Quantitative Robustness Analysis of Quantum Programs (Extended Version)

SHIH-HAN HUNG, KESHA HIETALA, and SHAOPENG ZHU, University of Maryland, College Park, USA
MINGSHENG YING, University of Technology Sydney, Australia, State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China, and Tsinghua University, China
MICHAEL HICKS and XIAODI WU, University of Maryland, College Park, USA

Quantum computation is a topic of significant recent interest, with practical advances coming from both research and industry. A major challenge in quantum programming is dealing with errors (quantum noise) during execution. Because quantum resources (e.g., qubits) are scarce, classical error correction techniques applied at the level of the architecture are currently cost-prohibitive. But while this reality means that quantum programs are almost certain to have errors, there as yet exists no principled means to reason about erroneous behavior. This paper attempts to fill this gap by developing a semantics for erroneous quantum while programs, as well as a logic for reasoning about them. This logic permits proving a property we have identified, called $\epsilon$-robustness, which characterizes possible "distance" between an ideal program and an erroneous one. We have proved the logic sound, and showed its utility on several case studies, notably: (1) analyzing the robustness of noisy versions of the quantum Bernoulli factory (QBF) and quantum walk (QW); (2) demonstrating the (in)effectiveness of different error correction schemes on single-qubit errors; and (3) analyzing the robustness of a fault-tolerant version of QBF.

CCS Concepts: • Theory of computation → Denotational semantics; Quantum information theory;

Additional Key Words and Phrases: quantum programming, quantum noise, approximate computing

ACM Reference Format:

1 INTRODUCTION
Quantum programming has been actively investigated for the past two decades. Early work on semantics and language design [Grattage 2005; Ömer 2003; Sabry 2003; Sanders and Zuliani 2000; Selinger 2004b] has been followed up, in the last few years, by the development of a number of mature languages, including Quipper [Green et al. 2013], Scaffold [Abhari et al. 2012], LIQU|⟩ [Wecker and Svore 2014], Q# [Svore et al. 2018], and QWIRE [Paykin et al. 2017]. Various program logics have also been extended for verification of quantum programs [Baltag and Smets 2011; Brunet and Jorrand 2004; Chadha et al. 2006; Feng et al. 2007; Kakutani 2009; Ying 2011; Ying et al. 2017]. For detailed surveys, see Selinger [2004a], Gay [2006], and Ying [2016].

Authors’ addresses: Shih-Han Hung; Kesha Hietala; Shaopeng Zhu, University of Maryland, College Park, USA; Mingsheng Ying, University of Technology Sydney, Australia; Michael Hicks; Xiaodi Wu, University of Maryland, College Park, USA.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2019 Copyright held by the owner/author(s).
2473-1421/2019/1-ART31
https://doi.org/10.1145/3290344
A major practical challenge in implementing quantum programs is dealing with errors (aka quantum noise) during execution. Most existing work on algorithms and programming languages assumes this problem will be solved by the hardware, as in classical computers, or with fault-tolerant protocols that are designed independently of any particular application [Chong et al. 2017]. As such, the semantics of programs is defined in a manner that ignores the possibility of errors [Green et al. 2013; Paykin et al. 2017; Wecker and Svore 2014].

Unfortunately, providing such a general-purpose, fault-tolerant quantum computing abstraction appears to be impractical for near-term quantum devices, for which precisely controllable qubits are expensive, error-prone, and scarce. Existing error correction techniques consume a substantial number of qubits, severely limiting the range of possible computations. For example, one logical qubit may require $10^3 - 10^4$ physical qubits [Fowler et al. 2012]. Furthermore, fault-tolerant operations on these logical qubits require many more physical operations than their non-fault-tolerant counterparts.

As such, research on practical quantum computation must focus on Noisy Intermediate-Scale Quantum (NISQ) computers (as phrased by Preskill [2018]), which will lack general-purpose fault tolerance. While some particular algorithms have been developed to reflect this reality [Moll et al. 2018; Peruzzo et al. 2014], there is as yet no principled method to reason about the error-affected performance of quantum applications. Such methods are needed to help guide the design of practical applications for near-term devices.

**Contributions.** This paper extends the quantum while-language [Ying 2011] with a semantics that accounts for the possibility of error, and defines an accompanying logic for reasoning about erroneous executions. Our work constitutes an alternative to the common, but impractical, one-size-fits-all approach to fault tolerance and instead elevates the question of errors to the level of the programming language. Our approach is inspired by the work of Carbin et al. [2013], which reasons about classical programs running on unreliable hardware. We make four main contributions.

First, we present the syntax and semantics (both operational and denotational) of the quantum while-language extended to include noisy operations. In particular, we modify unitary application to allow the noisy operation $\Phi$ (a superoperator) to occur with probability $p$. This approach permits modeling any local noise occurring during the execution of a quantum program, which is the standard noise model considered in the study of quantum error correction and fault-tolerant quantum computation [Gottesman 2010]. This error model is also used by experimental physicists for building and benchmarking quantum devices in both academia and industry.

Second, we define a notion of **quantum robustness**. In particular, we say that a noisy program $\bar{P}$ is $\epsilon$-robust under $(Q, \lambda)$ if it computes a quantum state at most $\epsilon$ distance away from that of that of its ideal equivalent $P$ when starting both from states satisfying quantum predicate $Q$ [D’Hondt and Panangaden 2006] to degree $\lambda$. Our definition makes use of the so-called diamond norm [Gilchrist et al. 2005] to account for the potential enlarging effect of entanglement on the distance. We generalize the diamond norm to what we call the $(Q, \lambda)$-diamond norm, which allows us to consider only input states that satisfy a quantum predicate $Q$ to degree $\lambda$. Doing so obtains more accurate bounds when considering specific quantum devices and/or knowledge of states owing to the use of classical control operators. We show that the $(Q, \lambda)$-diamond norm can be computed by a semidefinite program (SDP) by extending the algorithm of Watrous [2009].

