$ is a quantum predicate O_l over $\mathcal{H}$ such that for any density operator ρ, and for any prime set Π of paths from l_{in} to l, we have:*

$$\mathrm{tr}(\Theta\rho) \leq 1 - \mathrm{tr}\left(\mathcal{E}_\Pi(\rho)\right) + \mathrm{tr}\left(O_l \mathcal{E}_\Pi(\rho)\right), \tag{4.2.1}$$

where $\mathcal{E}_\Pi = \sum \{|\mathcal{E}_\pi : \pi \in \Pi|\}$.

Moreover, additive invariant at l_{out} can be used to prove partial correctness of the program P. Specifically,

Lemma 4.3 (Prime paths and semantics of quantum programs). *For quantum program S, let $\mathcal{S}$ be the SVTS defined by S with initial condition Θ. Let Π be the set of paths from l^S_{in} to l^S_{out}. Then $\mathcal{E}_\Pi = [\![S]\!]$.*

Proof. Deferred to Appendix D.1. □

Theorem 4.4 (Ying et al. [44]). *Let P be a quantum program and $\mathcal{S}_P$ the SVTS defined by P with initial condition Θ. If O is an additive invariant at l_{out} in $\mathcal{S}_P$, then $\models_{par} \{\Theta\} P \{O\}$.*

Proof. Deferred to Appendix D.2. □

We provide a complete proof of Theorem 4.4 in Appendix D.2. The complete proof will be helpful and make our technique self-contained in proving an approximate version of Theorem 4.4 in Section 6.

It is, however, nontrivial to prove that any quantum predicate O is an invariant at location l by definition, as one must verify (4.2.1) holds for every prime path from l_{in} to l. In classical programming, induction assertions provide an effective alternative, whose corresponding quantum notion was introduced by Ying et al. [44] as follows.

A *cut-set* of $\mathcal{S}_P$ is a subset $C \subseteq \mathcal{L}$ of locations such that every cyclic path in $\mathcal{S}_P$ passes through some location in C. Every location $l \in C$ is called a cut-point. A *basic* path π between two cut-points l and l' is a path that does not pass through any cut-point other than the endpoints. We note that for cut-set C and $l \in \mathcal{L}$, $C \cup \{l\}$ is also a cut-set.

An *additively inductive assertion map* is a mapping η from each cut-point $l \in C$ to a quantum predicate $\eta(l)$ in $\mathcal{H}$ [44] such that

$$\mathrm{tr}(\eta(l)\rho) \leq 1 - \mathrm{tr}\left(\mathcal{E}_{\Omega_l}(\rho)\right) + \sum_{\pi \in \Omega_l} \mathrm{tr}\left(\eta(l_\pi)\mathcal{E}_\pi(\rho)\right) \tag{4.2.2}$$

where the set Ω_l contains basic paths from l to other cut-points. There are two nice features of the inductive assertion map η shown in Ying et al. [44]: (1) for every cut-point $l \in C$, $\eta(l)$ is an additive invariant at l; (2) the definition of η allows a semi-definite program (SDP) that generate $\eta(l)$ for every cut-point $l \in C$.

Let us setup the SDP for η as follows. Choose a cut-set $C = \{l_0 = l_{in}, l_1, \ldots, l_{m-1}, l_m = l_{out}\}$ and let $O_i = \eta(l_i)$ for $i = 0, \ldots, m$ as the invariants to generate. Let $\mathcal{E}^*_{ij} = \sum_\pi \{|\mathcal{E}^*_\pi : l_i \overset{\pi}{\Rightarrow} l_j|\}$ for $i, j = 0, 1, \ldots, m$ and the summation is over all basic paths π; in particular, if there is no basic path from l_i to l_j, then $\mathcal{E}^*_{ij}$ is the zero super-operator. Consider the following generic SDP.

Invariant-SDP 4.1. *For precondition Θ, find $O_0, \cdots, O_m$ s.t.*

$$0 \sqsubseteq \sum_j \mathcal{E}_{0j}^*(O_j) - \Theta, \tag{4.2.3}$$

$$0 \sqsubseteq \sum_{j\neq i} \mathcal{E}_{ij}^*(O_j) + (\mathcal{E}_{ii}^* - \mathcal{I})(O_i)\ (i = 0, 1, ..., m), \tag{4.2.4}$$

$$0 \sqsubseteq O_i \sqsubseteq I\ (i = 0, 1, ..., m). \tag{4.2.5}$$

Note that $O_i = I$ is a trivial solution. Any non-trivial solution O_i $(i = 0, \cdots, m)$ are non-trivial invariants for the given precondition Θ, which will then imply $\models_{par} \{\Theta\}P\{O_m\}$.

Given a postcondition $O = O_m$, the weakest precondition can be found by maximizing the trace of Θ. This is because for any precondition Θ, it holds that $\Theta \sqsubseteq wp.P.O$. Since $\mathrm{tr}(\cdot)$ respects Löwner ordering, we have $\mathrm{tr}(\Theta) \leq \mathrm{tr}(wp.P.O)$. Similarly, given a precondition Θ, the strongest postcondition can be found by minimizing the trace of O.

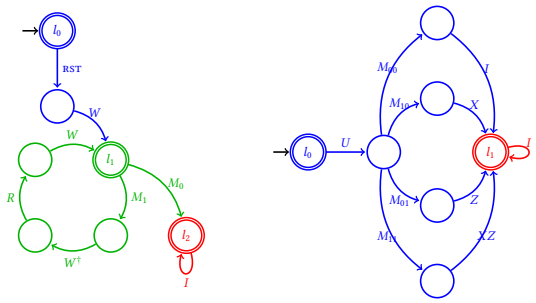


Figure 1. SVTS of RUS (left) and QT (right). The locations of the cut set are labelled with double-lined circles. The basic paths are highlighted with colors. Each colored set of paths can be converted to a positive semidefinite constraint in (4.2.6).

Example 4.1 (Invariants for RUS circuit)**.** *Continuing from Example 3.2, the SVTS of an RUS program is given in Figure 1.*

From the SVTS, the cut-set we choose is $\{l_0, l_1, l_2\}$. Given a postcondition $O = O_2$, we set the SDP problem as follows.

$$\begin{aligned} &\textit{maximize} && \mathrm{tr}(\Theta) \\ &\textit{subject to} && \Theta \sqsubseteq O_0 \\ &&& O_0 \sqsubseteq \mathit{RST}^*(W^\dagger O_1 W) \\ &&& O_1 \sqsubseteq E_0^\dagger O_2 E_0 + E_1^\dagger O_1 E_1\ , \\ &&& O_2 \sqsubseteq O_2 \\ &&& 0 \sqsubseteq \Theta, O_0, O_1, O_2 \sqsubseteq I. \end{aligned} \tag{4.2.6}$$

where $\eta(l_i) = O_i$, $\mathit{RST} = [\![p := |0^m\rangle]\!]$, $E_0 = M_0$, $E_1 = (VM_1)$ and $V := WRW^\dagger$. It can be shown that $\models_{tot} \{\Theta\}\mathrm{RUS}\{O\}$ Note that the constraint $O_2 \sqsubseteq O_2$ is trivial and can be removed from the SDP problem. Furthermore, the solution must satisfy $\Theta = O_0$.

We analyze the SDP problem analytically and numerically. For analytical solution of (4.2.6), we show that for any state $|\psi\rangle$, $\models_{tot} \{I_m \otimes |\psi\rangle\langle\psi|\}\mathrm{RUS}\{|0, U\psi\rangle\langle 0, U\psi|\}$ in Appendix D.3. We will discuss our numerical analysis and the performance in Section 7.

5 Quantum Hoare logic and invariants with ancillas

While quantum Hoare logic provides a framework to reason about quantum programs and the inductive assertion map enables numerical calculation by solving an SDP, a central task in program analysis is to verify the semantics of a program describes a desirable superoperator (instead of generating a particular quantum state). Thus a natural question to ask is whether it is possible to guarantee the correctness of semantics.

We answer the question in the affirmative by giving a framework, in which solving a *master Hoare triple* in an extended space via an SDP solver suffices to show the correctness of semantics.

In Section 5.1, we will layout the setup of extended quantum Hoare triples with ancillas. In Section 5.2, we will show how to generate extended Hoare triples and invariants using semidefinite programs and how it leads to a numerical solution to calculate the weakest (liberal) precondition of any program. Finally, in Section 5.3, we will demonstrate how to use extended Hoare triples of sub-programs for the invariant generation of the main program.

5.1 Quantum Hoare logic in the extended space

We extend the set of quantum variables *Var* for any quantum **while** program S with another isomorphic set of ancilla variables *aVar*, i.e., for each variable in *Var*, there is an ancilla register in *aVar* of the same size. All the rest of the definition of quantum **while** programs remain unchanged. In other words, program S still operates on the original *Var* and the introduction of *aVar* is only for *analysis* purpose.

Let $\mathcal{H}$ denote the Hilbert space for *Var* and $\mathcal{A}$ the Hilbert space for *aVar* and $\mathcal{H} \cong \mathcal{A}$. The (original) quantum predicates, denoted by uppercase letters $P, Q, \ldots$ are on $\mathcal{H}$. From now on, we also consider *extended* quantum predicates that are predicates on $\mathcal{H} \otimes \mathcal{A}$. To distinguish from original ones, we use calligraphic uppercase letters like $\mathcal{P}, \mathcal{Q}, \ldots$ to denote extended predicates.

Hence one can define the *extended Hoare triple* by extended predicates as $\models_{par,tot} \{\mathcal{P}\}S\{\mathcal{Q}\}$, where we overload the notation S for a new program that acts on $\mathcal{H}$ exactly as the original program S and do nothing on $\mathcal{A}$, i.e., with denotational semantics $[\![S]\!] \otimes I_{\mathcal{A}}$. One can trivially extend any original Hoare triple to an extended one by

$$\models_{par,tot} \{P\}S\{Q\} \Rightarrow \models_{par,tot} \{P \otimes I_{\mathcal{A}}\}S\{Q \otimes I_{\mathcal{A}}\}, \tag{5.1.1}$$

simple because $\mathrm{tr}((P \otimes I_A)\rho) = \mathrm{tr}(P\mathrm{tr}_A(\rho))$ for any ρ. However, a more interesting case is to consider extended predicates that are *entangled* between the space $\mathcal{H}$ and $\mathcal{A}$. In

particular, we will consider the extended predicate that is *maximally entangled* between $\mathcal{H}$ and $\mathcal{A}$ in the following.

5.1.1 The Choi-Jamiołkowski isomorphism

The *maximally entangled state* $|\Phi\rangle$ over two isomorphic spaces (such as $\mathcal{H}$ and $\mathcal{A}$) is $|\Phi\rangle := \frac{1}{\sqrt{d}} \sum_{i=1}^{d} |i\rangle_{\mathcal{H}} \otimes |i\rangle_{\mathcal{A}}$, where d is the dimension and $\{|i\rangle_{\mathcal{H}}\}_{i=1}^{d}$ is any orthonormal basis of $\mathcal{H}$ (and $\{|i\rangle_{\mathcal{A}}\}_{i=1}^{d}$ is its isometric one in $\mathcal{A}$). Note that $|\Phi\rangle$ remains the same under different choices of the orthonormal basis.

One interesting use of the maximally entangled state is to build an isomorphism between states and superoperators. The precise correspondence is given by the *Choi-Jamiołkowski* matrix.

Definition 5.1. *Given any superoperator $\mathcal{E} : \mathcal{D}(\mathcal{H}) \to \mathcal{D}(\mathcal{H}')$, the* Choi-Jamiołkowski *matrix $J(\mathcal{E})$ is*

$$J(\mathcal{E}) = d_{\mathcal{H}} \mathcal{J}(\mathcal{E}), \text{ and } J(\mathcal{E}) = \mathcal{E} \otimes I_{\mathcal{A}}(|\Phi\rangle\langle\Phi|), \tag{5.1.2}$$

where $d_{\mathcal{H}}$ is the dimenson of $\mathcal{H}$ and $\mathcal{J}(\mathcal{E})$ is called the normalized *Choi-Jamiołkowski matrix.*

The map J from a quantum superoperator $\mathcal{E}$ to its Choi-Jamiołkowski matrix $J(\mathcal{E})$ is an isomorphism [39]. The map J is also called the Choi-Jamiołkowski isomorphism.

5.1.2 The extended Hoare triples

Since Choi-Jamiołkowski matrix contains the full information of any superoperator and it can be generated by applying $\mathcal{E} \otimes I_{\mathcal{A}}$ on the maximally entangled state $|\Phi\rangle$, it is natural to choose $\Phi \equiv |\Phi\rangle\langle\Phi|$ as the postcondition, and we can derive the extended Hoare triple: $\models_{tot} \{wp.S.\Phi\}S\{\Phi\}$. By Proposition 3.3, we immediately have

$$wp.S.\Phi = (\llbracket S \rrbracket^* \otimes I_{\mathcal{A}})(\Phi) = \mathcal{J}(\llbracket S \rrbracket^*). \tag{5.1.3}$$

Definition 5.2 (Master Hoare Triple)**.** *For any program S, we call the following extended Hoare triple (in total correctness) the* master *Hoare triple:*

$$\models_{tot} \{\mathcal{J}(\llbracket S \rrbracket^*)\}S\{\Phi\}. \tag{5.1.4}$$

Note that for terminating S, the partial correctness $\models_{par} \{\mathcal{J}(\llbracket S \rrbracket^*)\}S\{\Phi\}$ will imply the master Hoare triple (5.1.4).

Example 5.1 (The master Hoare triple of QT)**.** *The denotational semantics of program* QT *is an identity superoperator $\mathcal{I} : \rho \mapsto \rho$. The dual superoperator of $\mathcal{I}$ is $\mathcal{I}$, and thus the master Hoare triple of* QT *is* $\{\Phi\}\text{QT}\{\Phi\}$.

5.1.3 The collapse rule

We derive a *new* rule for our quantum Hoare logic (called the Collapse rule) for total correctness to connect extended Hoare triples with normal ones. To the end, let us first define the collapse operator that maps a predicate on the extended space to the original space.

Definition 5.3 (Collapse Operator)**.** *Given any predicate $\mathcal{P}$ on the extended space $\mathcal{H} \otimes \mathcal{A}$ and any predicate R on the ancilla space $\mathcal{A}$, the collapse operator $\mathfrak{C}_R(\mathcal{P})$ is given by*

$$\mathfrak{C}_R(\mathcal{P}) \equiv d_{\mathcal{H}} \text{tr}_{\mathcal{A}}((I_{\mathcal{H}} \otimes \sqrt{R^T})\mathcal{P}(I_{\mathcal{H}} \otimes \sqrt{R^T})). \tag{5.1.5}$$

We will later see how the operator is used to deduce a Hoare triple in the program space.

We can also establish the following precise properties of the collapse operator.

Lemma 5.4 (Properties of $\mathfrak{C}_R(\cdot)$)**.** *For any predicate R on $\mathcal{A}$, we have*

1. *if $\mathcal{P} \sqsubseteq \mathcal{Q}$, then $\mathfrak{C}_R(\mathcal{P}) \sqsubseteq \mathfrak{C}_R(\mathcal{Q})$, i.e., $\mathfrak{C}_R(\cdot)$ respects Löwner ordering;*
2. *$\mathfrak{C}_R(\Phi) = R$ for the maximally entangled predicate Φ.*
3. *$\mathfrak{C}_R(\mathcal{P}) = d_{\mathcal{H}} \text{tr}_{\mathcal{A}}((I_{\mathcal{H}} \otimes R)\mathcal{P}) = d_{\mathcal{H}} \text{tr}_{\mathcal{A}}(\mathcal{P}(I_{\mathcal{H}} \otimes R))$.*

Proof. Deferred to Appendix D.4. □

Now we are ready to describe our Collapse rule for total correctness.

$$\frac{\{\mathcal{P}\}S\{\mathcal{Q}\}, 0 \sqsubseteq R, \mathfrak{C}_R(\mathcal{P}), \mathfrak{C}_R(\mathcal{Q}) \sqsubseteq I}{\{\mathfrak{C}_R(\mathcal{P})\}S\{\mathfrak{C}_R(\mathcal{Q})\}} \text{ (Collapse)}. \tag{5.1.6}$$

Theorem 5.5. *The rule* Collapse *is sound for total correctness, i.e., for every quantum predicate R, $\models_{tot} \{\mathcal{P}\}S\{\mathcal{Q}\}$ implies $\models_{tot} \{\mathfrak{C}_R(\mathcal{P})\}S\{\mathfrak{C}_R(\mathcal{Q})\}$.*

Proof. First we recall the definition of $\models_{tot} \{\mathcal{P}\}S\{\mathcal{Q}\}$, i.e., for any state $\sigma \in \mathcal{D}(\mathcal{H} \otimes \mathcal{A})$, $\text{tr}(\mathcal{P}\sigma) \leq \text{tr}(\mathcal{Q}(\llbracket S \rrbracket \otimes I_{\mathcal{A}})\sigma)$. For every state $\rho \in \mathcal{D}(\mathcal{H})$, considering $\sigma = \rho \otimes \frac{R}{\text{tr}(R)}$, we have

$$\text{tr}\left(\mathcal{P}\left(\rho \otimes \frac{R}{\text{tr}(R)}\right)\right) \leq \text{tr}\left(\mathcal{Q}(\llbracket S \rrbracket \rho) \otimes \frac{R}{\text{tr}(R)}\right). \tag{5.1.7}$$

Multiplying both sides by $d_{\mathcal{H}}\text{tr}(R)$, we have

$$\begin{aligned} &\text{tr}(d_{\mathcal{H}}\mathcal{P}(I_{\mathcal{H}} \otimes R)\rho \otimes I_{\mathcal{A}}) \\ &\quad \leq \text{tr}(d\mathcal{Q}(I_{\mathcal{H}} \otimes R)\llbracket S \rrbracket(\rho) \otimes I_{\mathcal{A}}). \end{aligned} \tag{5.1.8}$$

By standard calculation, and by Lemma 5.4 (item (3)), we have $\text{tr}(\mathfrak{C}_R(\mathcal{P})\rho) \leq \text{tr}(\mathfrak{C}_R(\mathcal{Q})\llbracket S \rrbracket(\rho))$, for any $\rho \in \mathcal{D}(\mathcal{H})$. This implies $\models_{tot} \{\mathfrak{C}_R(\mathcal{P})\}S\{\mathfrak{C}_R(\mathcal{Q})\}$ and we conclude the proof. □

Let us see how to apply Collapse to the master extended Hoare triple in (5.1.4). We will see that (1) it can be used to generate any postcondition Q; (2) if one starts with the weakest precondition in the extended space, then Collapse derives the weakest precondition in the original space.