Third, we define a logic for reasoning about quantum robustness, with the following judgment

$$
(Q, \lambda) + \bar{P} \leq \epsilon.
$$

1Our notation draws an analogy with the typing judgment $\Gamma \vdash P : t$. In particular, $Q$ and $\lambda$ are “assumptions” about inputs just as $\Gamma$ represents assumptions about input (their types); $\bar{P}$ is the program we are reasoning about; and $\epsilon$ is the proved robustness of this program, just as $t$ is the proved type of the program. Our rules are compositional like those of typing.
This judgment states that for any input state that satisfies a quantum predicate $Q$ to degree $\lambda$, the distance between $\|P\|$ and $\|P\|$ is then bounded by $\epsilon$.

We prove our logic is sound. A particular challenge is the rule for loops, owing to the termination problem first studied by Li and Ying [2018]. In particular, if the loop body generates some error and the loop does not terminate, it is hard to prove any non-trivial bound on the final accumulated error. To avoid this difficulty in the setting of approximate computing, Carbin et al. [2013] simply assume that the loop will terminate within a bounded number of iterations or a trivial upper bound will be applied. To capture more complicated cases, we introduce a concept called the $(a, n)$-boundedness of the loop. Intuitively, a loop is $(a, n)$-bounded if, for every input state, after $n$ iterations it is guaranteed that with probability at least $1 - a$ it has exited the loop. The probability is due to the quantum measurement in the loop guard. It is easy to see that $(a, n)$-boundedness implies the termination of the loop. Pleasantly, $(a, n)$ bounding the ideal loop is sufficient to reason about a noisy loop with any error model $\Phi$ in the loop body.

Our fourth and final contribution is to develop several case studies that demonstrate the utility of reasoning about errors at the program level.

- We start with important examples in the quantum while-language, the quantum Bernoulli factory (QBF) and the quantum walk (QW), and directly analyze the robustness of noisy versions $\overline{QBF}$ and $\overline{QW}$ using our logic. In the process, we prove the $(a, n)$-boundedness of the loops in $\overline{QBF}$ and $\overline{QW}$ using both analytical and numerical methods.
- We also demonstrate the use of our semantics to show the efficiency of different error correction schemes. Consider the error correction of a single qubit in the environment where a single bit flip error happens with probability $0 < p < 1/2$. We consider three schemes: (1) $P_1$ without any error correction; (2) $P_2$ with error correction for bit flips; (3) $P_3$ with error correction for phase flips. We prove that their corresponding robustnesses $\epsilon_1, \epsilon_2, \epsilon_3$ (under trivial precondition $Q = I, \lambda = 0$) satisfy $\epsilon_2 < \epsilon_1 < \epsilon_3$. In other words, we conclude that an error correction scheme that is appropriate for the error model can reduce noise in a program, while an inappropriate error correction scheme may do the opposite.
- Combining these ideas, we further analyze the robustness of a fault-tolerant version of QBF and demonstrate that the use of appropriate fault-tolerant gadgets makes QBF more robust.

**Organization.** We introduce preliminaries about quantum information and the quantum while-language in Section 3 and Section 4 respectively. We present the noisy quantum while-language (syntax, semantics) in Section 5 and define quantum robustness and a logic for bounding it in Section 6. We conclude the paper with case studies in Section 7. We discuss related work, next.

2 RELATED WORK

2.1 Reasoning about Errors in Classical Programs

There has been significant work on reasoning about errors in classical software, which we describe here. We believe that reasoning about errors is even more important in a quantum setting, where we can expect that errors will be prevalent and that the nature of errors will change rapidly as hardware progresses.

2.1.1 Faulty Hardware. One example of an error that a classical computer may experience is a transient hardware fault. Previous work has demonstrated that programming language techniques can help give safety guarantees even in the presence of such errors. For example, Walker et al. [2006] present a type-theoretic framework for analyzing fault-tolerant lambda calculus in the presence of transient faults. Their type system guarantees that well-typed programs can tolerate...
any single data fault. Perry et al. [2007] extend those ideas to typed assembly language. Similar to this work, we present a semantics that tracks errors during computation.

2.1.2 Program Continuity. Our work is also similar to existing work on verifying program continuity and robustness [Chaudhuri et al. 2010, 2011]. In order to prove that a program is continuous or robust, it is necessary to show that the output of that program will not change significantly given small changes to the program’s inputs. We are interested in bounding the distance between the output of a noisy program and its corresponding ideal program given the same input.

2.1.3 Approximate and Probabilistic Computing. In recent years, there have been many advances in programming language support for approximate [Baek and Chilimbi 2010; Boston et al. 2015; Carbin et al. 2012; Park et al. 2015; Sampson et al. 2011] and probabilistic [Bornholt et al. 2014; Sampson et al. 2014] computing. Both of these styles of computing rely on uncertainty during computation, either due to hardware errors or randomness, but still require that programs satisfy some correctness properties. Programming language tools in this area have aimed to provide these correctness guarantees.

Carbin et al. [2013] present a tool for verifying reliability conditions, which indicate the probability that a computation produces the correct result. Given a hardware specification, which lists the probability with which an operation executes correctly, their analysis computes a conservative probability that a program value is computed correctly. We extend this notion to a quantum setting by computing the robustness of a quantum program, which relates to the probability that the state of the system after executing the program is correct. Also, our hardware specification records not only the probability of error, but also the nature of the errors that may occur (operator $\Phi$). Doing so is more computationally expensive, but necessary to be able to reason about the effects of error correction schemes in quantum programs. Carbin et al. [2013] does not consider error correction; instead, errors are considered permanent and only probabilities of errors are used for characterizations.

2.2 Characterizing Error in Quantum Programs

To our knowledge, we are presenting the first semantics and logic for quantum computation that considers errors. Existing work on characterizing error in quantum programs has focused primarily on dynamic approaches such as simulation [Gutiérrez et al. 2013] or physical experimentation [Chuang and Nielsen 1997; Emerson et al. 2005; Knill et al. 2008; Magesan et al. 2011]. Resource estimation tools like QuRE can statically produce error estimates for algorithms given hardware specifications [Suchara et al. 2013]. However, these tools are targeted at quantum circuits (i.e. quantum programs without conditionals or loops) and typically assume that a single error model and single type of error correction will be used throughout the circuit.

3 QUANTUM INFORMATION: PRELIMINARIES AND NOTATIONS

This section presents background and notation on quantum information and quantum computation. For an extended background, we recommend notes by Watrous [2006] and the textbook by Nielsen and Chuang [2000]. A summary of notation we use appears in Table 1.

3.1 Preliminaries

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 $\| |\psi\rangle \| = \sqrt{\langle \psi | \psi \rangle}$.
The state space of a quantum system is a Hilbert space. The state space of a 1 respectively. Another important basis, called the ± 3.2 Quantum States

operations on quantum systems can be characterized by unitary operators. An operator D(H) denotes by D(H).

3.3 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 \( \rho \), the corresponding evolution is \( \rho \mapsto UP\rho U^\dagger \). Common single-qubit unitary operators include

\[
H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.
\]
The Hadamard operator $H$ transforms between the computational and the ± 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 $Z|1\rangle = |--\rangle$.

More generally, the evolution of a quantum system can be characterized by an admissible superoperator $E$, 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 $E$ is $k$-positive if for any $k$-dimensional Hilbert space $\mathcal{A}$, the superoperator $E \otimes I_{\mathcal{A}}$ 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 $E$ is trace-non-increasing if for any initial state $\rho \in \mathcal{D}(\mathcal{H})$, the final state $E(\rho) \in \mathcal{D}(\mathcal{H}')$ after applying $E$ satisfies $\text{tr}(E(\rho)) \leq \text{tr}(\rho)$.

For every superoperator $E : \mathcal{D}(\mathcal{H}) \to \mathcal{D}(\mathcal{H}')$, there exists a set of Kraus operators $\{E_k\}_k$ such that $E(\rho) = \sum_k E_k \rho E_k^\dagger$ 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 $E$ is written as $E = \sum_k E_k \circ E_k^\dagger$. A unitary evolution can be represented by the superoperator $E = U \circ U^\dagger$. An identity operation refers to the superoperator $I_{\mathcal{H}} = I_{\mathcal{H}} \circ I_{\mathcal{H}}$. The Schrödinger-Heisenberg dual of a superoperator $E = \sum_k E_k \circ E_k^\dagger$, denoted by $E^\ast$, is defined as follows: for every state $\rho \in \mathcal{D}(\mathcal{H})$ and any operator $A$, $\text{tr}(AE(\rho)) = \text{tr}(E^\ast(A)\rho)$. The Kraus form of $E^\ast$ is $\sum_k E_k^\dagger \circ E_k$.

### 3.4 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 $\sum M_m^\dagger M_m = I_{\mathcal{H}}$. If we perform a measurement $\{M_m\}$ on a state $\rho$, the outcome $m$ is observed with probability $p_m = \text{tr}(M_m \rho M_m^\dagger)$ 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 $\rho$ becomes $|0\rangle\langle 0|$. With probability $\frac{1}{2}$ the outcome is 1 and $\rho$ becomes $|1\rangle\langle 1|$.

### 4 QUANTUM PROGRAMS

Our work builds on top of the quantum **while**-language developed by Ying [2011, 2016]. Here we review the syntax and semantics of this language.

#### 4.1 Syntax

Define $\text{Var}$ as the set of quantum variables. We use the symbol $q$ as a metavariable ranging over quantum variables and define a quantum register $\overline{q}$ to be a finite set of distinct variables. For each $q \in \text{Var}$, its state space is denoted 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($q$) = $\text{Bool}$ then $\mathcal{H}_q$ is the two-dimensional Hilbert space with basis $\{|0\rangle, |1\rangle\}$. If type($q$) = $\text{Int}$ then $\mathcal{H}_q$ is the Hilbert space with basis $\{|n\rangle : n \in \mathbb{Z}\}$. The syntax of a quantum **while** program $P$ is defined as follows.

$$
P ::= \text{skip} \mid q := |0\rangle \mid \overline{q} := U[\overline{q}] \mid P_1; P_2 \mid \text{case } M[\overline{q}] = m \rightarrow P_m \text{ end} \mid \text{while } M[\overline{q}] = 1 \text{ do } P_1 \text{ done} \quad (4.1)
$$

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** $M[\overline{q}] = m \rightarrow P_m$ **end**
performs the measurement \( M = \{ M_m \} \) on the qubits in \( \Phi \), and executes program \( P_m \) if the outcome of the measurement is \( m \). The bar over \( m \rightarrow P_m \) indicates that there may be one or more repetitions of this expression. \(^2\) 

(6) while \( M(\Phi) = 1 \) do \( P_1 \) done performs the measurement \( M = \{ M_0, M_1 \} \) on the qubits in \( \Phi \), 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 \( |0\rangle \). \(^3\) (2) Evaluating the guard of a case statement or loop, which performs a measurement, potentially disturbs the state of the system.

We now present an example program written in the quantum while-language syntax. The quantum walk \([\text{Aharonov et al. } 2001]\) is a widely considered example in quantum programming, quantum algorithms, and quantum simulation literature \([\text{Georgescu et al. } 2014; \text{Ying et al. } 2017]\). Here we consider a quantum walk on a circle with \( n \) points. We let the initial position of the walker be 0, and say that the program halts if and only if the walker arrives at position 1.