Lemma 5.6. *Given $\models_{tot} \{\mathcal{J}(\llbracket S \rrbracket^*)\}S\{\Phi\}$, for any postcondition Q, it holds that $\models_{tot} \{\mathfrak{C}_Q(\mathcal{J}(\llbracket S \rrbracket^*))\}S\{Q\}$. Moreover, the weakest precondition of Q is given by $\mathfrak{C}_Q(\mathcal{J}(\llbracket S \rrbracket^*))$.*

Proof. Deferred to Appendix D.5. □

Note that all of our examples and the interesting programs in consideration terminate almost surely. Hence partial correctness and total correctness is equivalent; the termination guarantee is implicitly used throughout the paper.

Example 5.2 (Teleportation revisited)**.** *In Example 5.1, we have shown the master Hoare triple of* QT *is* $\{\Phi\}\text{QT}\{\Phi\}$. *For any postcondition* $|\psi\rangle\langle\psi|$, *we can choose* $R = |\psi\rangle\langle\psi|$ *and apply* Collapse, *then* $\models_{tot} \{|\psi\rangle\langle\psi|\}QT\{|\psi\rangle\langle\psi|\}$, *where we make use of* $\mathfrak{C}_{|\psi\rangle\langle\psi|}(\Phi) = |\psi\rangle\langle\psi|$ *from Lemma 5.4 (item(2)).*

Remark 5.1. *Throughout the paper, we will consider master Hoare triples (*$\models_{tot} \{\mathcal{J}([\![S]\!]^*)\}S\{\Phi\}$*) with the weakest precondition* $\mathcal{J}([\![S]\!]^*)$. *However, it might be expensive and unnecessary sometime to use the weakest precondition and it is easier to use extended precondition* $\mathcal{P} \sqsubseteq \mathcal{J}([\![S]\!]^*)$. *It is straightforward that* $\models_{tot} \{\mathcal{P}\}S\{\Phi\}$. *We can still derive Hoare triples with any postcondition by applying* Collapse, *similarly to Lemma 5.6. It is, however, no longer guaranteed the derived precondition is the weakest.*

Remark 5.2. *From Lemma 5.6, we conclude that the master Hoare triple is more expressive than a universally quantified statement. This is because* $\{\mathcal{J}([\![S]\!]^*)\}S\{\Phi\}$ *implies*

$$\forall Q,\ \{\mathfrak{C}_Q(\mathcal{J}([\![S]\!]^*))\}S\{Q\}. \tag{5.1.9}$$

Since $\mathfrak{C}_Q(\mathcal{J}([\![S]\!]^*)) = wp.S.Q$, *any Hoare triple can be derived by weakening the precondition.*

5.2 Generation of extended invariants and Hoare triples

In this section, we study how to generate master Hoare triples, which seems hard by definition in (5.1.4) since one needs to compute $[\![S]\!]^*$. We demonstrate here that it is possible to use quantum invariants and their SDP generation (Invariant-SDP 4.1) for terminating programs. We describe how to use the invariant generation framework introduced in Section 4 to compute the master Hoare triple.

To that end, first observe that master (or extended) Hoare triples are just normal Hoare triples on the extended space where the program trivially acts on the ancilla space. Thus, the invariant generation and the derived partial correctness from [44] trivially extend to our case when explicitly including the ancilla space in the setup. For terminating programs, the partial correctness implies total correctness.

Example 5.3 (Analysis of RUS circuit synthesis in the extended space)**.** *To generate the invariant in the extended space, similarly to (4.2.6), we numerically solve the following SDP*

$$\begin{aligned} \textit{maximize}\quad & \text{tr}(Q) \\ \textit{subject to}\quad & Q \sqsubseteq O_0 \\ & O_0 \sqsubseteq \text{RST}^*((W\otimes I)^\dagger O_1 (W\otimes I)) \\ & O_1 \sqsubseteq \widehat{E_0}^\dagger O_2 \widehat{E_0} + \widehat{E_1}^\dagger O_1 \widehat{E_1} \\ & O_2 \sqsubseteq O_2 \\ & 0 \sqsubseteq Q, O_0, O_1, O_2 \sqsubseteq I \end{aligned} \tag{5.2.1}$$

where $O_i = \eta(l_i)$ *is the additively inductive assertion map in the extended space* $\mathcal{H}\otimes\mathcal{A}$ *acting on the cut-points,* $\widehat{E_0} = M_0 \otimes I_{\mathcal{A}}$, *and* $\widehat{E_1} = (VM_1)\otimes I_{\mathcal{A}}$.

To yield a Hoare triple in space $\mathcal{H}$ *with postcondition* O_2, *we first set* $O_2 = \Phi_{\mathcal{HA}}$, *i.e., the maximally entangled state in space* $\mathcal{H}\otimes\mathcal{A}$. *Using an SDP solver to obtain the solution* Q, *we have* $\models_{tot} \{Q\}\text{RUS}\{O_2 = \Phi_{\mathcal{HA}}\}$. *Applying* Collapse *in (5.1.6), we obtain the precondition* $\Theta = \mathfrak{C}_{O_2}(\Phi_{\mathcal{HA}})$ *in space* $\mathcal{H}$.

We will compare the performance of (i) solving (4.2.6), and (ii) solving (5.2.1) followed by applying Collapse *in (5.1.6) in Section 7.*

5.3 Using extended Hoare triples for invariant generation

Imagine that one has already analyzed the invariant generation for some program S_0 and consider another quantum program S that uses S_0 as a subroutine, which is quite common in (quantum) algorithm design. However, in the framework introduced in Section 4, one must consider the additively inductive assertion map of S_0, and include the SDP constraints associated with S_0 for the invariant generation of S. This is because the invariant generation for S_0 itself (without extending the space) may not correspond to the input-output behavior of S_0 when called as a subroutine in S.

We now take a closer look at the precise technical difficulty when using Invariant-SDP 4.1 for S. Let $\mathcal{S}$ be the SVTS of the main program S that is inductively generated from $\mathcal{S}_0$, which is the SVTS of S_0 and other parts of S. We will follow the same setup of Invariant-SDP 4.1 with one exception: we will choose l^0_{in} and l^0_{out} locations of S_0 as cut-points and no other locations in S_0 will be cut-points. This is intuitive as one wants to treat S_0 as a whole and does not want to go inside $\mathcal{S}_0$. Moreover, we can further require all paths from any cut-point other than l^0_{in} and l^0_{out} to l^0_{out} will visit l^0_{in} first and all paths from l^0_{in} will remain in $\mathcal{S}_0$ until reaching l^0_{out}. This can be assumed without loss of generality by adding dummy locations with I transitions in $\mathcal{S}$ for terminating S_0. As a result, all basic paths that go inside $\mathcal{S}_0$ will have l^0_{in} and l^0_{out} as endpoints. Let O^0_{in} and O^0_{out} denote the quantum invariant at the l^0_{in} and l^0_{out} locations of S_0. Thus, the entire effect of S_0 in Invariant-SDP 4.1 will be represented by one semidefinite constraint in the form of (4.2.5) as $0 \sqsubseteq [\![S_0]\!]^*(O^0_{out}) - O^0_{in}$. Any normal Hoare triple of S_0 can only establish the connection for a specific pair of O^0_{in} and O^0_{out}, rather than for an arbitrary pair of O^0_{in} and O^0_{out} as variables.

The technical difficulty, however, can be well addressed by using the master Hoare triple of S_0. Intuitively, this is because the combination of the master Hoare triple and Collapse can derive any such pair as demonstrated in Lemma 5.6. Precisely, given the master Hoare triple of S_0, $\models_{tot} \{\mathcal{J}([\![S_0]\!]^*)\}S_0\{\Phi\}$. Using Lemma 5.6 and Lemma 5.4 (item(3)), any specific pair

of O^0_{in} and O^0_{out} can be connected by $O^0_{in} \sqsubseteq \mathfrak{C}_{(O^0_{out})}(\mathcal{J}([\![S_0]\!]^*))$, which is a semidefinite constraint in O^0_{in} and O^0_{out}.

As a result, we demonstrate a general approach to leverage the master Hoare triple (in total correctness) of any terminating sub-program S_0 to simplify the invariant generation SDP of the main program S (for partial correctness).

Example 5.4 (Consecutive unitary applications implemented with RUS)**.** *In this example, we consider two consecutive applications of unitaries W_1 and W_2 satisfying*

$$W_i|0^m\rangle|\psi\rangle = \sqrt{p_i}|0^m\rangle U_i|\psi\rangle + \sqrt{1-p_i}|\tau_i\rangle|\psi\rangle \tag{5.3.1}$$

for $i \in \{1,2\}$. The quantum program $S \equiv \text{RUS}_1; \text{RUS}_2$ where RUS_i is the RUS program associated with W_i. The SVTS of S is illustrated in Figure 2.

Without the extended space, for a given postcondition $O = O^{(2)}_2$, the corresponding SDP is

$$\begin{aligned}
\textit{maximize}\quad & \operatorname{tr}(\Theta)\\
\textit{subject to}\quad & \Theta \sqsubseteq O^{(1)}_0\\
& O^{(1)}_0 \sqsubseteq \text{RST}^{(1)*}(W^{(1)\dagger} O^{(1)}_1 W^{(1)})\\
& O^{(1)}_1 \sqsubseteq E^{(1)}_0 O^{(1)}_2 E^{(1)}_0 + E^{(1)}_1 O^{(1)}_1 E^{(1)}_1\\
& O^{(2)}_2 \sqsubseteq \text{RST}^{(2)*}(W^{(2)\dagger} O^{(2)}_1 W^{(2)})\\
& O^{(2)}_1 \sqsubseteq E^{(2)}_0 O^{(2)}_2 E^{(2)}_0 + E^{(2)}_1 O^{(2)}_1 E^{(2)}_1\\
& O^{(2)}_2 \sqsubseteq O^{(2)}_2\\
& 0 \sqsubseteq \Theta, O^{(1)}_0, O^{(1)}_1, O^{(1)}_2, O^{(1)}_1, O^{(2)}_2 \sqsubseteq I
\end{aligned} \tag{5.3.2}$$

where $\eta(l^{(k)}_i) = O^{(k)}_i$, $\text{RST}^{(k)} = [\![p := |0^m\rangle]\!]$, $E^{(k)}_0 = M^{(k)}_0$, $E^{(k)}_1 = (V^{(k)} M^{(k)}_1)$ and $V^{(k)} := W^{(k)} R^{(k)} (W^{(k)})^\dagger$ for RUS_k and $k \in \{1,2\}$. The SDP in (5.3.2) can be obtained by setting $O^{(1)}_2 = \Theta^{(2)}$ and combine the two sets constraints in Example 3.2. For each postcondition, we run an SDP solver to obtain a precondition.

Applying our method, in Example 5.3, we have shown that $\models_{tot} \{Q^{(i)}\}\text{RUS}_i\{O^{(i)}\}$. Having the master Hoare triples, with the postcondition $O = O^{(2)}$, we obtain the precondition by applying Collapse*, which leads to the following SDP*

$$\begin{aligned}
\textit{maximize}\quad & \operatorname{tr}(\Theta^{(1)})\\
\textit{subject to}\quad & \Theta^{(2)} \sqsubseteq \mathfrak{C}_O(Q^{(2)})\\
& \Theta^{(1)} \sqsubseteq \mathfrak{C}_{\Theta^{(2)}}(Q^{(1)}).
\end{aligned} \tag{5.3.3}$$

We will compare the performance of (i) using (5.3.2), and (ii) solving (5.2.1) for two subprograms RUS_1 and RUS_2 first following the application of (5.3.3) in Section 7.

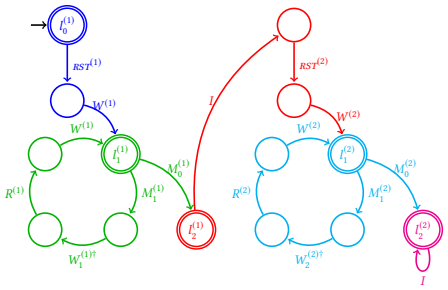


Figure 2. The SVTS of $\text{RUS}_1; \text{RUS}_2$. The SVTS is generated by connecting the graphs (see Figure 1) for each component. The cut-points are labelled with double-lined circles. Each colored path corresponds to a positive semidefinite constraint; see (5.3.2).

6 Correctness and invariants with approximations

The motivation of our discussion lies in the real world constraints in near-term quantum computers, where the operations are imprecise and noisy. To quantify the error due to noisy and imprecise operations in quantum programs, we propose the definition of ϵ-partial correctness, which can be seen as a relaxation of Definition 3.1. Our main contribution in this section is a bridge from the invariants of a program to approximate invariants of its nearby variations.

In Section 6.1, we present the setup of program variations, and the approximate correctness of the known Hoare triples applying on the variations. In Section 6.2, we show two approaches to generate the approximate correctness of the variations with loop boundedness. In Section 6.3, we apply our methods to the Trotter's algorithm for the Hamiltonian simulation problem to formally verify its approximate correctness.

6.1 Transferring Hoare triples to variations

In this subsection, we depict the overall picture of our approaches: applying the Hoare triples of one program to its nearby variations. To achieve it, we first define the meaning of variations as program isomorphisms, followed by the concepts of approximate correctness.

6.1.1 Bridging programs via isomorphism

Definition 6.1. *For quantum programs P_1, P_2, let their SVTS's be $\mathcal{S}_1 = \langle \mathcal{H}_1, \mathcal{L}_1, \mathcal{T}_1, l^{\mathcal{S}_1}_{in}, l^{\mathcal{S}_1}_{out}\rangle, \mathcal{S}_2 = \langle \mathcal{H}_2, \mathcal{L}_2, \mathcal{T}_2, l^{\mathcal{S}_2}_{in}, l^{\mathcal{S}_2}_{out}\rangle$ respectively. We say that $\mathcal{S}_1$ and $\mathcal{S}_2$ are* isomorphic*, if $\mathcal{H}_1 = \mathcal{H}_2, \mathcal{L}_1 = \mathcal{L}_2, l^{\mathcal{S}_1}_{in} = l^{\mathcal{S}_2}_{in}, l^{\mathcal{S}_1}_{out} = l^{\mathcal{S}_2}_{out}$, and there is a bijection $\mu : \mathcal{T}_1 \to \mathcal{T}_2$, s.t. for all $\tau = \langle l_1, l_2, \mathcal{E}_\tau\rangle \in \mathcal{T}_1$, $\mu(\tau) = \langle l_1, l_2, \mathcal{E}_{\mu(\tau)}\rangle \in \mathcal{T}_2$. We say that the program P_1 and P_2 are* isomorphic*, if $\mathcal{S}_1$ and $\mathcal{S}_2$ are isomorphic.*

Intuitively, $\mathcal{S}_1$ and $\mathcal{S}_2$ are isomorphic when they share the same graphical structure, with only values (superoperators) of the transitions differing. We can determine the program isomorphism in polynomial time by constructing and traversing the SVTS's from the initial locations.

As we will see, the isomorphic relation of programs is useful since it allows us to reason about a quantum program when the information about the ideal program is well-understood or easier to analyze. To explain the idea, we consider the following example.

Example 6.1. *We consider two isomorphic quantum 3-qubit RUS programs* RUS_3 *and* RUS_3'*. We construct* $\mathcal{E}_U$ *by*

$$W_U = (H\otimes H\otimes Z)CCX(U\otimes I\otimes S)CCX(H\otimes H\otimes I),$$
$$\mathcal{E}_U = \sum_{x\in\{0,1\}^2} W_U(|00\rangle\langle x|\otimes I)\circ(|x\rangle\langle 00|\otimes I)W_U^\dagger. \quad (6.1.1)$$

Here $CCX[p,q]$ *represents the Toffoli gate. Then* RUS_3 *is*

$\text{RUS}_3 \equiv$ $p,q := \mathcal{E}_I[p,q];$
while $M[p]\neq 00$ **do** $p,q := \mathcal{E}_I[p,q]$ **done**; $q := Z[q]$.