**Example 4.1 (Quantum Walk).** Define the coin (or "direction") space \( \mathcal{H}_c \) to be the 2-dimensional Hilbert space with orthonormal basis states \( |L\rangle \) and \( |R\rangle \), for Left and Right respectively. Define the position space \( \mathcal{H}_p \) to be the \( n \)-dimensional Hilbert space with orthonormal basis states \( |0\rangle, |1\rangle, \ldots, |n-1\rangle \), where vector \(|i\rangle\) represents position \( i \) for \( 0 \leq i < n \). Now the state space of the walk is \( \mathcal{H} = \mathcal{H}_c \otimes \mathcal{H}_p \) and the initial state is \( |L\rangle|0\rangle \). In each step of the walk:

1. Measure the position of the system to determine whether the walker has reached position 1. If the walker has reached position 1, the walk terminates. Otherwise, it continues. We use the measurement \( M = \{|1\rangle\langle 1|, \sum_{i \neq 1} |i\rangle\langle i|\} \).
2. Apply the “coin-tossing” operator \( H \) to the coin space \( \mathcal{H}_c \).
3. Perform the shift operator \( S \) defined by \( S|L, i\rangle = |L, i-1 (\text{mod} \ n)\rangle \), \( S|R, i\rangle = |R, i+1 (\text{mod} \ n)\rangle \) for \( i = 0, 1, \ldots, n-1 \) to the space \( \mathcal{H} \). The \( S \) operator can be written as
   \[
   S = \sum_{i=0}^{n-1} |L\rangle\langle L| \otimes |i-1(\text{mod } n)\rangle\langle i| + \sum_{i=0}^{n-1} |R\rangle\langle R| \otimes |i+1(\text{mod } n)\rangle\langle i|.
   \]

In this algorithm, the walker takes one step left or one step right corresponding to the coin flip result \( |L\rangle \) or \( |R\rangle \). However, unlike the classical case, the result of the coin flip may be a superposition of \( |L\rangle \) and \( |R\rangle \), allowing the walker to take a step to the left and right simultaneously. This quantum walk can be described by the following program

\[
\text{QW}_n \equiv p := |0\rangle; c := |L\rangle; \text{while } M[p] = 1 \text{ do } c := H[c]; c, p := S[c, p] \text{ done.} \tag{4.2}
\]

### 4.2 Operational Semantics

The operational semantics of the quantum while-language are presented in Figure 1a. \( \langle P, \rho \rangle \rightarrow \langle P’, \rho’ \rangle \), 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 \( \rho \) and \( \rho’ \) are partial density operators representing the current state. Intuitively, in one step, we can evaluate program \( P \) on input state \( \rho \) to program

\(^2\)Our syntax for conditional/case statements differs from that presented by Ying [2016] to make it more clear that there are multiple programs \( P_m \).

\(^3\)In our examples, we may write \( q := |\psi\rangle \) for some fixed basis state \( |\psi\rangle \). What we mean in this case is \( q := |0\rangle; q := U[q] \) where \( U \) is the unitary operation that transforms \( |0\rangle \) into \( |\psi\rangle \).
with variable \( q \) written as \( P \). An identity operation is applied to the rest of quantum variables. For example, applying \( |\psi\rangle \) to the quantum system (e.g., a subset of quantum variables of the program), one should assume that the probabilities associated with each transition are encoded in the output partial density operator. For instance, \( \rho \) is a superoperator that handles quantum variables, while \( \rho_0 \) is a superoperator that handles classical variables. The superoperators \( \epsilon_{q \to 0}^{\text{while}} \) and \( \epsilon_{q \to 0}^{\text{int}} \) are used to model quantum and classical transitions, respectively.

\[ \langle q := |0\rangle, \rho \rangle \rightarrow \langle E, \rho \rangle \]

\[ \langle q := |0\rangle, \rho \rangle \rightarrow \langle E, \rho_0' \rangle \]

where \( \rho_0 = \begin{cases} \epsilon_{q \to 0}^{\text{bool}} & \text{if type}(q) = \text{Bool} \\ \epsilon_{q \to 0}^{\text{int}} & \text{if type}(q) = \text{Int} \end{cases} \)

\[ \langle q := U[q], \rho \rangle \rightarrow \langle E, U\rho U^\dagger \rangle \]

\[ \langle E; P_2, \rho \rangle \rightarrow \langle P_2, \rho \rangle \]

\[ \langle P_1, \rho \rangle \rightarrow \langle P_1', \rho' \rangle \]

\[ \langle P_1; P_2, \rho \rangle \rightarrow \langle P_1'; P_2, \rho' \rangle \]

\[ \langle \text{case } M[q] = m \rightarrow P_m \text{ end}, \rho \rangle \rightarrow \langle P_m, M_m\rho M_m' \rangle \]

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

\[ \langle \text{while } M[q] = 1 \text{ do } P_1 \text{ done}, \rho \rangle \rightarrow \langle E, M_0\rho M_0' \rangle \]

\[ \langle \text{while } M[q] = 1 \text{ do } P_1 \text{ done}, \rho \rangle \rightarrow \langle P_1; \text{while } M[q] = 1 \text{ do } P_1 \text{ done}, M_1\rho M_1' \rangle \]

\[ \|\text{skip}\|\rho = \rho \]

\[ \|q := |0\rangle\|\rho = \begin{cases} \epsilon_{q \to 0}^{\text{bool}} & \text{if type}(q) = \text{Bool} \\ \epsilon_{q \to 0}^{\text{int}} & \text{if type}(q) = \text{Int} \end{cases} \]

\[ \|q := U[q]\|\rho = U\rho U^\dagger \]

\[ \|P_1; P_2\|\rho = \|P_2\|\|P_1\|\rho \]

\[ \|\text{case } M[q] = m \rightarrow P_m \text{ end}\|\rho = \sum_m \|P_m\|(M_m\rho M_m') \]

\[ \|\text{while } M[q] = 1 \text{ do } P_1 \text{ done}\|\rho = \bigcup_{k=0}^{\infty} \|\text{while}^k\|\rho \]

\section{Quantum While Programs}

\[ P' \text{ (or } E) \text{ and output state } \rho'. \]

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.\(^4\)

In the \textit{Initialization} rule, the superoperators \( \epsilon_{q \to 0}^{\text{bool}}(\rho) \) and \( \epsilon_{q \to 0}^{\text{int}}(\rho) \), which initialize the variable \( q \) in \( \rho \) to \( |0\rangle\langle 0| \), are defined by \( \epsilon_{q \to 0}^{\text{bool}}(\rho) = |0\rangle\langle 0|\rho|0\rangle\langle 0| + |0\rangle\langle 1|\rho|0\rangle\langle 0| + |0\rangle\langle 0|\rho|0\rangle\langle 0| \) and \( \epsilon_{q \to 0}^{\text{int}}(\rho) = \sum_{n=0}^{\infty} |n\rangle\langle n|\rho\langle n| + |0\rangle\langle 0|\rho|0\rangle\langle 0| \). Here, \( |\psi\rangle\langle \phi| \) 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 \( |\psi\rangle\langle \phi| \) 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. For example, applying \( |\psi\rangle\langle \phi| \) to \( \rho \) means applying \( |\psi\rangle\langle \phi| \otimes I_{\mathcal{H}_q} \) to \( \rho \), where \( \bar{q} \) denotes the set of all variables except \( q \). The identity operation is usually omitted for simplicity.

\(4\)If we had instead considered a probabilistic transition system, then the transition rule for case statements could have been written as \( \langle \text{case } M[q] = m \rightarrow P_m \text{ end}, \rho \rangle \xrightarrow{P_m} \langle P_m, \rho_m \rangle \) where \( P_m = \text{tr}(M_m\rho M_m') \) and \( \rho_m = M_m\rho M_m'/p_m \).

\(318\) S. Hung, K. Hietala, S. Zhu, M. Ying, M. Hicks, and X. Wu

We do not explain the rules in detail, but hope their meaning is self-evident given the description of the language in Section 4.1.

To illustrate the use of the operational semantics, we consider the classic beam splitter experiment, which demonstrates the difference between quantum and classical mechanics. In this experiment, as shown in Figure 2, a photon source sends photons through two beam splitters. The final locations of the photons are determined using photon detectors. A classical analysis would assume that, for each photon, each beam splitter flips a fair coin to either reflect the photon or allow it to pass through. The full mirrors reflect incoming photons with probability one. As a result, we might expect half of the photons to reach one detector and half to reach the other. However, this is not what is observed in experiments. On the contrary, all photons will reach one detector.

Example 4.2 (Beam Splitter Experiment). Model each beam splitter as a Hadamard gate and say that the photon source produces photons on the ‘0’ path, which corresponds to the $|0\rangle$ state. Then the beam splitter experiment (BSE) corresponds to the program

$$BSE \equiv q_1 := |0\rangle; q_1 := H[q_1]; q_1 := H[q_1]$$

(4.3)

where $\text{type}(q_1) = \text{Bool}$. Let $\rho = |1\rangle_{q_1} \langle 1|$. Then the evaluation of program $BSE$ on input $\rho$ proceeds as follows.

$$\langle BSE, \rho \rangle = \langle q_1 := |0\rangle; q_1 := H[q_1]; q_1 := H[q_1], |1\rangle_{q_1} \langle 1| \rangle$$

$$\rightarrow \langle q_1 := H[q_1]; q_1 := H[q_1], |0\rangle_{q_1} \langle 0| \rangle$$

$$\rightarrow \langle q_1 := H[q_1], |+\rangle_{q_1} \langle +| \rangle$$

$$\rightarrow \langle E, |0\rangle_{q_1} \langle 0| \rangle$$

At the first beam splitter, the photons may either continue on their current path (corresponding to $|0\rangle$) or be reflected to the ‘1’ path (corresponding to $|1\rangle$). In quantum mechanics, both possibilities happen simultaneously, resulting in each photon continuing on a superposition of both paths. The superposition of both paths corresponds to the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. At the second beam splitter, the paths in superposition interfere with each other, resulting in the ‘1’ path being cancelled.
4.3 Denotational Semantics

The denotational semantics of a quantum while program is given in Figure 1b. It defines $\|P\|$ as a superoperator that acts on $\rho \in \mathcal{H}_{\text{Var}}$ [Ying 2016]. The semantics of each term is given compositionally. We write $\text{while}^{(k)}$ for the $k$th syntactic approximation (i.e., unrolling) of while and $\sqsubset$ for the least upper bound operator in the complete partial order generated by Löwner comparison. For more detail on the semantics of loops, we refer the reader to Ying [2011, 2016].

We connect the denotational semantics to the operational semantics through the following proposition.

**Proposition 4.1** ([Ying 2016]). For any program $P$

$$\|P\|\rho \equiv \sum \{\rho' : \langle P, \rho \rangle \rightarrow^* \langle E, \rho' \rangle\}, \quad \quad (4.4)$$

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

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

The semantics presented so far assume that no noise will occur during computation. In Section 5, we extend the semantics to include possible errors that may occur during unitary application.

4.4 Quantum Predicates and Hoare Logic

A quantum predicate is a Hermitian operator $M$ such that $0 \subseteq M \subseteq I$ [D’Hondt and Panangaden 2006]. For a predicate $M$ and a state $\rho$, $\text{tr}(M\rho)$ is the expectation of the truth value of predicate $M$ in state $\rho$. Restricting $M$ to be between 0 and $I$ ensures that $0 \leq \text{tr}(M\rho) \leq 1$ for any $\rho \in \mathcal{D}(\mathcal{H})$.

The identity matrix corresponds to the true predicate because for any density operator $\rho$, $\text{tr}(I\rho) = 1$. The zero matrix corresponds to the false predicate because for any density operator $\rho$, $\text{tr}(0\rho) = 0$. $|0\rangle\langle 0|\}$ is the predicate that says that a state is in the subspace spanned by $|0\rangle$. As an example, the density operator $\rho_0$ corresponding to the state $|0\rangle$ is such that $\text{tr}(|0\rangle\langle 0|\rho_0) = 1$, and the density operator $\rho_1$ corresponding to the state $\sqrt{1/3}|0\rangle + \sqrt{2/3}|1\rangle$ is such that $\text{tr}(|0\rangle\langle 0|\rho_1) = \frac{1}{3}$.