The program RUS_3 *realizes the same funcionality as the RUS program introduced in Example 5.3.*

We introduce its variation RUS_3' *by perturbating* $\mathcal{E}_I$ *in the* RUS_3 *to* $\mathcal{N} = 0.9\mathcal{E}_I + 0.05\mathcal{E}_X + 0.05\mathcal{E}_Z$*:*

$\text{RUS}_3' \equiv$ $p,q := \mathcal{N}[p,q];$
while $M[p]\neq 00$ **do** $p,q := \mathcal{N}[p,q]$ **done**; $q := Z[q]$.

The difference between RUS_3' *and* RUS_3 *is the noise added on the first qubit between the two Toffoli gates. Their SVTS's are shown in Figure 3. Apparently* RUS_3' *and* RUS_3 *are isomorphic.*

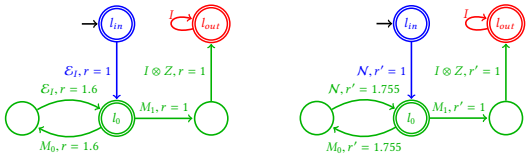


Figure 3. The SVTS and repetition function for RUS_3 and RUS_3'.

6.1.2 ϵ-partial correctness and ϵ-approximate additive invariants

The total correctness and partial correctness allow no tolerance in their definitions, i.e., the invariants generated from an *ideal* program no longer applies when the program is noisy. We start to define the semantics for a Hoare triple parameterized with error threshold ϵ.

Definition 6.2 (ϵ-partial correctness)**.** *For* $\epsilon\in[0,1]$*, a correctness formula* $\{P\}S\{Q\}$ *is true in the sense of* ϵ-partial correctness*, denoted by* $\models_{par,\epsilon}\{P\}S\{Q\}$*, if for every state* ρ*, it holds that*

$$\text{tr}(P\rho)\le \text{tr}(Q[\![S]\!]\rho) + (1+\epsilon)\text{tr}\rho - \text{tr}([\![S]\!]\rho). \quad (6.1.2)$$

Definition 6.2 can be viewed as a relaxation since the (ideal) Hoare triple is the special case $\epsilon = 0$. To see why Definition 6.2 gives a reasonable definition when reasoning about approximate programs, we give some intuition here. Suppose that we want to verify a quantum program S which on input state $|\psi\rangle$, produces a state $\sqrt{1-\epsilon}|\phi\rangle + \sqrt{\epsilon}|\Psi^\perp\rangle$, i.e., with probability $(1-\epsilon)$ it produces the desirable output state $|\phi\rangle$ and with probability ϵ it produces some unknown and undesirable state $|\Psi^\perp\rangle$ orthogonal to $|\phi\rangle$. In this case, for program S which terminates almost surely, $\models_{par,0}\{|\psi\rangle\langle\psi|\}S\{|\phi\rangle\langle\phi|\}$ is not valid with the state $\rho = |\psi\rangle\langle\psi|$ since $\text{tr}(Q[\![S]\!]\rho) = 1-\epsilon$ where $Q = |\phi\rangle\langle\phi|$. On the other hand, we know that $\models_{par,\epsilon}\{|\psi\rangle\langle\psi|\}S\{|\phi\rangle\langle\phi|\}$ is true.

With ϵ-partial correctness, we define ϵ-approximate additive invariants. Similarly to [44], the ϵ-approximate additive invariant can be utilized to prove the ϵ-partial correctness of a quantum program.

Definition 6.3 (ϵ-approximate additive invariants)**.** *Let* $\mathcal{S} = \langle\mathcal{H},\mathcal{L},\mathcal{T},l_{in},l_{out}\rangle$ *be an SVTS, and* Θ *be the initial predicate at* l_{in}*. An* ϵ-approximate additive invariant O_l *at location* $l\in\mathcal{L}$ *is a quantum predicate in Hilbert space* $\mathcal{H}$ *such that for any state* $\rho\in\mathcal{D}(\mathcal{H})$ *and any prime set of paths* Π *from* l_0 *to* l*, we have*

$$\text{tr}(\Theta\rho)\le \text{tr}(O_l[\![S]\!]\mathcal{E}_\Pi(\rho)) + (1+\epsilon)\text{tr}(\rho) - \text{tr}([\![S]\!]\rho).$$

Theorem 6.4. *For quantum program* S*, let* $\mathcal{S}$ *be the SVTS defined by* S *with initial condition* Θ*. If* O *is an* ϵ*-approximate additive invariant at* l_{out}^S *in* $\mathcal{S}$*, then* $\models_{par,\epsilon}\{\Theta\}S\{O\}$.

Proof. By Definition 6.3 and Definition 6.2, it suffices to show that $\mathcal{E}_\Pi = [\![S]\!]$. By Lemma 4.3, we conclude the proof. □

6.2 Generation of approximate correctness

In this section, we present two efficient methods to employ the information of a program to generate the ϵ-partial correctness for its variations.

1. *Inductive offset tracking* requires the inductive assertion map and the description of their transitions.
2. *Superoperator distance bounds* does not require the information of the inductive assertion map; instead, it requires only the invariant at the desirable locations.

In most cases, the second method generates looser bounds than the first method.

In the following analysis, our programs of interest will be those which terminates. For each while loop in the program, we invoke the definition of boundedness [19] and require every loop in the program to be bounded.

Definition 6.5 ([19])**.** *A loop* ***while*** $M[q]=1$ ***do*** P_1 ***done*** *is said to be* (a,n)*-bounded for* $a\in[0,1)$ *and* $n\in\mathbb{Z}_+$ *if* $(\mathcal{E}^*)^n(M_1^\dagger M_1)\sqsubseteq aM_1^\dagger M_1$*, where* $\mathcal{E}(\rho) = [\![P_1]\!](M_1\rho M_1^\dagger)$ *and* $\mathcal{E}^*$ *is the dual superoperator of* $\mathcal{E}$*.*

6.2.1 Inductive offset tracking

Following the inductive assertion map introduced in [44] and their SDP constraints, we consider the error of each term induced on each SDP constraint. In this section, without loss of generality, we select the cut-set of the inductive assertion map to be all locations in the SVTS. This simplifies our statements, while the same idea of proofs can be applied on any selected cut-set.

It is helpful to define some notations before we state our results. We use the prime sign to indicate all the concepts in the variation program corresponding to the ones in the original program. For two isomorphic programs P, P', let $\mathcal{S} = \langle \mathcal{H}, \mathcal{L}, \mathcal{T}, l_{in}^{\mathcal{S}}, l_{out}^{\mathcal{S}} \rangle$ and $\mathcal{S}' = \langle \mathcal{H}, \mathcal{L}, \mathcal{T}', l_{in}^{\mathcal{S}}, l_{out}^{\mathcal{S}} \rangle$ be their SVTS's, and $\mu : \mathcal{T} \to \mathcal{T}'$ be the isomorphism. Because the graphical structures of $\mathcal{S}$ and $\mathcal{S}'$ are identical, by μ we can relate the superoperators composed by transitions, paths, and prime path sets in $\mathcal{S}$ to those in $\mathcal{S}'$. Because the graphical structures of $\mathcal{S}$ and $\mathcal{S}'$ are identical, we can map the superoperators Formally, we denote the value of a transition $\mu(\tau) \in \mathcal{T}'$ by $\mathcal{E}'_\tau = \mathcal{E}_{\mu(\tau)}$. For a path π in $\mathcal{S}$, a prime set Π of paths in $\mathcal{S}$, let $\mathcal{E}'_\pi$ be the superoperator in $\mathcal{S}'$ along the shared path π, and $\mathcal{E}'_\Pi = \sum_{\pi \in \Pi} \{|\mathcal{E}'_\pi|\}$.

For a program P with every **while** loop (a, n)-bounded, and its SVTS $\mathcal{S} = \langle \mathcal{H}, \mathcal{L}, \mathcal{T}, l_{in}, l_{out} \rangle$, we introduce its repetition function $r : \mathcal{T} \to \mathbb{R}_{\geq 0}$. The repetition function r depicts the worst-case expected number of repetitions of a transition in the execution of P.

Definition 6.6. *For program P with every **while** loop (a, n)-bounded, and its SVTS $\mathcal{S}$, its repetition function r is defined inductively with the generation of $\mathcal{S}$.*

- $P \equiv$ ***skip***, $\bar{q} := U[\bar{q}], q := |0\rangle$: $r \equiv 1$.
- $P \equiv P_1; P_2$: *let $\mathcal{T}^{P_1}, \mathcal{T}^{P_2}$ be the set of transitions of the SVTSs of P_1, P_2 respectively. Suppose that $r_1 : \mathcal{T}^{P_1} \to \mathbb{R}_{\geq 0}, r_2 : \mathcal{T}^{P_2} \to \mathbb{R}_{\geq 0}$ are the repetition functions for P_1, P_2. Set*

$$r(\tau) = \begin{cases} r_1(\tau), & \tau \in \mathcal{T}^{P_1} \\ r_2(\tau), & \tau \in \mathcal{T}^{P_2}. \end{cases} \tag{6.2.1}$$

  For the transition $\tau = \langle l_{out}^{P_1}, l_{in}^{P_2}, I \rangle$, set $r(\tau) = 1$.
- $P \equiv$ ***case*** $(M[q] = m \to P_m)$ ***end***: *let $\mathcal{T}^{P_m}$ be the set of transitions of the SVTS for P_m respectively. Suppose that $r_m : \mathcal{T}^{P_m} \to \mathbb{R}_{\geq 0}$ are the transition-bounding functions for P_m. Set*

$$r(\tau) = \begin{cases} r^m(\tau), & \exists m, \tau \in \mathcal{T}^{P_m}, \\ 1, & \textit{else}. \end{cases} \tag{6.2.2}$$

- $P \equiv$ ***while*** $M[\bar{q}] = 1$ ***do*** Q ***done***: *We assume this loop is (a, n)-bounded. Let τ_0 be the transition with value M_0 starting from l_{in}, and τ_1 be the one with value M_1. Denote the transition set of the SVTS Q by $\mathcal{T}^Q$. With $r_Q : \mathcal{T}^Q \to \mathbb{R}_{\geq 0}$ the repetition function of Q, we set*

$$r(\tau) = \begin{cases} \frac{n}{1-a} r_Q(\tau), & \tau \in \mathcal{T}^Q, \\ 1, & \tau = \tau_0, \\ \frac{n}{1-a}, & \tau = \tau_1. \end{cases} \tag{6.2.3}$$

We can verify that the repetition functions are well-defined with the renamings in the construction of SVTS.

Definition 6.7. *For a program P, its additive assertion map η, its isomorphic program P', we define the offset δ_τ of transition τ in $\mathcal{T}$ as :*

$$\delta_\tau = \max\left\{0, \lambda_{\max}\left((\mathcal{E}_\tau^* - \mathcal{E}_\tau'^*)(I - \eta(l_\tau))\right)\right\}, \tag{6.2.4}$$

where l_τ is the end-point of τ, and $\lambda_{\max}(M)$ is the largest eigenvalue of M.

Then for any $\tau \in \mathcal{T}$, it holds that $\mathcal{E}_\tau'^*(I - \eta(l_\tau)) - \delta_l I \sqsubseteq \mathcal{E}_\tau^*(I - \eta(l_\tau))$.

In order to obtain accurate error bounds, we observe that a transition does not always influence the behavior of all the locations. A location l is only effected by those transitions that may lead a path to it, formally $\mathcal{T}_l = \{\tau \in \mathcal{T} : \tau = \langle l', l'', \mathcal{E} \rangle, \exists \pi, \pi = l' \xrightarrow{\mathcal{E}} l'' \to \cdots \to l\}$.

Lemma 6.8. *For isomorphic programs P, P', let r' be the repetition function of P' and η be the inductive assertion map of P. For any location l and any prime set Π of paths from l_{in} to l, it holds that*

$$\sum_{\pi \in \Pi} \mathcal{E}_\pi'^*(I - \eta(l_\pi)) \sqsubseteq I - \eta(l_{in}) + \sum_{\tau \in \mathcal{T}_l} r'(\tau)\delta_\tau I. \tag{6.2.5}$$

Proof. Deferred to Appendix D.8. □

Theorem 6.9. *For isomorphic programs P, P', let r' be the repetition function of P' and η be the inductive assertion map of P. For any location $l \in \mathcal{L}$, $\eta(l)$ is an ϵ-approximate invariant of P', where $\epsilon = \sum_{\tau \in \mathcal{T}_l} r'(\tau)\delta_\tau$.*

Proof. For any ρ and prime path set Π, we apply Lemma 6.8:

$$\begin{aligned} &\sum_{\pi \in \Pi} \mathrm{tr}((I - \eta(l_\pi))\mathcal{E}'_\pi(\rho)) = \sum_{\pi \in \Pi} \mathrm{tr}(\mathcal{E}_\pi'^*(I - \eta(l_\pi))\rho) \\ &\leq (1 + \sum_{\tau \in \mathcal{T}_l} r'(\tau)\delta_\tau)\mathrm{tr}(\rho) - \mathrm{tr}(\eta(l_\epsilon)\rho). \end{aligned} \tag{6.2.6}$$

Hence it is $\sum_{\tau \in \mathcal{T}_l} r'(\tau)\delta_\tau$-approximation. □

Example 6.2. *We illustrate the usage of Theorem 6.9 by variation* RUS_3' *of the program* RUS_3 *as introduced in Example 6.1.*

By solving SDP with $\Theta = |000\rangle\langle 000|$, the RUS_3 *program has a additive assertion map $\eta(l_{in}) = \Theta = |000\rangle\langle 000|$, $\eta(l_0) = O_0 = |\phi\rangle\langle\phi| \otimes |0\rangle\langle 0|$ and $\eta(l_{out}) = O = |000\rangle\langle 000|$, where $|\phi\rangle = ((3+i)|00\rangle + (1-i)|01\rangle + (1-i)|10\rangle - (1-i)|11\rangle)/4$.*

Denote the transition from l_a to l_b by $\tau_{a,b}$. We construct the SDP with relaxation for RUS_3'*, and calculate the offset $\delta_{\tau_{in,0}} = \max\{0, \lambda_{\max}((\mathcal{E}_I - \mathcal{N})^*(I - \eta(l_0)))\} = 0.0140, \delta_{\tau_{0,0}} = 0.0140, \delta_{\tau_{0,out}} = 0$. We can verify that the loop in* RUS_3' *is*

$(0.4302, 1)$-*bounded. Thus, the repetition function is:* $r'(\tau_{in,0}) = 1$, $r'(\tau_{0,0}) = 1.7550$, *and* $r'(\tau_{0,out}) = 1$.

We can obtain an error bound $\epsilon(l_{out}) = 0.0386$ *by Theorem 6.9, thus construct the* ϵ*-partial correctess* $\models_{par,0.0386} \{\Theta\}\text{RUS}'_3\{O\}$ *according to Theorem 6.4.*

6.2.2 Superoperator distance bounds

While the method in Section 6.2.1 provides an accurate bound, it is somewhat impractical to acquire the boundedness of the variation and the inductive assertion map of the ideal program, since as mentioned before, In this section, we provide a different approach, which only requires the boundedness of the ideal program.

We introduce the distance measurements of superoperators and quantum states, to depict the error bound. The definitions follow [39].

Definition 6.10. *For vector space* $\mathcal{X}, \mathcal{Y}$ *and linear operator* $A : \mathcal{X} \to \mathcal{Y}$*, the* 1-norm $\|A\|_1 = \text{tr}\sqrt{A^\dagger A}$*, which is also interpreted as the summation of absolute values of A's eigenvalues.*

Definition 6.11. *For Hilbert space* $\mathcal{H}$ *and* $\mathcal{A} \cong \mathcal{H}$*, The diamond norm for superoperators* $\mathcal{E}_1, \mathcal{E}_2 : \mathcal{D}(\mathcal{H}) \to \mathcal{D}(\mathcal{H})$ *is defined as*

$$\|\mathcal{E}_1 - \mathcal{E}_2\|_\diamond = \max_{\rho \in \mathcal{H}\otimes\mathcal{A}} \frac{1}{2} \|\mathcal{E}_1 \otimes I_{\mathcal{A}}(\rho) - \mathcal{E}_2 \otimes I_{\mathcal{A}}(\rho)\|_1 .$$

The diamond norm of two superoperators is equivalent to that of the superoperators restricted on the subspace capturing the differences. Specifically, in a quantum program, the diamond norm of the superoperators derived from two commands is captured by the subspace of the registers mentioned in the commands. For example, the diamond norm of superoperators $\mathcal{E}_1$ and $\mathcal{E}_2$ acting on register q is $\|\mathcal{E}_1 - \mathcal{E}_2\|_\diamond$ defined on the Hilbert space of q. Thus it is not necessary to compute the diamond norm of the entire Hilbert space. The diamond norm is computable by an SDP solver [37].

Lemma 6.12. *Let* P *be an ideal program in which every loop is* (a, n)*-bounded,* P' *be a variation isomorphic to* P*, and* $\mathcal{S}$ *be the SVTS of* P*. Let* r *be the repetition function of* P *and* $l \in \mathcal{L}$*. Then, for any prime set* Π *of paths from* l_{in} *to* l*, it holds that*

$$\|\mathcal{E}_\Pi - \mathcal{E}'_\Pi\|_\diamond \le \sum_{\tau \in \mathcal{T}_l} r(\tau) \left\|\mathcal{E}_\tau - \mathcal{E}'_\tau\right\|_\diamond . \tag{6.2.7}$$

Proof. Deferred to Appendix D.6. □

Theorem 6.13. *Let* P *be an ideal program in which every loop is* (a, n)*-bounded,* P' *be a variation isomorphic to* P*, and* $\mathcal{S}$ *be the SVTS of* P*. Let* r *be the repetition function of* P *and* $l \in \mathcal{L}$*. If* O *is an additive invariant for* P *at* l*, then it is also an* ϵ*-approximate additive invariant for* P' *at* l*, where* $\epsilon = \sum_{\tau \in \mathcal{T}_l} r(\tau) \left\|\mathcal{E}_\tau - \mathcal{E}'_\tau\right\|_\diamond$.

Proof. Deferred to Appendix D.7. □

Example 6.3. *We illustrate this approach via* RUS_3 *and* RUS'_3 *programs. We utilize the notations in Example 6.2.*

From above we know $\models_{par} \{\Theta\}\text{RUS}_3\{O\}$*. We can verify the while loop in* RUS_3 *is* $(0.375, 1)$*-bounded, thus the transition-bounding function is* $r(\tau_{in,0}) = 1, r(\tau_{0,0}) = 1.6$ *and* $r(\tau_{0,out}) = 1$*. Meanwhile,* $\|\mathcal{E}_I - \mathcal{N}\|_\diamond \le \|\mathcal{I} - \mathcal{N}_1\|_\diamond = 0.1$*, where* $\mathcal{N}_1 = (0.9\mathcal{I} + 0.05\mathcal{X} + 0.05\mathcal{Z})$*, and* $\mathcal{X}$ *and* $\mathcal{Z}$ *are the channels applying* X *and* Z *operators. This diamond norm gives a bound to the difference of transition* $\tau_{0,0}$ *as well.*

Applying Theorem 6.13, we know the invariant O *for* RUS *at location* l_{out} *is an* 0.260*-approximate additive invariant for* RUS′*. This constructs the* ϵ*-partial correctess* $\models_{par,0.260} \{\Theta\}\text{RUS}'_3\{O\}$.

6.3 Case study: Hamiltonian simulation with Trotter's formula

In this section, we demonstrate how to verify the correctness of the quantum algorithms involving approximations. As a case study, we verify the error bound on the Trotter's formula for the Hamiltonian simulation problem, in which we are given a Hamiltonian $H = \sum_j H_j$, aiming to simulate e^{-iHt}. Using Trotter's formula, we divide the time t into n steps, in each of which $e^{-i\frac{t}{n}H_j}$ is applied for $j = 1 \ldots L$ sequentially.

First we introduce a sub-program J and its ideal version K used in the main programs

$$J \equiv q := \left(\prod_{j=L}^{1} e^{-\frac{it}{n}H_j}\right)[q], \qquad K \equiv q := (e^{-\frac{it}{n}H})[q].$$

To clarify, J and K are abbreviations of the detailed clauses in the presentation, and do not work like procedures. Then the program TF (TrotterFormula) realizing the Trotter's formula for a quantum state stored in q, and the corresponding ideal isomorphic program HS are

$$\text{TF} \equiv g := |0\rangle;\ \textbf{while}\ M_{<n}[g] = 1\ \textbf{do}\ J; g := \text{Inc}[g]\ \textbf{done},$$
$$\text{HS} \equiv g := |0\rangle;\ \textbf{while}\ M_{<n}[g] = 1\ \textbf{do}\ K; g := \text{Inc}[g]\ \textbf{done},$$

where $M_{<n} = \sum_{j=0}^{n-1} |j\rangle\langle j|$, $g \in \mathcal{H}_{n+1}$, and $\text{Inc} = |0\rangle\langle n| + \sum_{j=0}^{n-1} |j+1\rangle\langle j|$.

Since the counter on g increments in each step, the **while** loop is $(0, n)$-bounded. The SVTS of TF and transition-bounding function of HS are given in Fig. 4. First we consider the program without auxiliary space. Because $e^{-\frac{ikt}{n}H} e^{-\frac{it}{n}H} = e^{-\frac{i(k+1)t}{n}H}$ for all k, and the behavior of operations on g are equivalent to resetting g to $|n\rangle$, for any $|\psi\rangle$, we have

$$\models_{par} \{I \otimes |\psi\rangle\langle\psi|\} HS \{|n\rangle\langle n| \otimes (e^{-itH}|\psi\rangle\langle\psi|e^{itH})\}. \tag{6.3.1}$$

By [10, Appendix F.3] and [6, Proposition 7], we have

$$\|[\![J]\!] - [\![K]\!]\|_\diamond \le 2\left(\frac{L\Lambda t}{n}\right)^2 e^{\frac{L\Lambda t}{n}},$$

		RUS x 1		RUS x 2		RUS x 7		RUS x 10	
	Master	Normal	Collapse	Normal	Collapse	Normal	Collapse	Normal	Collapse
RUS_2	1.91	603.83	2.30	687.16	1.77	1,073.65	5.17	1,279.16	7.35
RUS_3	21.2	649.34	2.14	871.35	4.17	1,687.38	14.47	2,135.04	20.67

Table 1. Benchmarks of computing 1,000 random Hoare triples of repeating an RUS program n times in sequence for $n = 1, 2, 7, 10$ using the normal method and the master+collapse method. The unit of the CPU times is second (s).The matser Hoare triples of RUS_2 and RUS_3 are reused for the calculation in the collapse part.

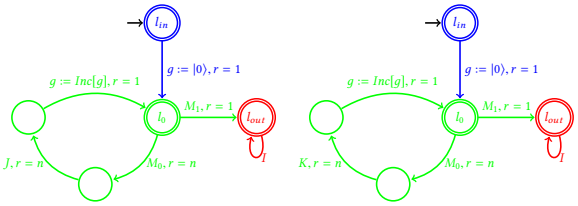


Figure 4. SVTSs for TF and HS, and the transition-bounding function f for HS.

Method	TF		HS_{noisy}	
	CPU time	Error bound	CPU time	Error bound
Accurate calculation	5951	0	31428	0
Method via Theorem 6.9	0.4800	0.0862	2.910	0.2448
Method via Theorem 6.13	711.8	0.0881	2092	0.7201

Table 2. We benchmark three approaches to establish the approximate correctness of the variations of HS. The unit for the CPU time is millisecond (ms). The inductive offset tracking method utilizes Theorem 6.9, and the superoperator distance bound method utilizes Theorem 6.13. The analytical bound in Section 6.3 gives 1.2149-partial correctness for TF using HS's Hoare triple.

where $\Lambda = \max_j \left\|H_j\right\|_\infty$. Thus, for any $|\psi\rangle$, it holds that $\models_{par,\epsilon} \{I \otimes |\psi\rangle\langle\psi|\} TF\{|n\rangle\langle n| \otimes (e^{-itH}|\psi\rangle\langle\psi|e^{itH})\}$, where $\epsilon = \frac{2L^2\Lambda^2t^2}{n} e^{\frac{L\Lambda t}{n}}$.

7 Evaluations

In this section, we present the results of our numerical experiments.

7.1 Invariant generation by solving SDPs

Our first experiment is to compare the performance of computing the Hoare triples in space $\mathcal{H}$ and applying rule Collapse to the master Hoare triple in the extended space $\mathcal{H} \otimes \mathcal{A}$. In particular, we measure the running time of invariant generation for RUS programs in Example 3.2, using the following methods.

1. Normal: solving the SDP in (4.2.6) with 1,000 random postconditions.
2. Master+collapse: solving the SDP in (5.2.1) with postcondition $\Phi_{\mathcal{H}\mathcal{A}}$ for the master Hoare triple, followed by the application of rule Collapse in (5.1.6) with the same 1,000 postconditions.

Our test cases are listed in Appendix E. We implement our methods using CVX, a MATLAB-based modeling system for convex optimization on iMac with Intel(R) Core(TM) i5-7500 Processor@3.4GHz and 16GB RAM. To demonstrate the reuse of master Hoare triples, we repeat the same program for $n = 1, 2, 7, 10$ times.

As shown in Table 1, we show that our master+collapse method significantly improves the running time over the normal method when calculating a large amount of triples of the same program. For example, our method achieves 51 times speedup for computing 1,000 random Hoare triples of 10 RUS_3, and 143 times speedup for RUS_2.

Our improvement is due to a preprocessing phase which contains all the information to derive any Hoare triple. Upon obtaining the master Hoare triple, a weakest precondition can be derived by applying a collapse operator $\mathfrak{C}$, which can be computed by applying partial trace[1] and matrix multiplication. This collapse step is less time consuming than recomputing the SDP directly. On the contrary, using the normal method, preprocessing seems impossible, and thus we must construct an SDP for each postcondition.

For computing repeating programs, we expect that the speedup can be significantly better since we only need to compute $\mathfrak{C}$ sequentially, though in practice the speedup may depend on the implementation of the SDP solver. On the other hand, using the normal method, since the invariants for a subprogram cannot be precomputed in general, one must construct a large SDP problem which contains all the constraints from every subprogram.

7.2 Approximate invariant generation

We present the significant efficiency advantages of our methods compared to accurate invariants calculation by experimenting their efficiencies and bounded errors on the Hamiltonian simulation problem. We set up the Hamiltonian simulation problem with 3 qubits and 5 steps. Two programs are evaluated: 1) the TF program introduced in Section 6.3. 2) HS_{noisy}, where we add three random noise channels $\mathcal{N}$. We benchmark our two methods in Section 6.2, and compare them with the accurate results.

[1]We use the MATLAB program for computing partial trace at http://www.dr-qubit.org/matlab/TrX.m.

The experiment details can be found in Appendix E.3, and the benchmark results are in Table 2. The error bounds from superoperator distance bound method are larger than the bounds obtained from inductive offset tracking method in our experiments. For TF program, they provide similar error bounds. For HS_{noisy} program, the superoperator distance bound methods gives much larger bounds, due to the fact that the noise channel affects largely on the diamond norm.

Acknowledgments

SH, SZ, and XW were supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Testbed Pathfinder Program under Award Number DE-SC0019040. YP and XW were also supported in part by DoD AFOSR MURI No. FA9550-16-1-0082.

References

[1] 2018. Google AI Quantum team. https://github.com/quantumlib/Cirq.
[2] 2018. Rigetti Forest team. .https://www.rigetti.com/forest.
[3] Ali Javadi Abhari, Arvin Faruque, Mohammad Javad Dousti, Lukas Svec, Oana Catu, Amlan Chakrabati, Chen-Fu Chiang, Seth Vanderwilt, John Black, Fred Chong, Margaret Martonosi, Martin Suchara, Ken Brown, Massoud Pedram, and Todd Brun. 2012. *Scaffold: Quantum Programming Language.* Technical Report TR-934-12. Princeton University.
[4] Gadi Aleksandrowicz, Thomas Alexander, P Barkoutsos, L Bello, Y Ben-Haim, D Bucher, FJ Cabrera-Hernández, J Carballo-Franquis, A Chen, CF Chen, et al. 2019. Qiskit: An open-source framework for quantum computing. *Accessed on: Mar* 16 (2019).
[5] Alexandru Baltag and Sonja Smets. 2011. Quantum Logic as a Dynamic Logic. *Synthese* 179, 2 (2011).
[6] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. 2015. Hamiltonian Simulation with Nearly Optimal Dependence on All Parameters. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS) (FOCS '15)*. IEEE Computer Society, Washington, DC, USA, 792–809. https://doi.org/10.1109/FOCS.2015.54
[7] Alex Bocharov, Martin Roetteler, and Krysta M Svore. 2015. Efficient synthesis of universal repeat-until-success quantum circuits. *Physical review letters* 114, 8 (2015), 080502.
[8] Olivier Brunet and Philippe Jorrand. 2004. Dynamic Quantum Logic for Quantum Programs. *International Journal of Quantum Information* 2, 1 (2004).
[9] Rohit Chadha, Paulo Mateus, and Amílcar Sernadas. 2006. Reasoning About Imperative Quantum Programs. *Electronic Notes in Theoretical Computer Science* 158 (2006).
[10] Andrew M. Childs, Dmitri Maslov, Yunseong Nam, Neil J. Ross, and Yuan Su. 2018. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences* 115, 38 (2018), 9456–9461. https://doi.org/10.1073/pnas.1801723115 arXiv:https://www.pnas.org/content/115/38/9456.full.pdf
[11] Ellie D'Hondt and Prakash Panangaden. 2006. Quantum Weakest Preconditions. *Mathematical Structures in Computer Science* 16, 3 (2006).
[12] Yuan Feng, Runyao Duan, Zhengfeng Ji, and Mingsheng Ying. 2007. Proof Rules for the Correctness of Quantum Programs. *Theoretical Computer Science* 386, 1-2 (2007).
[13] Cormac Flanagan, Cormac Flanagan, K. Rustan M. Leino, Mark Lillibridge, Greg Nelson, James B. Saxe, and Raymie Stata. 2002. Extended Static Checking for Java. In *Proceedings of the ACM SIGPLAN 2002 Conference on Programming Language Design and Implementation (PLDI '02)*. ACM, New York, NY, USA, 234–245. https://doi.org/10.1145/512529.512558
[14] Simon J. Gay. 2006. Quantum Programming Languages: Survey and Bibliography. *Mathematical Structures in Computer Science* 16, 4 (2006).
[15] Jonathan Grattage. 2005. A Functional Quantum Programming Language. In *LICS*.
[16] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. 2013. Quipper: A Scalable Quantum Programming Language. In *PLDI*.
[17] C.A.R. Hoare. 2009. Viewpoint: Retrospective: An Axiomatic Basis for Computer Programming. *Commun. ACM* 52, 10 (Oct. 2009), 30–32. https://doi.org/10.1145/1562764.1562779
[18] C. A. R. Hoare. 1969. An Axiomatic Basis for Computer Programming. *Commun. ACM* 12, 10 (Oct. 1969), 576–580. https://doi.org/10.1145/363235.363259
[19] Shih-Han Hung, Kesha Hietala, Shaopeng Zhu, Mingsheng Ying, Michael Hicks, and Xiaodi Wu. 2019. Quantitative Robustness Analysis of Quantum Programs. *Proc. ACM Program. Lang.* 3, POPL, Article 31 (Jan. 2019), 29 pages. https://doi.org/10.1145/3290344
[20] Nathaniel Johnston. 2016. QETLAB: A MATLAB toolbox for quantum entanglement, version 0.9. http://qetlab.com. https://doi.org/10.5281/zenodo.44637
[21] Yoshihiko Kakutani. 2009. A Logic for Formal Verification of Quantum Programs. In *ASIAN*.
[22] Thomas Kleymann. 1999. Hoare Logic and Auxiliary Variables. *Form. Asp. Comput.* 11, 5 (Dec. 1999), 541–566. https://doi.org/10.1007/s001650050057
[23] Yangjia Li and Mingsheng Ying. 2017. Algorithmic Analysis of Termination Problems for Quantum Programs. *Proc. ACM Program. Lang.* 2, POPL, Article 35 (Dec. 2017), 29 pages. https://doi.org/10.1145/3158123
[24] Seth Lloyd. 1996. Universal Quantum Simulators. *Science* 273, 5278 (1996), 1073–1078. https://doi.org/10.1126/science.273.5278.1073 arXiv:https://science.sciencemag.org/content/273/5278/1073.full.pdf
[25] Carlos Navarrete-Benlloch. 2015. An introduction to the formalism of quantum information. *arXiv preprint arXiv:1504.05270* (2015).
[26] Bernhard Ömer. 2003. *Structured Quantum Programming.* Ph.D. Dissertation. Vienna University of Technology.
[27] Adam Paetznick and Krysta M Svore. 2013. Repeat-Until-Success: Non-deterministic decomposition of single-qubit unitaries. *arXiv preprint arXiv:1311.1074* (2013).
[28] Adam Paetznick and Krysta M Svore. 2014. Repeat-until-success: non-deterministic decomposition of single-qubit unitaries. *Quantum Information & Computation* 14, 15-16 (2014), 1277–1301.
[29] Jennifer Paykin, Robert Rand, and Steve Zdancewic. 2017. QWIRE: A Core Language for Quantum Circuits. In *POPL*.
[30] Amr Sabry. 2003. Modeling Quantum Computing in Haskell. In *The Haskell Workshop*.
[31] Jeff W. Sanders and Paolo Zuliani. 2000. Quantum Programming. In *MPC*.
[32] Peter Selinger. 2004. A Brief Survey of Quantum Programming Languages. In *FLOPS*.
[33] Peter Selinger. 2004. Towards a Quantum Programming Language. *Mathematical Structures in Computer Science* 14, 4 (2004).
[34] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. 2018. Q#: Enabling Scalable Quantum Computing and Development with a High-level DSL. In *RWDSL*.
[35] Dominique Unruh. 2019. Quantum Hoare Logic with Ghost Variables. *arXiv preprint arXiv:1902.00325* (2019).

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

[37] John Watrous. 2009. Semidefinite programs for completely bounded norms. *arXiv preprint arXiv:0901.4709* (2009).

[38] John Watrous. 2018. *The Theory of Quantum Information*. Cambridge University Press.

[39] John Watrous. 2018. *The Theory of Quantum Information*. Cambridge University Press. https://doi.org/10.1017/9781316848142

[40] Nathan Wiebe and Martin Roetteler. 2014. Quantum arithmetic and numerical analysis using Repeat-Until-Success circuits. *arXiv preprint arXiv:1406.2040* (2014).

[41] Mingsheng Ying. 2011. Floyd–Hoare Logic for Quantum Programs. *ACM Transactions on Programming Languages and Systems* 33, 6 (2011).

[42] Mingsheng Ying. 2016. *Foundations of Quantum Programming*. Morgan Kaufmann.