Ying [2011, 2016] uses quantum predicates as the basis for defining quantum preconditions and postconditions in his quantum Hoare logic. Let $M$ and $N$ be quantum predicates and let $P$ be a quantum while program. Then $M$ is a precondition of $N$ with respect to $P$, written $\{M\}P\{N\}$, if

$$\forall \rho. \text{tr}(M\rho) \leq \text{tr}(N\|P\|\rho). \quad \quad (4.5)$$

This inequality can be seen as the probabilistic version of the following statement: if state $\rho$ satisfies predicate $M$, then after applying the program $P$ the resulting state will satisfy predicate $N$. If we include an auxiliary space $\mathcal{A}$, then the equivalent statement is

$$\forall \rho. \text{tr}((M \otimes I_{\mathcal{A}})\rho) \leq \text{tr}((N \otimes I_{\mathcal{A}})(\|P\| \otimes I_{\mathcal{A}})(\rho)). \quad \quad (4.6)$$

5 NOISY QUANTUM PROGRAMS

In this section, we present the syntax and semantics for the quantum while-language with noise, as an extension of the quantum while-language. Our syntax allows one to explicitly encode any error model that describes local noise during the execution of a quantum program.

5.1 Noise in Quantum Computation

Here we briefly discuss how noise is modeled in the study of quantum error correction and fault-tolerant quantum computation [Gottesman 2010], which in turn comes from the noise model in quantum physical experiments. It is a convention to only consider “local” noise rather than correlated noise, because benign white noise is more likely than “adversarial” noise in actual
quantum devices. A few types of natural local noise arise in realistic quantum systems, which generalize classical bit-flip errors, including:

- The bit flip noise flips the state with probability \( p \), and can be represented by
  \[
  \Phi_{p, \text{bit}} = (1 - p) I \circ I + p X \circ X. \tag{5.1}
  \]

- The phase flip noise flips the phase with probability \( p \), and can be represented by
  \[
  \Phi_{p, \text{phase}} = (1 - p) I \circ I + p Z \circ Z. \tag{5.2}
  \]

Other types of noise include depolarization, amplitude damping, and phase damping [Nielsen and Chuang 2000].

This model of noise is used by experimental physicists for building and benchmarking quantum devices in both academia and industry [Terhal 2015]. Noisy information of specific quantum devices can also be publicly available (e.g., the IBM Q-experience 5).

5.2 Syntax

The syntax of a noisy quantum program \( \overline{P} \) is defined as follows

\[
\overline{P} ::= \text{skip} \mid q := \ket{0} \mid \overline{q} := \Phi \ U[\overline{q}] \mid \overline{P}_1 ; \overline{P}_2 \mid \text{case } M[\overline{q}] = m \rightarrow \overline{P}_m \text{ end} \mid \text{while } M[\overline{q}] = 1 \text{ do } \overline{P}_1 \text{ done} \tag{5.3}
\]

This syntax is identical to that of the standard quantum while language described in Section 4, except that we have annotated the unitary application construct with an error probability \( p \) and an error model \( \Phi \), which is the superoperator of the noisy operation. The statement \( \overline{q} := \Phi \ U[\overline{q}] \) will apply the correct operation \( U \) on \( q \) with probability \( 1 - p \) and will apply the noisy (or erroneous) operation \( \Phi \) on \( q \) with probability \( p \). The nature of \( \Phi \) will depend on the underlying hardware, and is a parameter to our language.

For any noisy program \( \overline{P} \), its corresponding ideal program can be obtained by simply replacing any noisy unitary operations by their ideal versions (i.e., we ignore \( p \) and \( \Phi \)). We write \( \text{ideal}(\overline{P}) \) for this program, or simply \( P \) when there is no ambiguity.

We remark that noisy unitary operations are already expressive enough to capture many types of noise. First, any noise can depend on the quantum state of the system by the nature of modeling it as a quantum operation \( \Phi \). Second, noisy initialization can be modeled as initialization followed by application of a noisy identity operation, and noisy measurement can be modeled as application of a noisy identity operation followed by measurement. Third, errors that occur between applications of subsequent unitaries can also be modeled by noisy identity operations.

5.3 Semantics

The operational and denotational semantics of noisy quantum while programs are also identical to those of the standard quantum while programs, except that the rules related to unitary application now include an error term, as shown in Figure 3a and Figure 3b. Note that we do not require \( p \) and \( \Phi \) to be the same in every instance of a unitary application. They may depend on the type of unitary being applied, or on other features of the program such as the number of operations performed so far. We use explicit characterizations of the noise given by \( \Phi \) to enable us to argue about the effect of error-correcting gadgets explicitly written in the program. If we only considered error probability (like Carbin et al. [2013]) then we could only reason about error as accumulating throughout the program, and we would not be able to show that error-correcting gadgets reduce noise by cancelling previous errors.

worth noting that for any operators $A$ and $B$, the variation distance is given by
\begin{align*}
\|A - B\|_1 = \sum_x |A(x) - B(x)|.
\end{align*}

In the context of computation with noise, we wish to measure the distance between the ideal state and the state influenced by noise. In classical probabilistic computation, the output can be described as a probability distribution over all possible outputs. A common measure is the total variation distance of two distributions $p, q$, defined by
\begin{align*}
|p - q|_1 = 1/2 \sum_x |p(x) - q(x)|.
\end{align*}

In quantum computation, we define the trace distance as the quantum generalization of the total variation distance. For Hermitian operator $A$, let the trace norm $\|A\|_1$ be the summation of the absolute value of all its eigenvalues. When $A$ is positive semidefinite, one has $\|A\|_1 = \text{tr}(A)$. It is worth noting that for any operators $A$ and $B$ the following triangle inequality holds:
\begin{align*}
\|A + B\|_1 \leq \|A\|_1 + \|B\|_1. \quad (6.1)
\end{align*}
For two states \( \rho_1, \rho_2 \in \mathcal{D}(\mathcal{H}) \), the trace distance between \( \rho_1 \) and \( \rho_2 \), denoted \( T(\rho_1, \rho_2) \), is defined to be \( \frac{1}{2} ||\rho_1 - \rho_2||_1 \). It can be shown that the trace distance satisfies
\[
T(\rho_1, \rho_2) \equiv \frac{1}{2} ||\rho_1 - \rho_2||_1 = \max_{P \in \mathcal{P}} \text{tr}(P(\rho_1 - \rho_2)),
\]
where the maximization is over all possible measurements \( P \) (i.e., \( 0 \subseteq P \subseteq \mathcal{I} \)). By definition, \( 0 \leq T(\rho, \sigma) \leq 1 \) for any pair of states \( \rho, \sigma \). One can also interpret the trace distance as the advantage with which one can distinguish two states \( \rho \) and \( \sigma \). For example, the states \( |0\rangle\langle 0| \) and \( |1\rangle\langle 1| \) are perfectly distinguishable by measurement \( \{ |0\rangle\langle 0|, |1\rangle\langle 1| \} \), so \( T(|0\rangle\langle 0|, |1\rangle\langle 1|) = 1 \).