[43] Mingsheng Ying. 2019. Toward automatic verification of quantum programs. *Formal Aspects of Computing* 31, 1 (01 Feb 2019), 3–25. https://doi.org/10.1007/s00165-018-0465-3

[44] Mingsheng Ying, Shenggang Ying, and Xiaodi Wu. 2017. Invariants of Quantum Programs: Characterisations and Generation. In *POPL*.

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

A Quantum information theory

This section is mostly taken from [19] with minor changes.

A.1 Basic definitions and notations

For any finite integer n, an n-dimensional Hilbert space $\mathcal{H}$ is essentially the space $\mathbb{C}^n$ of complex vectors. We use Dirac's notation, $|\psi\rangle$, to denote a complex vector in $\mathbb{C}^n$. The inner product of two vectors $|\psi\rangle$ and $|\phi\rangle$ is denoted by $\langle\psi|\phi\rangle$, which is the product of the Hermitian conjugate of $|\psi\rangle$, denoted by $\langle\psi|$, and vector $|\phi\rangle$. The norm of a vector $|\psi\rangle$ is denoted by $\||\psi\rangle\| = \sqrt{\langle\psi|\psi\rangle}$.

We define (linear) *operators* as linear mappings between Hilbert spaces. Operators between n-dimensional Hilbert spaces are represented by $n \times n$ matrices. For example, the identity operator $I_{\mathcal{H}}$ can be identified by the identity matrix on $\mathcal{H}$. The Hermitian conjugate of operator A is denoted by $A^\dagger$. Operator A is *Hermitian* if $A = A^\dagger$. The trace of an operator A is the sum of the entries on the main diagonal, i.e., $\mathrm{tr}(A) = \sum_i A_{ii}$. We write $\langle\psi|A|\psi\rangle$ to mean the inner product between $|\psi\rangle$ and $A|\psi\rangle$. A Hermitian operator A is *positive semidefinite* (resp., *positive definite*) if for all vectors $|\psi\rangle \in \mathcal{H}$, $\langle\psi|A|\psi\rangle \geq 0$ (resp., > 0).

A.2 Quantum States

The state space of a quantum system is a Hilbert space. The state space of a *qubit*, or quantum bit, is a 2-dimensional Hilbert space. One important orthonormal basis of a qubit system is the *computational* basis with $|0\rangle = (1,0)^\dagger$ and $|1\rangle = (0,1)^\dagger$, which encode the classical bits 0 and 1 respectively. Another important basis, called the $\pm$ basis, consists of $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The state space of multiple qubits is the *tensor product* of single qubit state spaces. For example, classical 00 can be encoded by $|0\rangle \otimes |0\rangle$ (written $|0\rangle|0\rangle$ or even $|00\rangle$ for short) in the Hilbert space $\mathbb{C}^2 \otimes \mathbb{C}^2$. The Hilbert space for an m-qubit system is $(\mathbb{C}^2)^{\otimes m} \cong \mathbb{C}^{2^m}$.

A *pure* quantum state is represented by a unit vector, i.e.,$|\psi\rangle$ with $\||\psi\rangle\| = 1$. A *mixed* state can be represented by a classical distribution over an ensemble of pure states $\{(p_i, |\psi_i\rangle)\}_i$, i.e., the system is in state $|\psi_i\rangle$ with probability p_i. One can also use *density operators* to represent both pure and mixed quantum states. A density operator ρ representing the ensemble $\{(p_i, |\psi_i\rangle)\}_i$ is a positive semidefinite operator $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$, where $|\psi_i\rangle\langle\psi_i|$ is the outer-product of $|\psi_i\rangle$; in particular, a pure state $|\psi\rangle$ can be identified with the density operator $\rho = |\psi\rangle\langle\psi|$. Note that $\mathrm{tr}(\rho) = 1$ holds for all density operators. A positive semidefinite operator ρ on $\mathcal{H}$ is said to be a *partial* density operator if $\mathrm{tr}(\rho) \leq 1$. The set of partial density operators is denoted by $\mathcal{D}(\mathcal{H})$.

A.3 Quantum Operations

Operations on quantum systems can be characterized by unitary operators. An operator U is *unitary* if its Hermitian conjugate is its own inverse, i.e., $U^\dagger U = UU^\dagger = I$. For a pure state $|\psi\rangle$, a unitary operator describes an *evolution* from $|\psi\rangle$ to $U|\psi\rangle$. For a density operator ρ, the corresponding evolution is $\rho \mapsto U\rho U^\dagger$. Common single-qubit unitary operators include

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{A.3.1}$$

The *Hadamard* operator H transforms between the computational and the $\pm$ basis. For example, $H|0\rangle = |+\rangle$ and $H|1\rangle = |-\rangle$. The *Pauli X* operator is a bit flip, i.e., $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. The *Pauli Z* operator is a phase flip, i.e., $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$.

More generally, the evolution of a quantum system can be characterized by an *admissible superoperator* $\mathcal{E}$, which is a *completely-positive* and *trace-non-increasing* linear map from $\mathcal{D}(\mathcal{H})$ to $\mathcal{D}(\mathcal{H}')$ for Hilbert spaces $\mathcal{H}, \mathcal{H}'$. A superoperator is positive if it maps from $\mathcal{D}(\mathcal{H})$ to $\mathcal{D}(\mathcal{H}')$ for Hilbert spaces $\mathcal{H}, \mathcal{H}'$. A superoperator $\mathcal{E}$ is k-positive if for any k-dimensional Hilbert space $\mathcal{A}$, the superoperator $\mathcal{E} \otimes I_{\mathcal{A}}$ is a positive map on $\mathcal{D}(\mathcal{H} \otimes \mathcal{A})$. A superoperator is said to be completely positive if it is k-positive for any positive integer k. A superoperator $\mathcal{E}$ is trace-non-increasing if for any initial state $\rho \in \mathcal{D}(\mathcal{H})$, the final state $\mathcal{E}(\rho) \in \mathcal{D}(\mathcal{H}')$ after applying $\mathcal{E}$ satisfies $\mathrm{tr}(\mathcal{E}(\rho)) \leq \mathrm{tr}(\rho)$.

For every superoperator $\mathcal{E} : \mathcal{D}(\mathcal{H}) \to \mathcal{D}(\mathcal{H}')$, there exists a set of Kraus operators $\{E_k\}_k$ such that $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$ for any input $\rho \in \mathcal{D}(\mathcal{H})$. Note that the set of Kraus operators is finite if the Hilbert space is finite-dimensional. The *Kraus form* of $\mathcal{E}$ is written as $\mathcal{E} = \sum_k E_k \circ E_k^\dagger$. A unitary evolution can be represented by the superoperator $\mathcal{E} = U \circ U^\dagger$. An identity operation refers to the superoperator $\mathcal{I}_{\mathcal{H}} = I_{\mathcal{H}} \circ I_{\mathcal{H}}$. The Schrödinger-Heisenberg *dual* of a superoperator $\mathcal{E} = \sum_k E_k \circ E_k^\dagger$, denoted by $\mathcal{E}^*$, is another superoperator such that for every state $\rho \in \mathcal{D}(\mathcal{H})$ and any operator A, $\mathrm{tr}(A\mathcal{E}(\rho)) = \mathrm{tr}(\mathcal{E}^*(A)\rho)$. The Kraus form of $\mathcal{E}^*$ is $\sum_k E_k^\dagger \circ E_k$.

A.4 Quantum Measurements

The way to extract information about a quantum system is called a quantum *measurement*. A quantum measurement on a system over Hilbert space $\mathcal{H}$ can be described by a set of linear operators $\{M_m\}_m$ with $\sum_m M_m^\dagger M_m = I_{\mathcal{H}}$. If we perform a measurement $\{M_m\}$ on a state ρ, the outcome m is observed with probability $p_m = \mathrm{tr}(M_m \rho M_m^\dagger)$ for each m. A major difference between classical and quantum computation is that a quantum measurement changes the state. In particular, after a measurement yielding outcome m, the state collapses to $M_m \rho M_m^\dagger / p_m$. For example, a measurement in the computational basis is described by $M = \{M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|\}$. If we perform the computational basis measurement M on state $\rho = |+\rangle\langle +|$, then with probability $\frac{1}{2}$ the outcome is 0

and ρ becomes $|0\rangle\langle 0|$. With probability $\frac{1}{2}$ the outcome is 1 and ρ becomes $|1\rangle\langle 1|$.

B The syntax and the semantics of quantum programs

This section is mostly taken from [19] with minor changes.

B.1 Syntax

Let *Var* be the set of quantum variables. The symbol q represents a metavariable ranging over quantum variables and a *quantum register* $\overline{q}$ is a finite set of distinct variables. For each $q \in Var$, denote its state space by $\mathcal{H}_q$. The quantum register $\overline{q}$ is associated with the Hilbert space $\mathcal{H}_{\overline{q}} = \bigotimes_{q\in\overline{q}} \mathcal{H}_q$. If $type(q) = \mathbf{Bool}$ then $\mathcal{H}_q$ is the Hilbert space with basis $\{|0\rangle, |1\rangle\}$. If $type(q) = \mathbf{Int}$ then $\mathcal{H}_q$ has the basis $\{|n\rangle : n \in \mathbb{Z}\}$. The syntax of a *quantum* ***while*** *program* P is defined as follows.

$$\begin{aligned} P \;::=\;& \mathbf{skip} \mid q := |0\rangle \mid \overline{q} := U[\overline{q}] \mid P_1; P_2 \mid \\ & \mathbf{case}\ M[\overline{q}] = \overline{m \to P_m}\ \mathbf{end} \mid \\ & \mathbf{while}\ M[\overline{q}] = 1\ \mathbf{do}\ P_1\ \mathbf{done}. \end{aligned} \tag{B.1.1}$$

The language constructs above are similar to their classical counterparts. (1) **skip** does nothing. (2) $q := |0\rangle$ sets quantum variable q to the basis state $|0\rangle$. (3) $\overline{q} := U[\overline{q}]$ applies the unitary U to the qubits in $\overline{q}$. (4) Sequencing has the same behavior as its classical counterpart. (5) **case** $M[\overline{q}] = \overline{m \to P_m}$ **end** performs the measurement $M = \{M_m\}$ on the qubits in $\overline{q}$, and executes program P_m if the outcome of the measurement is m. The bar over $\overline{m \to P_m}$ indicates that there may be one or more repetitions of this expression. [2] (6) **while** $M[\overline{q}] = 1$ **do** P_1 **done** performs the measurement $M = \{M_0, M_1\}$ on the qubits in $\overline{q}$, and executes P_1 if measurement produces the outcome corresponding to M_1 or terminates if measurement produces the outcome corresponding to M_0.

We highlight two differences between quantum and classical while languages: (1) Qubits may only be initialized to the basis state $|0\rangle$. There is no quantum analogue for initialization to any expression (i.e. $x := e$) because of the *no-cloning* theorem of quantum states. Any state $|\psi\rangle \in \mathcal{H}_q$, however, can be constructed by applying some unitary U to $|0\rangle$. (2) Evaluating the guard of a case statement or loop, which performs a measurement, potentially disturbs the state of the system.

B.2 Operational Semantics

The *operational* semantics of the quantum **while**-language are presented in Figure 5a. $\langle P,\ \rho\rangle \to \langle P',\ \rho'\rangle$, where $\langle P,\ \rho\rangle$ and $\langle P',\ \rho'\rangle$ are quantum *configurations*. In configurations, P (or P') could be a quantum program or the empty program E, and ρ and ρ' are partial density operators representing the current state. Intuitively, in one step, we can evaluate program P on input state ρ to program P' (or E) and output state ρ'. In order to present the rules in a non-probabilistic manner, the probabilities associated with each transition are encoded in the output partial density operator.[3]

In the *Initialization* rule, the superoperators $\mathcal{E}^{\text{bool}}_{q\to 0}(\rho)$ and $\mathcal{E}^{\text{int}}_{q\to 0}(\rho)$, which initialize the variable q in ρ to $|0\rangle\langle 0|$, are defined by $\mathcal{E}^{\text{bool}}_{q\to 0}(\rho) = |0\rangle_q\langle 0|\rho|0\rangle_q\langle 0| + |0\rangle_q\langle 1|\rho|1\rangle_q\langle 0|$ and $\mathcal{E}^{\text{int}}_{q\to 0}(\rho) = \sum_{n=-\infty}^{\infty} |0\rangle_q\langle n|\rho|n\rangle_q\langle 0|$. Here, $|\psi\rangle_q\langle\phi|$ denotes the outer product of states $|\psi\rangle$ and $|\phi\rangle$ associated with variable q; that is, $|\psi\rangle$ and $|\phi\rangle$ are in $\mathcal{H}_q$ and $|\psi\rangle_q\langle\phi|$ is a matrix over $\mathcal{H}_q$. It is a convention in the quantum information literature that when operations or measurements only apply to part of the quantum system (e.g., a subset of quantum variables of the program), one should assume that an identity operation is applied to the rest of quantum variables. [4] The identity operation is usually omitted for simplicity.

We do not explain the rules in detail, but hope their meaning is self-evident given the description of the language in Section B.1.

B.2.1 Denotational Semantics

The *denotational* semantics of a quantum **while** program is given in Figure 5b (with more details in Ying [41, 42]). It defines $[\![P]\!]$ as a superoperator on $\rho \in \mathcal{H}_{Var}$ [42]. The semantics of each term is given compositionally. We write $\mathbf{while}^{(k)}$ for the kth syntactic approximation (i.e., unrolling) of **while** and $\bigsqcup$ for the *least upper bound* operator in the complete partial order generated by Löwner comparison. We connect the denotational semantics to the operational semantics through the following proposition.

Proposition B.1 ([42]). *For any program P*

$$[\![P]\!]\rho \equiv \sum \{|\rho' : \langle P, \rho\rangle \to^* \langle E, \rho'\rangle|\}, \tag{B.2.1}$$

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

In short, the meaning of running program P on input state ρ is the sum of all possible output states, weighted by their probabilities.

C Summary of notations

We summarize the notations used in this paper in Table 3.

[2] Our syntax for conditional/case statements differs from Ying [42] to make it clear that there are multiple programs P_m.

[3] If we had instead considered a probabilistic transition system, then the transition rule for case statements could have been written as $\langle \mathbf{case}\ M[\overline{q}] = \overline{m \to P_m}\ \mathbf{end}, \rho\rangle \xrightarrow{p_m} \langle P_m, \rho_m\rangle$ where $p_m = \text{tr}(M_m \rho M_m^\dagger)$ and $\rho_m = M_m \rho M_m^\dagger / p_m$.

[4] For example, applying $|\psi\rangle_q\langle\phi|$ to ρ means applying $|\psi\rangle_q\langle\phi| \otimes I_{\mathcal{H}_{\bar{q}}}$ to ρ, where $\bar{q}$ denotes the set of all variables except q.

(Skip) $\overline{\langle \textbf{skip},\ \rho\rangle \to \langle E,\ \rho\rangle}$

(Initialization) $\overline{\langle q := |0\rangle,\ \rho\rangle \to \langle E,\ \rho_0^q\rangle}$

where $\rho_0^q = \begin{cases} \mathcal{E}^{\text{bool}}_{q\to 0}(\rho) & \text{if } type(q) = \textbf{Bool} \\ \mathcal{E}^{\text{int}}_{q\to 0}(\rho) & \text{if } type(q) = \textbf{Int} \end{cases}$

(Unitary) $\overline{\langle \overline{q} := U[\overline{q}],\ \rho\rangle \to \langle E,\ U\rho U^\dagger\rangle}$

(Sequence E) $\overline{\langle E; P_2,\ \rho\rangle \to \langle P_2,\ \rho\rangle}$

(Sequence) $\dfrac{\langle P_1,\ \rho\rangle \to \langle P_1',\ \rho'\rangle}{\langle P_1; P_2,\ \rho\rangle \to \langle P_1'; P_2,\ \rho'\rangle}$

(Case m) $\overline{\langle \textbf{case } M[\overline{q}] = \overline{m \to P_m} \textbf{ end},\ \rho\rangle \to \langle P_m,\ M_m\rho M_m^\dagger\rangle}$

for each outcome m of measurement $M = \{M_m\}$

(While 0) $\overline{\langle \textbf{while } M[\overline{q}] = 1 \textbf{ do } P_1 \textbf{ done},\ \rho\rangle \to \langle E,\ M_0\rho M_0^\dagger\rangle}$

(While 1) $\overline{\langle \textbf{while } M[\overline{q}] = 1 \textbf{ do } P_1 \textbf{ done},\ \rho\rangle \to \langle P_1; \textbf{while } M[\overline{q}] = 1 \textbf{ do } P_1 \textbf{ done},\ M_1\rho M_1^\dagger\rangle}$

(a)

$$
\begin{array}{lcl}
[\![\textbf{skip}]\!]\rho & = & \rho \\
[\![q := |0\rangle]\!]\rho & = & \begin{cases} \mathcal{E}^{\text{bool}}_{q\to 0}(\rho) & \text{if } type(q) = \textbf{Bool} \\ \mathcal{E}^{\text{int}}_{q\to 0}(\rho) & \text{if } type(q) = \textbf{Int} \end{cases} \\
[\![\overline{q} := U[\overline{q}]]\!]\rho & = & U\rho U^\dagger \\
[\![P_1; P_2]\!]\rho & = & [\![P_2]\!]([\![P_1]\!]\rho) \\
[\![\textbf{case } M[\overline{q}] = \overline{m \to P_m} \textbf{ end}]\!]\rho & = & \sum_m [\![P_m]\!](M_m\rho M_m^\dagger) \\
[\![\textbf{while } M[\overline{q}] = 1 \textbf{ do } P_1 \textbf{ done}]\!]\rho & = & \bigsqcup_{k=0}^{\infty} [\![\textbf{while}^{(k)}]\!]\rho
\end{array}
$$

(b)

Figure 5. quantum **while** programs: (a) operational semantics (b) denotational semantics.

Hilbert Spaces:	$\mathcal{H}, \mathcal{A}$	$L(\mathcal{H})$ (Linear operators on $\mathcal{H}$)
States:	(pure states)	$\|\psi\rangle, \|\phi\rangle$ (metavariables); $\|0\rangle, \|1\rangle, \|+\rangle, \|-\rangle$ (notable states)
	(density operators)	ρ, σ (metavariables); $\|\psi\rangle\langle\psi\|$ (as outer product)
Operations:	(unitaries)	U, V (metavariables); H, X, Y, Z (notable operations)
	(superoperators)	$\mathcal{E}$ (general); Φ (quantum channels)
Measurements:	M	$\{M_m\}_m$ (general); $\{M_0 = \|0\rangle\langle 0\|, M_1 = \|1\rangle\langle 1\|\}$ (example)
Quantum Predicates	(in space $\mathcal{H}$)	P, Q
	(in space $\mathcal{H} \otimes \mathcal{A}$)	$\mathcal{P}, \mathcal{Q}$
Hoare triples	(total correctness)	$\models_{tot} \{P\}S\{Q\}, \models_{tot} \{\mathcal{P}\}S\{\mathcal{Q}\}$
	(partial correctness)	$\models_{par} \{P\}S\{Q\}, \models_{par} \{\mathcal{P}\}S\{\mathcal{Q}\}$

Table 3. A brief summary of notation used in this paper

D Proofs of technical lemmas

D.1 Proof of Lemma 4.3

Proof of Lemma 4.3. We prove the claim by induction on the structure of S.

1. For $S \equiv \textbf{skip}, \overline{q} := U[\overline{q}], q := |0\rangle$: the claim is straightforward since $\Pi = \left\{ l_{in}^S \xrightarrow{[\![S]\!]} l_{out}^S \right\}$.
2. For $S \equiv S_1; S_2$: let Π_i be the set of prime paths of SVTS defined by S_i for $i \in \{1, 2\}$. Since the only incoming

location to $l_{in}^{S_2}$ is $l_{out}^{S_1}$, the set of prime paths Π defined by S is

$$\Pi = \{l_{in}^{S_1} \overset{\pi_1}{\Rightarrow} l_{out}^{S_1} \xrightarrow{I} l_{in}^{S_2} \overset{\pi_2}{\Rightarrow} l_{out}^{S_2} : \pi_1 \in \Pi_1, \pi_2 \in \Pi_2\}. \tag{D.1.1}$$

Thus by the inductive hypothesis,

$$\mathcal{E}_{\Pi_i}(\rho) = \sum_{\pi_i \in \Pi_i} \mathcal{E}_{\pi_i}(\rho) = [\![S_i]\!](\rho) \tag{D.1.2}$$

for every state ρ, we know that

$$\begin{aligned}\mathcal{E}_{\Pi}(\rho) &= \sum_{\pi_1,\pi_2} \mathcal{E}_{\pi_2}(\rho)(\mathcal{E}_{\pi_1}(\rho)) \\ &= [\![S_2]\!]\left(\sum_{\pi_1} \mathcal{E}_{\pi_1}(\rho)\right) \\ &= [\![S_2]\!]([\![S_1]\!](\rho)) = [\![S]\!](\rho).\end{aligned} \tag{D.1.3}$$

3. For $S \equiv$ **case** $M[q] = m \to S_m$ **end**: let Π_m be the set of prime paths from $l_{in}^{S_m} = l_m$ to $l_{out}^{S_m}$. The set of prime paths for S is then

$$\Pi = \{l_{in}^{S} \xrightarrow{M_m} l_m \overset{\pi_m}{\Longrightarrow} l_{out}^{S} : \forall m,\ \pi_m \in \Pi_m\}. \tag{D.1.4}$$

Then by the inductive hypothesis, since

$$\mathcal{E}_{\Pi_m}(\rho) = \sum_{\pi_m \in \Pi_m} \mathcal{E}_{\pi_m}(\rho) = [\![S_m]\!](\rho), \tag{D.1.5}$$

we know that

$$\begin{aligned}\mathcal{E}_{\Pi}(\rho) &= \sum_m \sum_{\pi_m \in \Pi_m} M_m \mathcal{E}_{\pi}(\rho) M_m^\dagger \\ &= \sum_m M_m [\![S_m]\!](\rho) M_m^\dagger = [\![S]\!](\rho).\end{aligned} \tag{D.1.6}$$

4. For $S \equiv$ **while** $M[q] = 1$ **do** S_1 **done**: Let Π_1 be the set of prime paths from $l_{in}^{S_1}$ to $l_{out}^{S_1}$. The set of prime paths for S truncated to k passes to $l_{in}^{S_1}$ is then

$$\begin{aligned}\Pi^{(k)} = \Big\{ & l_{in}^{S} \xrightarrow{M_1} l_{in}^{S_1} \overset{\pi^{(1)}}{\Longrightarrow} l_{out}^{S_1} \xrightarrow{I} l_{in}^{S} \xrightarrow{M_1} \dots \to \\ & l_{in}^{S} \xrightarrow{M_1} l_{in}^{S_1} \overset{\pi^{(n)}}{\Longrightarrow} l_{out}^{S_1} \xrightarrow{I} l_{in}^{S} \xrightarrow{M_0} l_{out}^{S} : \\ & 0 \le n \le k,\ \pi^{(1)}, \dots, \pi^{(k)} \in \Pi_1 \Big\}.\end{aligned} \tag{D.1.7}$$

By (D.1.7), the set can also be recursively written as

$$\Pi^{(k+1)} = \{l_{in}^{S} \xrightarrow{M_1} l_{in}^{S_1} \overset{\pi}{\Rightarrow} l_{out}^{S_1} \xrightarrow{I} l_{in}^{S} \overset{\gamma}{\Rightarrow} l_{out}^{S} : \pi \in \Pi_1, \gamma \in \Pi^{(k)}\}. \tag{D.1.8}$$

By the inductive hypothesis, $\sum_{\pi \in \Pi_1} \mathcal{E}_\pi(\rho) = [\![S_1]\!](\rho)$. The set $\Pi = \bigsqcup_{k=0}^{\infty} \Pi^{(k)}$. We show that $\mathcal{E}_{\Pi^{(k)}}(\rho) = \sum_{i=0}^{k} M_0 \mathcal{E}^i(\rho) M_0^\dagger$ for each $k \in \mathbb{N}$ and $\mathcal{E}(\rho) = [\![S_1]\!](M_1 \rho M_1^\dagger)$. The base case $k = 0$ is straightforward since $\Pi^{(0)} = \{l_{in}^{S} \xrightarrow{M_0} l_{out}^{S}\}$. Suppose that the statement holds for $k = n$. Then by (D.1.8),

$$\begin{aligned}\mathcal{E}_{\Pi^{(n+1)}}(\rho) &= \sum_{\pi \in \Pi_1} \sum_{\gamma \in \Pi^{(k)}} \mathcal{E}_\gamma(\mathcal{E}_\pi(M_1 \rho M_1^\dagger)) \\ &= \sum_{\gamma \in \Pi^{(k)}} \mathcal{E}_\gamma([\![S_1]\!](M_1 \rho M_1^\dagger)) \\ &= \sum_{\gamma \in \Pi^{(k)}} \mathcal{E}_\gamma(\mathcal{E}(\rho)) \\ &= \mathcal{E}_{\Pi^{(k)}}(\mathcal{E}(\rho)) \\ &= \sum_{i=0}^{k+1} M_0 \mathcal{E}^i(\rho) M_0^\dagger.\end{aligned} \tag{D.1.9}$$

Taking $k \to \infty$, we conclude the proof.

□

D.2 Proof of Theorem 4.4

Proof of Theorem 4.4. By definition, the set of prime paths from l_{in}^{S} to l_{out}^{S} is

$$\Pi = \left\{\pi : l_{in}^{S} \overset{\pi}{\Rightarrow} l_{out}^{S} : l_{out}^{S} \text{ is visited only once}\right\}. \tag{D.2.1}$$

By the definitions of additive invariant and partial correctness, it suffices to show that $\mathcal{E}_\Pi = [\![S]\!]$. By Lemma 4.3, we conclude the proof. □

D.3 Solving the SDP in Example 3.2

In this section, we derive an analytic solution to the SDP problem in (4.2.6). In Figure 1, we take l_0, l_1, l_2 to be our cut-points. The SDP problem corresponding to *RUS* can be formulated using Invariant-SDP 4.1. The positive semidefinite constraints are

$$O_0 \sqsubseteq \text{RST}^*(W^\dagger O_1 W) \tag{D.3.1}$$

and

$$O_1 \sqsubseteq M_0^\dagger O_2 M_0 + M_1^\dagger V^\dagger O_1 V M_1, \tag{D.3.2}$$

where $V := WRW^\dagger = V^\dagger$ and $\text{RST} = [\![p := |0^m\rangle]\!]$.

We show that $(O_1, O_2) = (|\Psi\rangle\langle\Psi| + |\Psi^\perp\rangle\langle\Psi^\perp|, |\Psi\rangle\langle\Psi|)$ is a solution to (D.3.2) and the equality holds. In the 2-dimensional subspace $\text{span}\{|\Psi\rangle, |\Psi^\perp\rangle\}$, the unitary V is a reflection about the state

$$|\Upsilon\rangle = \sqrt{p}|\Psi\rangle + \sqrt{1-p}|\Psi^\dagger\rangle. \tag{D.3.3}$$

Therefore $V^\dagger(|\Psi\rangle\langle\Psi| + |\Psi^\perp\rangle\langle\Psi^\perp|)V = |\Psi\rangle\langle\Psi| + |\Psi^\perp\rangle\langle\Psi^\perp|$. Then the equality holds since

$$\begin{aligned}&|\Psi\rangle\langle\Psi| + M_1^\dagger V^\dagger(|\Psi\rangle\langle\Psi| + |\Psi^\perp\rangle\langle\Psi^\perp|)V M_1 \\ &\quad = |\Psi\rangle\langle\Psi| + M_1^\dagger(|\Psi\rangle\langle\Psi| + |\Psi^\perp\rangle\langle\Psi^\perp|)M_1 \\ &\quad = |\Psi\rangle\langle\Psi| + |\Psi^\perp\rangle\langle\Psi^\perp|.\end{aligned} \tag{D.3.4}$$

Next we show that

$$(O_0, O_1) = (I_m \otimes |\psi\rangle\langle\psi|, |\Psi\rangle\langle\Psi| + |\Psi^\perp\rangle\langle\Psi^\perp|) \tag{D.3.5}$$

is a solution to (D.3.1). First we observe that

$$\begin{aligned} &W^\dagger(|\Psi\rangle\langle\Psi| + |\Psi^\perp\rangle\langle\Psi^\perp|)W \\ &\quad = W^\dagger(|\Upsilon\rangle\langle\Upsilon| + |\Upsilon^\perp\rangle\langle\Upsilon^\perp|)W \\ &\quad = (|0,\psi\rangle\langle 0,\psi| + W^\dagger|\Upsilon^\perp\rangle\langle\Upsilon^\perp|W), \end{aligned} \tag{D.3.6}$$

where $|\Upsilon^\perp\rangle := -\sqrt{1-p}|\Psi\rangle + \sqrt{p}|\Psi^\perp\rangle$, because

$$\mathrm{span}\{|\Psi\rangle, |\Psi^\perp\rangle\} = \mathrm{span}\{|\Upsilon\rangle, |\Upsilon^\perp\rangle\}. \tag{D.3.7}$$

Then we have

$$\begin{aligned} &\textsc{rst}^*(|0^m,\psi\rangle\langle 0^m,\psi| + W^\dagger|\Upsilon^\perp\rangle\langle\Upsilon^\perp|W) \\ &\quad = I_m \otimes (\langle 0^m| \otimes I)(|0^m,\psi\rangle\langle 0^m,\psi| \\ &\qquad\qquad + W^\dagger|\Upsilon^\perp\rangle\langle\Upsilon^\perp|W)(|0^m\rangle \otimes I) \\ &\quad = I_m \otimes |\psi\rangle\langle\psi|. \end{aligned} \tag{D.3.8}$$

since by the proof of [27, Corollary 3.2],

$$(\langle 0^m| \otimes I)(W^\dagger|\Upsilon^\perp\rangle\langle\Upsilon^\perp|W)(|0^m\rangle \otimes I) = 0. \tag{D.3.9}$$

Finally the solution implies that

$$\{I_m \otimes |\psi\rangle\langle\psi|\}RUS\{|0^m, U\psi\rangle\langle 0^m, U\psi|\}, \tag{D.3.10}$$

the intuition to which is provided as follows: to yield a quantum state $|0^m, U\psi\rangle$ that satisfies the postcondition $|0^m, U\psi\rangle\langle 0^m, U\psi|$, the input state must be any quantum state of the form $\rho_p \otimes |\psi\rangle\langle\psi|_q$, since the reset command resets ρ to the zero state, and the semantics of the while loop, along with the initial application of W, is a unitary U when the state on p is a zero state.

D.4 Proof of Lemma 5.4

First we recall some facts which will be used in the proof of Lemma 5.4.

Fact D.1 ([39]). *The maximally entangled state $|\Phi\rangle$ and the Choi-Jamiołkowski isomorphism satisfies the following properties:*

$$(1)\ (M \otimes I)|\Phi\rangle = \left(I \otimes M^T\right)|\Phi\rangle, \tag{D.4.1}$$

$$(I \otimes M)|\Phi\rangle = \left(M^T \otimes I\right)|\Phi\rangle, \forall M; \tag{D.4.2}$$

$$(2)\ \mathcal{E}(\rho) = \mathrm{tr}_{\mathcal{H}}\left(J(\mathcal{E})(I_{\mathcal{H}'} \otimes \rho^T)\right), \forall \rho \in \mathcal{D}(\mathcal{H}); \tag{D.4.3}$$

$$(3)\ J(\mathcal{E}^*) = (J(\mathcal{E}))^T, \text{ for any channel } \mathcal{E}. \tag{D.4.4}$$

Now we prove the lemma.

Proof of Lemma 5.4.

1. By definition $\mathfrak{C}_R(\cdot)$ is a completely positive map due to, e.g., [39, Theorem 2.22]. Thus,
$$\mathfrak{C}_R(\mathcal{P} - \mathcal{Q}) \sqsubseteq 0 \Rightarrow \mathfrak{C}_R(\mathcal{P}) \sqsubseteq \mathfrak{C}_R(\mathcal{Q}).$$
2. By definition, we have
$$\begin{aligned} \mathfrak{C}_R(\Phi) &= d_{\mathcal{H}}\mathrm{tr}_A((I_{\mathcal{H}} \otimes \sqrt{R^T})|\Phi\rangle\langle\Phi|(I_{\mathcal{H}} \otimes \sqrt{R^T})) \\ &= d_{\mathcal{H}}\mathrm{tr}_A((\sqrt{R} \otimes I_{\mathcal{A}})|\Phi\rangle\langle\Phi|(\sqrt{R} \otimes I_{\mathcal{A}})) \\ &= R, \end{aligned} \tag{D.4.5}$$
where the second equality holds by (D.4.2).
3. This follows from the cyclic property of the partial trace when the operator (in our case, $I_{\mathcal{H}} \otimes \sqrt{R}$) acts as the identity on the non-traced subspaces (see, e.g., [25, Section 6.4.1]).