One can further define the distance between two different superoperators. For any two superoperators \( E \) and \( E' \) over \( \mathcal{H} \), an intuitive way to define their distance is to find the input state to both superoperators that maximizes the trace distance of their output states. However, it turns out that this definition alone is insufficient to capture the distance between superoperators because of a unique quantum feature called entanglement. The distinguishability between \( E \) and \( E' \) can be significantly enlarged \(^6\) when one introduces an auxiliary Hilbert space \( \mathcal{A} \) and tries to distinguish \( E \otimes I_\mathcal{A} \) and \( E' \otimes I_\mathcal{A} \) with some entangled input state over \( \mathcal{H} \otimes \mathcal{A} \) [Gilchrist et al. 2005].

To account for this, we define the diamond norm between two superoperators \( E \) and \( E' \) as
\[
||E - E'||_\diamond \equiv \max_{\rho \in \mathcal{D}(\mathcal{H} \otimes \mathcal{A}) : \text{tr}(\rho) = 1} T(E \otimes I_\mathcal{A}(\rho), E' \otimes I_\mathcal{A}(\rho))
\]
for any auxiliary space \( \mathcal{A} \). Without loss of generality, one can assume \( \mathcal{A} \) is a copy of \( \mathcal{H} \), i.e., \( \mathcal{A} = \mathcal{H} \) [Watrous 2018]. Given the representations of \( E, E' \), one can efficiently calculate the diamond norm \( ||E - E'||_\diamond \) by a semidefinite program (SDP) [Watrous 2009].

Imagine superoperators that represent different components of a (noisy or ideal) quantum program, which may act on different parts of the quantum system. The diamond norm will allow us to address potential entanglement between different parts of the state and ensure that we can compose the distances computed from different components of the program.

### 6.2 Definition of Quantum Robustness

To capture how noise (error) impacts the execution of quantum program \( \widetilde{P} \), we want to compare \( \|\widetilde{P}\| \) and \( \|P\| \), which are superoperators representing the execution of quantum program \( P \) with and without noise respectively. A natural candidate is to use the aforementioned diamond norm to measure the distance between \( \|\widetilde{P}\| \) and \( \|P\| \). To account for prior knowledge of the input state, we extend the definition of the diamond norm to consider only input states that satisfy predicate \( Q \) to degree at least \( \lambda \). More explicitly, we have

**Definition 6.1 ((Q, \lambda)-Diamond Norm).** Given superoperators \( E, E' \), quantum predicate \( Q \) over \( \mathcal{H} \), and \( 0 \leq \lambda \leq 1 \), the \((Q, \lambda)\)-diamond norm between \( E \) and \( E' \), denoted \( ||E - E'||_{Q, \lambda} \), is defined by
\[
||E - E'||_{Q, \lambda} \equiv \max_{\rho \in \mathcal{D}(\mathcal{H} \otimes \mathcal{A}) : \text{tr}(\rho) = 1, \text{tr}(Q\rho) \geq \lambda} T(E \otimes I_\mathcal{A}(\rho), E' \otimes I_\mathcal{A}(\rho)),
\]
where \( \mathcal{A} \) is any auxiliary space.

We remark that \( \mathcal{A} \) can be assumed to be \( \mathcal{H} \) without loss of generality due to a similar reason for the original diamond norm (see, for example, Watrous [2018, Chap 3]).

---

\(^6\)For example, consider a 2-dimensional Hilbert space \( \mathcal{H} = \mathbb{C}^2 \) and two superoperators \( E, E' \) over \( \mathcal{H} \) with \( E(\rho) = \frac{1}{2}\text{tr}(\rho)\mathbb{1} + \frac{1}{2}\rho^T \) and \( E'(\rho) = \text{tr}(\rho)\mathbb{1} + \rho^T \), where \( \rho^T \) is the transpose of \( \rho \). Without introducing an auxiliary space, direct calculation gives \( T(E(\rho), E'(\rho)) = \frac{1}{2} ||I - 2|\rho| || \leq \frac{1}{2} \) for any single qubit state \( \rho \). However, if we use an auxiliary qubit and apply \( E \) to the maximally entangled state \( |\phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \), we have \( T(E(|\phi^+\rangle \langle \phi^+|), E'(|\phi^+\rangle \langle \phi^+|)) = 1 \).
We argue that \((Q, \lambda)\)-diamond norm is a seminorm in Appendix A.1. Intuitively, this is conceivable since we only restrict the input state from all density operators to a convex subset satisfying \(\text{tr}(Q\rho) \geq \lambda\). Note that, by definition, \(\|\rho\|_\theta = \|\rho\|_{I, \lambda}\) for any \(0 \leq \lambda \leq 1\).