□

D.5 Proof of Lemma 5.6

Proof of Lemma 5.6. By (5.1.4), we know that

$$wp.S.\Phi = \mathcal{J}([\![S]\!]^*) = [\![S]\!]^*(\Phi). \tag{D.5.1}$$

Applying the map $\mathfrak{C}_Q(\cdot)$, we have

$$\begin{aligned} &\mathfrak{C}_Q(\mathcal{J}([\![S]\!]^*)) \\ &\quad = \mathfrak{C}_Q(([\![S]\!]^* \otimes I_{\mathcal{A}})\Phi) \\ &\quad = d_{\mathcal{H}}\mathrm{tr}_{\mathcal{A}}((I_{\mathcal{H}} \otimes \sqrt{Q^T})(([\![S]\!]^* \otimes I_{\mathcal{A}})\Phi)(I_{\mathcal{H}} \otimes \sqrt{Q^T})) \\ &\quad = d_{\mathcal{H}}[\![S]\!]^*\mathrm{tr}_{\mathcal{A}}((I_{\mathcal{H}} \otimes \sqrt{Q^T})\Phi(I_{\mathcal{H}} \otimes \sqrt{Q^T})) \qquad \text{(D.5.2)} \\ &\quad = [\![S]\!]^*(\mathfrak{C}_Q(\Phi)) = [\![S]\!]^*(Q). \qquad \text{(D.5.3)} \end{aligned}$$

Since the equality holds, $\mathfrak{C}_Q(\mathcal{J}([\![S]\!]^*))$ is the weakest precondition. It remains to show that the equality in (D.5.2) holds. Let $[\![S]\!] = \sum_k E_k \circ E_k^\dagger$ be the Kraus form. We have

$$\begin{aligned} &\mathrm{tr}_{\mathcal{A}}((I_{\mathcal{H}} \otimes \sqrt{Q^T})(([\![S]\!]^* \otimes I_{\mathcal{A}})\Phi)(I_{\mathcal{H}} \otimes \sqrt{Q^T})) \\ &\quad = \sum_k \mathrm{tr}_{\mathcal{A}}((E_k \otimes \sqrt{Q^T})\Phi(E_k^\dagger \otimes \sqrt{Q^T})) \\ &\quad = \sum_i \sum_k \left(E_k^\dagger \otimes (\langle i|\sqrt{Q^T})\right)\Phi\left(E_k^{\otimes}(\sqrt{Q^T}|i\rangle)\right) \\ &\quad = \sum_k E_k^\dagger\left[\sum_i \left(I_{\mathcal{H}} \otimes (\langle i|\sqrt{Q^T})\right)\Phi\left(I_{\mathcal{H}} \otimes (\sqrt{Q^T}|i\rangle)\right)\right]E_k \\ &\quad = \sum_k E_k^\dagger\left[\mathrm{tr}_{\mathcal{A}}((I_{\mathcal{H}} \otimes \sqrt{Q^T})\Phi(I_{\mathcal{H}} \otimes \sqrt{Q^T}))\right]E_k \\ &\quad = [\![S]\!]^*\left(\mathrm{tr}_{\mathcal{A}}((I_{\mathcal{H}} \otimes \sqrt{Q^T})\Phi(I_{\mathcal{H}} \otimes \sqrt{Q^T}))\right). \end{aligned} \tag{D.5.4}$$

□

D.6 Proof of Lemma 6.12

Proof of Lemma 6.12. Before we start our proof, we introduce several handy inequalities.

- For any density operator ρ, superoperators $\mathcal{E}, \mathcal{E}', \mathcal{E}_1, \mathcal{E}_1' \ldots, \mathcal{E}_k, \mathcal{E}_k'$ such that $\sum_{j=1}^k \mathcal{E}_j \sqsubseteq I$ and $\sum_{j=1}^k \mathcal{E}_j' \sqsubseteq I$, there is

$$\begin{aligned}
&\sum_j \left\|\mathcal{E}_j(\mathcal{E}(\rho)) - \mathcal{E}_j'(\mathcal{E}'(\rho))\right\|_1 \\
&= 2\sum_j \mathrm{tr}(Q_j(\mathcal{E}_j(\mathcal{E}(\rho)) - \mathcal{E}_j'(\mathcal{E}(\rho)) \\
&\quad + \mathcal{E}_j'(\mathcal{E}(\rho)) - \mathcal{E}_j'(\mathcal{E}(\rho)))) \\
&\le 2\sum_j \mathrm{tr}(\mathcal{E}_j^*(Q_j)(\mathcal{E}(\rho) - \mathcal{E}'(\rho)) \\
&\quad + 2\sum_j \mathrm{tr}(Q_j(\mathcal{E}_j - \mathcal{E}_j')(\mathcal{E}(\rho))) \\
&\le \|\mathcal{E}(\rho) - \mathcal{E}'(\rho)\|_1 \\
&\quad + 2\sum_j \mathrm{tr}(\mathcal{E}(\rho))\mathrm{tr}\left(Q_j(\mathcal{E}_j - \mathcal{E}_j')\left(\frac{\mathcal{E}(\rho)}{\mathrm{tr}(\mathcal{E}(\rho))}\right)\right) \\
&\le \|\mathcal{E}(\rho) - \mathcal{E}'(\rho)\|_1 \\
&\quad + \mathrm{tr}(\mathcal{E}(\rho))\sum_j \left\|(\mathcal{E}_j - \mathcal{E}_j')\left(\frac{\mathcal{E}(\rho)}{\mathrm{tr}(\mathcal{E}(\rho))}\right)\right\|_1 \\
&\le 2\|\mathcal{E} - \mathcal{E}'\|_\diamond \\
&\quad + 2\mathrm{tr}(\mathcal{E}(\rho))\sum_j \left\|\mathcal{E}_j - \mathcal{E}_j'\right\|_\diamond, \qquad \text{(D.6.1)}
\end{aligned}$$

  where in the intermediate steps, $0 \sqsubseteq Q_j \sqsubseteq I$ is the predicate to reach $2\mathrm{tr}(Q_j(\mathcal{E}_j(\mathcal{E}(\rho)) - \mathcal{E}_j'(\mathcal{E}'(\rho)))) = \left\|\mathcal{E}_j(\mathcal{E}(\rho)) - \mathcal{E}_j'(\mathcal{E}'(\rho))\right\|_1$.
- For any prime path set Π, we have

$$\begin{aligned}
&\left\|\mathcal{E}_\Pi - \mathcal{E}_\Pi'\right\|_\diamond \\
&= \max_{\rho \in \mathcal{H}\otimes\mathcal{A}} \left\|\mathcal{E}_\Pi \otimes I_{\mathcal{A}}(\rho) - \mathcal{E}_\Pi' \otimes I_{\mathcal{A}}(\rho)\right\|_1 \\
&\le \max_{\rho \in \mathcal{H}\otimes\mathcal{A}} \sum_{\pi\in\Pi} \left\|\mathcal{E}_\pi \otimes I_{\mathcal{A}}(\rho) - \mathcal{E}_\pi' \otimes I_{\mathcal{A}}(\rho)\right\|_1 \\
&\le \sum_{\pi\in\Pi} \left\|\mathcal{E}_\pi - \mathcal{E}_\pi'\right\|_\diamond \qquad \text{(D.6.2)}
\end{aligned}$$

Now we find the upper bound of $\left\|\mathcal{E}_\Pi - \mathcal{E}_\Pi'\right\|_\diamond$ for any prime path set Π from l_{in} to l. We track the generation of S from P. In the generation of S, nodes are only added and renamed, so we can always find l in the generation procedure. We inductively prove the following three facts simultaneously:

- For any prime path set Π with paths starting from l_{in}, there is:

$$\sum_{\pi\in\Pi} \left\|\mathcal{E}_\pi - \mathcal{E}_\pi'\right\|_\diamond \le \sum_{\tau\in\mathcal{T}} r(\tau)\left\|\mathcal{E}_\tau - \mathcal{E}_\tau'\right\|_\diamond. \qquad \text{(D.6.3)}$$

- For any Π with paths from l_{in} to l, there is:

$$\sum_{\pi\in\Pi} \left\|\mathcal{E}_\pi - \mathcal{E}_\pi'\right\|_\diamond \le \sum_{\tau\in\mathcal{T}_l} r(\tau)\left\|\mathcal{E}_\tau - \mathcal{E}_\tau'\right\|_\diamond. \qquad \text{(D.6.4)}$$

- $\mathcal{T}_{l_{out}} = \mathcal{T}$.

Note that if l is not in current SVTS, the second inequality is meaningless, and we ignore it. Here we start induction.

- $P \equiv \mathbf{skip}, \bar{q} := U[\bar{q}], q := |0\rangle$: The transition ends at l_{out}, and inequalities holds naturally.
- $P \equiv P_1; P_2$: Let S_1, S_2 be the SVTSs for P_1, P_2, and r_1, r_2 be the repetition functions for P_1, P_2. Because in S_2, $l_{in}^{P_2}$ can reach l_{out}, every transition in $\mathcal{T}$ can reach l_{out}. To prove the inequality (D.6.3) in this case, consider a prime path set Π starting from l_{in}. For any $\pi \in \Pi$, if it contains $l_{out}^{P_1}$, we can divide it into two parts separated by $l_{out}^{P_1}$. Denote $U(\pi)$ the part of π from l_{in} to $l_{out}^{P_1}$. If π does not contain $l_{out}^{P_1}$, we set $U(\pi) = \pi$. We can pick out $U(\Pi) = \{U(\pi) : \pi \in \Pi\}$, and the succeeding path set $V(\Pi, \pi_u) = \{\pi_v : \pi_u + \pi_v \in \Pi\}$, where path addition is the concatenation. We can see if Π is prime, $V(\Pi, \pi_u)$ is prime. Thus we have $\mathcal{E}_\Pi = \sum_{\pi_u\in U(\Pi)}\{|\mathcal{E}_{V(\Pi,\pi_u)} \circ \mathcal{E}_{\pi_u}|\}$, and so is $\mathcal{E}_\Pi'$. Thus, for any $\rho \in \mathcal{H}\otimes\mathcal{A}$, applying inequality (D.6.1) and induction hypothesis, we have

$$\begin{aligned}
&\sum_{\pi\in\Pi} \left\|\mathcal{E}_\pi(\rho) - \mathcal{E}_\pi'(\rho)\right\|_1 \qquad \text{(D.6.5)} \\
&\le \sum_{\pi_u\in U(\Pi)} 2\left\|\mathcal{E}_{\pi_u} - \mathcal{E}_{\pi_u}'\right\|_\diamond \\
&\quad + \sum_{\pi_u\in U(\Pi)} 2\mathrm{tr}[\mathcal{E}_{\pi_u}(\rho)] \sum_{\pi_v\in V(\Pi,\pi_u)} \left\|\mathcal{E}_{\pi_v} - \mathcal{E}_{\pi_v}'\right\|_\diamond \qquad \text{(D.6.6)} \\
&\le 2\left(\sum_{\pi_u\in U(\Pi)} \mathrm{tr}[\mathcal{E}_{\pi_u}(\rho)]\right) \sum_{\tau\in\mathcal{T}^{P_2}} r(\tau)\left\|\mathcal{E}_\tau - \mathcal{E}_\tau'\right\|_\diamond \\
&\quad + 2\sum_{\pi_u\in U(\Pi)} \left\|\mathcal{E}_{\pi_u} - \mathcal{E}_{\pi_u}'\right\|_\diamond \qquad \text{(D.6.7)} \\
&\le 2\sum_{\tau\in\mathcal{T}} r(\tau)\left\|\mathcal{E}_\tau - \mathcal{E}_\tau'\right\|_\diamond. \qquad \text{(D.6.8)}
\end{aligned}$$

  Taking all $\rho \in \mathcal{H}\otimes\mathcal{A}$, inequality (D.6.3) holds. For the inequality (D.6.4), we find where l is. If l is in S_1, every path in Π cannot go beyond $l_{out}^{P_1}$, because there is no way back. Thus we only need to employ inductive hypothesis. If l is in S_2, $V(\Pi, \pi_u)$ is a path set with paths from $l_{in}^{P_2}$ to l, thus the inductive hypothesis for the inequality (D.6.4) applies. Applying the similar arguments as above, with $\mathcal{T}_l = \mathcal{T}_l^{P_2} \cup \mathcal{T}^{P_1}$, we finish the proof. If l is not in S_1 nor S_2, the inequality is meaningless.
- $P \equiv \mathbf{case}\ (M[q] = m \to P_m)\ \mathbf{end}$: In this case, any path π in a prime path set Π from l_{in} to l_{out} starts with a choice of branches. Denote Out the set of transitions outgoint from l_{in}, $\Pi_\tau = \{\pi : \pi = l_{in} \xrightarrow{\tau} \cdots\}$, and $V(\pi)$ the path without the first transition. Any transition $\tau \in Out$ can reach the entry of corresponding subprogram, so it can reach l_{out}. For each branch, the proof of the sequential case applys here. Thus for any $\rho \in \mathcal{H}\otimes\mathcal{A}$,

applying the inequality D.6.1,

$$\begin{aligned}
&\sum_{\pi\in\Pi}\left\|\mathcal{E}_\pi(\rho)-\mathcal{E}'_\pi(\rho)\right\|_1\\
&=\sum_{\tau\in Out}\sum_{\pi\in\Pi_\tau}\left\|\mathcal{E}_\pi(\rho)-\mathcal{E}'_\pi(\rho)\right\|_1\\
&\le\sum_{\tau\in Out}\sum_{\pi\in\Pi_\tau}\left\|\mathcal{E}_{V(\pi)}(\mathcal{E}_\tau(\rho))-\mathcal{E}'_{V(\pi)}(\mathcal{E}'_\tau(\rho))\right\|_1\\
&\le 2\sum_{\tau\in\mathcal{T}} r(\tau)\left\|\mathcal{E}_\tau-\mathcal{E}'_\tau\right\|_\diamond. \qquad \text{(D.6.9)}
\end{aligned}$$

As for the second inequality, we find the branch in which l locates, then the corresponding first transition τ in any path to l is the only new transition in $\mathcal{T}_l$. By similar argument as above, we can show the correctness.

- $P\equiv$ **while** $M[\bar{q}]=1$ **do** Q **done**. Let τ_0 be the transition with measurement result 0 starting from l_{in}, and τ_1 be the one with measurement result 1. If $l\neq l_{out}$, because it forms a loop, every transition except the transition τ_0 with M_0 is in $\mathcal{T}_l$. If $l=l_{out}$, $\mathcal{T}_l=\mathcal{T}$. We assume the loop is (a,n)-bounded. For any path π in any prime path set Π, we let $C(\pi,k)$ be the prefix of π that reaches l_{in}^Q for exactly k times, $h(\pi)$ be the number of l_{in}^Q in path π, and $h(\Pi)=\max_{\pi\in\Pi}h(\pi)$. Set $r(\tau)=\frac{n}{1-a}f^Q(\tau)$. First we show that, for any $k\ge 0$, we have for any prime Π with paths starts from l_{in} such that $h(\Pi)\le k$, and for any density operator $\rho\in\mathcal{H}\otimes\mathcal{A}$, the following inequality holds

$$\begin{aligned}
&\sum_{\pi\in\Pi}\left\|\mathcal{E}_\pi(\rho)-\mathcal{E}'_\pi(\rho)\right\|_1\\
\le&2\left\|\mathcal{E}_{\tau_0}-E'_{\tau_0}\right\|_\diamond+2k\left\|\mathcal{E}_{\tau_1}-E'_{\tau_1}\right\|_\diamond\\
&+2\sum_{\tau\in\mathcal{T}\setminus\{\tau_0,\tau_1\}} kf^Q(\tau)\left\|\mathcal{E}_\tau-\mathcal{E}'_\tau\right\|_\diamond. \qquad \text{(D.6.10)}
\end{aligned}$$

We employ induction here. The base case is $k=0$, so the only possible path is $\pi_0=(\tau_0)$, and the basic inequality holds. Then we assume that the inequality holds for $k-1$. Consider $U(\Pi)=\{C(\pi,1):\pi\in\Pi\}$. We can apply arguments similar to the sequential case to show the correctness, by bounding the terms in $U(\Pi)$ using the inequality (D.6.3)for Q, and bounding the rest terms using the inductive hypothesis. Now we show the correctness of the inequality (D.6.3) for the **while** case. We employ induction on $\lfloor\frac{h(\Pi)-1}{n}\rfloor$ to prove that for any density operator ρ, there is

$$\sum_{\tau\in\mathcal{T}}\left\|\mathcal{E}_\pi(\rho)-\mathcal{E}'_\pi(\rho)\right\|_1\le 2\sum_{\tau\in\mathcal{T}} r(\tau)\left\|\mathcal{E}_\tau-\mathcal{E}'_\tau\right\|_\diamond. \qquad \text{(D.6.11)}$$

For the base case, where $h(\Pi)<n$, it is indeed the above inequality (D.6.10). We assume that the inequality holds for $k-1$. The definition of (a,n)-boundedness shows that if $\mathcal{E}_{bound}(\rho)=[\![Q]\!](M_1\rho M_1^\dagger)$, for any density operator $\rho\in\mathcal{H}\otimes\mathcal{A}$, there is

$$\mathrm{tr}(M_1\mathcal{E}^n_{bound}(\rho)M_1^\dagger)\le a\,\mathrm{tr}(M_1\rho M_1^\dagger)\le a. \qquad \text{(D.6.12)}$$

Set $U(\Pi)=\{C(\pi,n):\pi\in\Pi\}$, so there is

$$\begin{aligned}
&\sum_{\pi_u\in U(\Pi)}\mathrm{tr}[\mathcal{E}_{\pi_u}(\rho)] && \text{(D.6.13)}\\
=&\mathrm{tr}[\mathcal{E}_{U(\Pi)}(\rho)] && \text{(D.6.14)}\\
\le&\mathrm{tr}[M_1\mathcal{E}^n_{bound}(\rho)M_1^\dagger]\le a. && \text{(D.6.15)}
\end{aligned}$$

Thus for any $\rho\in\mathcal{H}\otimes\mathcal{A}$:

$$\begin{aligned}
&\sum_{\pi\in\Pi}\left\|\mathcal{E}_\pi(\rho)-\mathcal{E}'_\pi(\rho)\right\|_1 && \text{(D.6.16)}\\
\le&\sum_{\pi_u\in U(\Pi)}2\left\|\mathcal{E}_{\pi_u}-\mathcal{E}'_{\pi_u}\right\|_\diamond\\
&+2\sum_{\pi_u\in U(\Pi)}\mathrm{tr}[\mathcal{E}_{\pi_u}(\rho)]\sum_{\pi_v\in V(\Pi,\pi_u)}\left\|\mathcal{E}_{V(\Pi,\pi_u)}-\mathcal{E}'_{V(\Pi,\pi_u)}\right\|_\diamond && \text{(D.6.17)}\\
\le&2\sum_{\pi_u\in U(\Pi)}\left\|\mathcal{E}_{\pi_u}-\mathcal{E}'_{\pi_u}\right\|_\diamond\\
&+2\left(\sum_{\pi_u\in U(\Pi)}\mathrm{tr}[\mathcal{E}_{\pi_u}(\rho)]\right)\sum_{\tau\in\mathcal{T}} r(\tau)\left\|\mathcal{E}_\tau-\mathcal{E}'_\tau\right\|_\diamond) && \text{(D.6.18)}\\
\le&2n\left\|\mathcal{E}_{\tau_1}-\mathcal{E}'_{\tau_1}\right\|_\diamond+\sum_{\tau\in\mathcal{T}^Q}2nf^Q(\tau)\left\|\mathcal{E}_\tau-\mathcal{E}'_\tau\right\|_\diamond\\
&+2a\sum_{\tau\in\mathcal{T}} r(\tau)\left\|\mathcal{E}_\tau-\mathcal{E}'_\tau\right\|_\diamond && \text{(D.6.19)}\\
\le&2\sum_{\tau\in\mathcal{T}} r(\tau)\left\|\mathcal{E}_\tau-\mathcal{E}'_\tau\right\|_\diamond. && \text{(D.6.20)}
\end{aligned}$$

Maximizing it over $\rho\in\mathcal{H}\otimes\mathcal{A}$ gives the result for **while** case.

Thus we have the final result. □

D.7 Proof of Theorem 6.13

Proof of Theorem 6.13. Since the operations are linear, we may consider an arbitrary normalized state ρ. We claim that for any density operator $\rho\in\mathcal{D}(\mathcal{H})$ with unit trace, and prime path set Π with paths from l_{in} to l, we claim that

$$\mathrm{tr}[\Theta\rho]\le 1-\mathrm{tr}[\mathcal{E}'_\Pi(\rho)]+\mathrm{tr}[O\mathcal{E}'_\Pi(\rho)]+\|\mathcal{E}_\Pi-\mathcal{E}'_\Pi\|_\diamond. \qquad \text{(D.7.1)}$$

To prove this claim, we start from the definition of invariant without approximation, which gives:

$$\begin{aligned}\mathrm{tr}[\Theta\rho] \le& 1 - \mathrm{tr}[\mathcal{E}_\Pi(\rho)] + \mathrm{tr}[O\mathcal{E}_\Pi(\rho)] && \text{(D.7.2)}\\ =& 1 + \mathrm{tr}[(O-I)\mathcal{E}'_\Pi(\rho)] + \mathrm{tr}[(O-I)(\mathcal{E}_\Pi(\rho) - \mathcal{E}'_\Pi(\rho))] && \text{(D.7.3)}\\ \le& 1 - \mathrm{tr}[\mathcal{E}'_\Pi(\rho)] + \mathrm{tr}[O\mathcal{E}'_\Pi(\rho)] + \frac{1}{2}\left\|\mathcal{E}_\Pi(\rho) - \mathcal{E}'_\Pi(\rho)\right\|_1 && \text{(D.7.4)}\\ \le& 1 - \mathrm{tr}[\mathcal{E}'_\Pi(\rho)] + \mathrm{tr}[O\mathcal{E}'_\Pi(\rho)] + \|\mathcal{E}_\Pi - \mathcal{E}'_\Pi\|_\diamond. && \text{(D.7.5)}\end{aligned}$$

This gives the proof of the claim. Then directly applying Lemma 6.12 shows the correctness of the theorem. □

D.8 Proof of Lemma 6.8

Proof of Lemma 6.8. We claim that for any finite prime path set Π starting at l_{in}, there is

$$\sum_{\pi\in\Pi} \mathcal{E}'^*_\pi(I - \eta(l_\pi)) \sqsubseteq I - \eta(l_{in}) + \sum_{\pi\in P(\Pi)} \mathcal{E}'^*_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I). \tag{D.8.1}$$

Here for path π, $P(\pi)$ is the set of π's non-empty prefixes, and $P(\Pi) = \cup_{\pi\in\Pi} P(\pi)$. τ_π is the ending transition in the path π, and $\pi\setminus\{\tau_\pi\}$ the path π without the last transition.

Following [44], we define the length of the path as $h(\pi)$, and $h(\Pi) = \max_{\pi\in\Pi} h(\pi)$. We prove this result by induction on $h(\Pi)$. For the base case, $h(\Pi) = 0$, hence $\Pi = \phi$. By $0 \sqsubseteq \eta(l_{in}) \sqsubseteq I$, the ineqaulity holds. Now consider the case that $h(\Pi) = k+1$. Then Π can be decomposed as $\Pi = \Delta \cup \left(\cup_{\sigma:h(\sigma)=k}\Pi_\sigma\right)$, where $h(\Delta) = k$, σ is a valid path, and $\Pi_\sigma \subset \Pi$ contains all the paths whose prefix is σ. For any σ, we have

$$\begin{aligned} I - \eta(l_\sigma) \sqsupseteq& \sum_{\tau\in\Omega_{l_\sigma}} \mathcal{E}^*_\tau(I - \eta(l_\tau)) && \text{(D.8.2)}\\ \sqsupseteq& \sum_{\pi\in\Pi_\sigma} \mathcal{E}^*_{\tau_\pi}(I - \eta(l_\pi)) && \text{(D.8.3)}\\ \sqsupseteq& \sum_{\pi\in\Pi_\sigma} \left(\mathcal{E}'^*_{\tau_\pi}(I - \eta(l_\pi)) - \delta_{\tau_\pi} I\right). && \text{(D.8.4)}\\ \mathcal{E}'^*_\sigma(I - \eta(l_\sigma)) \sqsupseteq& \sum_{\pi\in\Pi_\sigma} \left(\mathcal{E}'^*_\pi(I - \eta(l_\pi)) - \mathcal{E}'^*_\sigma(\delta_{\tau_\pi} I)\right). && \text{(D.8.5)}\end{aligned}$$

Notice that Δ has no intersection with $\Sigma = \{\sigma : h(\sigma) = k, \Pi_\sigma \neq\}$ because Π is prime, and $P(\Pi) = P(\Delta\cup\Sigma)\cup(\cup_{\sigma\in\Sigma}\Pi_\sigma)$.

By inductive hypothesis, we know

$$\begin{aligned} I - \eta(l_{in}) \sqsupseteq& \sum_{\pi\in\Delta} \mathcal{E}'^*_\pi(I - \eta(l_\pi)) + \sum_{\sigma\in\Sigma} \mathcal{E}'^*_\sigma(I - \eta(l_\sigma))\\ &- \sum_{\pi\in P(\Delta\cup\Sigma)} \mathcal{E}'^*_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I) && \text{(D.8.6)}\\ \sqsupseteq& \sum_{\pi\in\Pi} \mathcal{E}'^*_\pi(I - \eta(l_\pi)) - \sum_{\sigma\in\Sigma}\sum_{\pi\in\Pi_\sigma} \mathcal{E}'^*_\sigma(\delta_{\tau_\pi} I)\\ &- \sum_{\pi\in P(\Delta\cup\Sigma)} \mathcal{E}'^*_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I) && \text{(D.8.7)}\\ \sqsupseteq& \sum_{\pi\in\Pi} \mathcal{E}'^*_\pi(I - \eta(l_\pi)) - \sum_{\pi\in P(\Pi)} \mathcal{E}'^*_{\pi\setminus\{\tau\}}(\delta_{\tau_\pi} I). && \text{(D.8.8)}\end{aligned}$$

Then we show for any l, any finite prime path set Π from l_{in} to l, there is

$$\sum_{\pi\in P(\Pi)} \mathcal{E}'^*_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I) \sqsubseteq \sum_{\tau\in\mathcal{T}_l} r'(\tau)\delta_\tau I. \tag{D.8.9}$$

Following the definition of repetition function, we prove it by inductino on the generation of SVTS. In the proof of Lemma 6.12, we have showed $\mathcal{T}_{l_{out}} = \mathcal{T}$ within the generation of SVTS. Notice that if we retrieve the final observable of the inductive assertion map for the intermediate locations in the generation, it still keeps the property of inductive assertion map in the intermediate state, so we can apply it with our generation.

- $P \equiv \mathbf{skip}, \bar{q} := U[\bar{q}], q := |0\rangle$: There is one transition, so the inequality holds naturally.
- $P \equiv P_1; P_2$: Let $\mathcal{S}_1, \mathcal{S}_2$ be the SVTSs for P_1, P_2, and r_1, r_2 be the repetition functions for P_1, P_2. If $l \in \mathcal{L}_1$, then inductive hypothesis concludes it. To prove the inequality when $l \in \mathcal{L}_2$, consider a prime path set Π. For any $\pi \in \Pi$, there is only one occurence of $l^{P_1}_{out}$. We divide the path set with $l^{P_1}_{out}$. Let $\Pi = \Delta \cup (\cup_{\sigma\in\Sigma}\Pi_\sigma)$, where Σ is the prefixes of Π ending at $l^{P_1}_{out}$, and $\Pi_\sigma \subseteq \Pi$ is set of paths whose prefixes include σ, and deleting σ at the beginning. Then for any σ, by the inductive hypothesis in $\mathcal{S}_2$, there is

$$\begin{aligned}&\sum_{\pi\in P(\Pi_\sigma)} \mathcal{E}'^*_\sigma \mathcal{E}'^*_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I) && \text{(D.8.10)}\\ \sqsupseteq& \sum_{\pi\in P(\Pi_\sigma)} \left(\mathcal{E}'^*_\sigma \mathcal{E}'^*_{\pi\setminus\tau_\pi}(\delta_{\tau_\pi} I) - r'(\tau_\pi)\delta_{\tau_\pi}\mathcal{E}'^*_\sigma(I)\right) && \text{(D.8.11)}\end{aligned}$$

With the inductive hypothesis in $\mathcal{S}_1$, we get

$$\sum_{\pi\in P(\Pi)} \mathcal{E}'^{*}_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I) \tag{D.8.12}$$

$$\sqsubseteq \sum_{\pi\in P(\Delta\cup\Sigma)} \mathcal{E}'^{*}_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I) + \sum_{\sigma\in\Sigma}\sum_{\pi\in P(\Pi_\sigma)} \mathcal{E}'^{*}_{\sigma}\mathcal{E}'^{*}_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I) \tag{D.8.13}$$

$$\sqsubseteq \sum_{\tau\in\mathcal{T}^{P_1}} r'_1(\tau)\delta_\tau I + \sum_{\sigma\in\Sigma}\mathcal{E}'^{*}_{\sigma}\Big(\sum_{\tau\in\mathcal{T}_l^{P_2}} r'_2(\tau)\delta_\tau I\Big) \tag{D.8.14}$$

$$\sqsubseteq \sum_{\tau\in\mathcal{T}_l} r'(\tau)\delta_\tau I. \tag{D.8.15}$$

- $P \equiv$ **case** $(M[q] = m \to P_m)$ **end**: In this case, any path π in a prime path set Π from l_{in} to l_{out} starts with a choice of branches. If $l = l_{in}$, then the inequality holds trivially. If $l \notin \{l_{in}, l_{out}\}$, then with it is the same case as sequential case. Thus the only non-trivial case is $l = l_{out}$. Denote the set of transitions outgoint from l_{in} as Out, the subset of Π with path starting with τ as Π_τ. We delete the first occurence of τ in Π_τ for convecience. Hence

$$\sum_{\pi\in P(\Pi)} \mathcal{E}'^{*}_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I) \tag{D.8.16}$$

$$\sqsubseteq \sum_{\tau\in Out} \mathcal{E}'^{*}_{\tau}\left(\sum_{\pi\in P(\Pi_\tau)} \mathcal{E}'^{*}_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I)\right) \tag{D.8.17}$$

$$\sqsubseteq \sum_{\tau\in\mathcal{T}} r'(\tau)\delta_\tau I. \tag{D.8.18}$$

- $P \equiv$ **while** $M[\bar{q}] = 1$ **do** Q **done**. Similar to the corresponding case in Lemma 6.12, we let τ_0 be the transition with measurement result 0 starting from l_{in}, and τ_1 be the one with measurement result 1. We let $C(\pi, k)$ be the prefix of π that reaches l_{in}^Q for exactly k times, $h(\pi)$ be the number of l_{in}^Q in path π, and $h(\Pi) = \max_{\pi\in\Pi} h(\pi)$.
  First we show that, for any $k \geq 0$, we have for any prime Π with paths starts from l_{in} such that $h(\Pi) \leq k$, there is

$$\sum_{\pi\in P(\Pi)} \mathcal{E}'^{*}_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I) \sqsubseteq \delta_{\tau_0} + k\delta_{\tau_1} + \sum_{\tau\in\mathcal{T}^Q} kr^Q(\tau)\delta_\tau I \tag{D.8.19}$$

  This inequality is the same as the sequential case applying for k times. Then we move on to the main inequality. We employ induction on $\lfloor \frac{h(\Pi)-1}{n} \rfloor$ to prove that for any prime path set Π, there is

$$\sum_{\pi\in P(\Pi)} \mathcal{E}'^{*}_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I) \sqsubseteq \delta_{\tau_0} + \frac{n}{1-a}\delta_{\tau_1} + \sum_{\tau\in\mathcal{T}^Q} \frac{n}{1-a} r^Q(\tau)\delta_\tau I. \tag{D.8.20}$$

  For the base case, where $h(\Pi) < n$, it is the previous inequality. We assume that the inequality holds for $k-1$. The definition of (a, n)-boundedness shows that if $\mathcal{E}_{bound}(\rho) = [\![Q]\!](M_1\rho M_1^\dagger)$ there is

$$(\mathcal{E}^*_{bound})^n \mathcal{E}^*_{M_1}(I) \sqsubseteq a\mathcal{E}^*_{M_1}(I). \tag{D.8.21}$$

  Set $U(\Pi) = \{C(\pi, n) : \pi \in \Pi\}$, we can divide Π into two parts: Δ to be those π wth $h(\pi) < n$, and the rest with their prefixes in $U(\Pi)$. This is similar to the sequential case.

$$\sum_{\pi\in P(\Pi)} \mathcal{E}'^{*}_{\pi\setminus\{\tau_\pi\}}(\delta_{\tau_\pi} I) \tag{D.8.22}$$

$$\sqsubseteq \sum_{\pi\in\Delta} \mathcal{E}'^{*}_{\pi\setminus\tau_\pi}(\delta_{\tau_\pi} I) + \sum_{\sigma\in U(\Pi)} \mathcal{E}'^{*}_{\sigma}\left(\sum_{\pi\in\Pi_\sigma} \mathcal{E}'^{*}_{\pi\setminus\tau_\pi}(\delta_{\tau_\pi} I)\right) \tag{D.8.23}$$

$$\sqsubseteq \delta_{\tau_0} + n\delta_{\tau_1} + \sum_{\tau\in\mathcal{T}^Q} nr^Q(\tau)\delta_\tau I + a\sum_{\pi\in\Pi_\sigma} \mathcal{E}^*_{M_1}\mathcal{E}'^{*}_{\pi\setminus\tau_\pi}(\delta_{\tau_\pi} I) \tag{D.8.24}$$

$$\sqsubseteq \delta_{\tau_0} + \frac{n}{1-a}\delta_{\tau_1} + \sum_{\tau\in\mathcal{T}^Q} \frac{n}{1-a} r^Q(\tau)\delta_\tau I \tag{D.8.25}$$

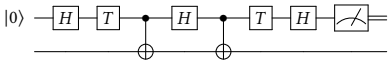
$$= \sum_{\tau\in\mathcal{T}} r'(\tau)\delta_\tau I. \tag{D.8.26}$$

This concludes the proof. □

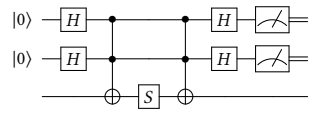
E Test cases of RUS programs

In our experiment in Section 7, we implement the unitary W, defined in (3.2.2), with the following quantum circuits. The test cases are randomly taken from the combination of the database of [28]. For each circuit, the zero state $|0\rangle$ specifies the qubits to reset in the beginning and the reflection operator in the loop body. The measurement specifies the condition of termination of the while loop.

E.1 The testcase RUS_2



E.2 The testcase RUS_3



E.3 Experiment details for HS

We set up the Hamiltonian simulation with step $N = 5$, time $t = 0.5$, a typical Hamiltonian $H = X\otimes Z\otimes I + I\otimes X\otimes Z + Z\otimes I\otimes X$ in Ising model and the starting state $|\psi\rangle = |000\rangle$. One kind of variations is the Trotter's formula program TF as described in Section 6.3. Notice that the order of sub-Hamiltonian evolutions matter in TF, while by symmetry we only consider one order for our H: $X \otimes Z \otimes I \to I \otimes X \otimes Z \to Z \otimes I \otimes X$.

Another variation $\mathrm{HS}_{\mathrm{noisy}}$ considers noisy channel in the *HS* program, by adding noise $\mathcal{N}_3$ to the working qubits at the end of K. Here we select random p_1, p_2, p_3 uniformly in $[0, 0.1]$, and construct $\mathcal{N}_3 = \otimes_{i=1}^{3}(0.9\mathcal{I} + p_i\mathcal{X} + (1 - p_i)\mathcal{Z})$ as a noise, where $\mathcal{X}, \mathcal{Z}$ are as described in Example 6.3. We repeat this selection for 50 times, and present the average statistics. Three approaches are applied and benchmarked: 1) directly calculating the accurate invariant of the programs; 2) ultilizing Theorem 6.9; 3) ultilizing Theorem 6.13 and direct calculation of diamond norm. The diamond is calculated using QETLAB [20].