Let \(\mathcal{E}\) and \(\mathcal{E}'\) be superoperators over \(\mathcal{H}\). By extending Watrous [2009], we show that \(\|\mathcal{E} - \mathcal{E}'\|_{Q, \lambda}\) can be efficiently computed by the following semidefinite program (SDP):

\[
\begin{align*}
\max \ & \text{tr}(J(\Phi)W) \\
\text{s.t.} \ & W \leq I_\mathcal{H} \otimes \rho, \ \text{tr}(Q\rho) \geq \lambda, \\
\ & \rho \in \mathcal{D}(\mathcal{H}), \ W \text{ is a positive semidefinite operator over } \mathcal{H} \otimes \mathcal{H},
\end{align*}
\]

(6.5)

where \(\Phi = \mathcal{E} - \mathcal{E}'\) and \(J(\Phi)\) is the Choi-Jamiołkowski representation of \(\Phi\). Note that the above SDP is identical to the SDP used in Watrous [2009, Section 4] to compute \(\|\mathcal{E} - \mathcal{E}'\|_\theta\), except that we have added the additional constraint \(\text{tr}(Q\rho) \geq \lambda\) to capture the requirement on input states. The correctness of the above SDP then basically follows from the analysis of Watrous [2009] and the definition of \((Q, \lambda)\)-diamond norm.

The standard diamond norm and \((Q, \lambda)\)-diamond norm can be significantly different for the same pair of superoperators \(\mathcal{E}, \mathcal{E}'\). For example, consider \(\mathcal{E} = H \circ H\) and \(\mathcal{E}' = HZ \circ ZH\). We can show \(^7\) that \(\|\mathcal{E} - \mathcal{E}'\|_\theta = 1\), whereas \(\|\mathcal{E} - \mathcal{E}'\|_{0, 3/4} = \sqrt{3}/2\). Thus, the \((Q, \lambda)\)-diamond norm can help us leverage prior knowledge about input states to obtain more accurate bounds.

Using the \((Q, \lambda)\)-diamond norm, we define a notion of quantum robustness as follows.

**Definition 6.2 (Quantum Robustness).** The noisy program \(\bar{P}\) (over \(\mathcal{H}\), having ideal program \(P = \text{ideal}(\bar{P})\)) is \(\epsilon\)-robust under \((Q, \lambda)\) if and only if

\[
\|\|\bar{P}\| - \|P\|\|_{Q, \lambda} \leq \epsilon.
\]

(6.8)

Here, \(Q\) is a quantum predicate over \(\mathcal{H}\) and \(0 \leq \lambda, \epsilon \leq 1\).

By the definition of the \((Q, \lambda)\)-diamond norm and the trace distance, (6.8) can be equivalently stated as the following.

\[
\forall \rho \in \mathcal{D}(\mathcal{H} \otimes \mathcal{H}), \ \text{tr}(Q\rho) \geq \lambda \text{tr}((\rho) \Rightarrow \frac{1}{2} \|\|\bar{P}\| \otimes I_{\mathcal{H}}(\rho) - \|P\| \otimes I_{\mathcal{H}}(\rho)\|_1 \leq \epsilon \text{tr}(\rho).
\]

(6.9)

Since \(\epsilon\) measures the distance between \(\|\bar{P}\|\) and \(\|P\|\), the smaller \(\epsilon\) is, the closer the noisy program \(\bar{P}\) is to the ideal program \(P\). One could think of \(\epsilon\) as measuring both the probability that noise can happen and intensity of that noise. When the noise is strong, a noisy program \(\bar{P}\) being \(\epsilon\)-robust implies that the probability of noise is at most \(\epsilon\). When the noise is weak, it could occur with greater probability, but its effect will be much smaller.

Intuitively, the use of precondition \(Q\) can help us obtain more accurate bounds. For example, one can use \(Q\) to characterize prior information about the input state due to the nature of underlying physical systems. Even without any prior knowledge about the input state, preconditions can still be leveraged for different branches of the program in case statements and loops.

---

\(^7\)For the normal diamond norm, consider the input state to be \(\rho = |+\rangle\langle +| \otimes \sigma\) for some ancilla state \(\sigma\). It is easy to see that \(\mathcal{E}(\rho) = |0\rangle\langle 0|\) and \(\mathcal{E}'(\rho) = |1\rangle\langle 1|\), which are perfectly distinguishable. For the \((|0\rangle\langle 0|, 1/4)\)-diamond norm, without loss of generality, consider any input state \(|\psi\rangle = \cos \theta |0\rangle |\psi_0\rangle + \sin \theta |1\rangle |\psi_1\rangle\) where \(\theta \in [0, \pi/4]\). Requiring \(|\psi\rangle\) to satisfy \(|0\rangle\langle 0|\) to degree at least \(1/4\), we have \(\cos^2 \theta \geq 1/4\) and therefore \(\theta \in [0, \pi/4]\). Simple calculation gives \(H|\psi\rangle = \cos \theta |+\rangle |\psi_0\rangle + \sin \theta |\rangle |\psi_1\rangle\) and \(HZ|\psi\rangle = \cos \theta |+\rangle |\psi_0\rangle - \sin \theta |\rangle |\psi_1\rangle\). The projector that maximally distinguishes these states is \(|0\rangle\langle 0| \otimes I\), so we have

\[
\|\mathcal{E} - \mathcal{E}'\|_{0, 3/4} = \frac{1}{2} \left( (\cos \theta + \sin \theta)^2 - (\cos \theta - \sin \theta)^2 \right) = \sin 2\theta \leq \frac{\sqrt{3}}{2},
\]

the equality of which holds when \(\theta = \pi/6\).
A program's robustness can be proved by working out the (denotational) semantics of programs \( P \) triples as in Section 4.4. Sequence their computed upper bounds. Note that in the \( \lambda \) and \( Q \) and ideal programs. The Rescale state must satisfy the predicate, or increase the upper bound on the distance between the noisy noise operation \( \Phi \) says that if we can bound the \( Q \)-error-free. These operations will not increase the distance between \( Q \). In Section 6.3.1 Simple Rules. The \( \text{skip} \) rule says that equivalent forms of our judgment can be obtained \( P \equiv (q := |0\rangle) \leq 0 \) (Init).

\[
(Q, \lambda) \vdash \text{skip} \leq 0 \quad \text{(Skip)} \\
(Q, \lambda) \vdash (q := |0\rangle) \leq 0 \quad \text{(Init)}
\]

\[
\| U \circ U^\dagger - \Phi \|_{Q, \lambda} \leq \epsilon \\
(Q, \lambda) \vdash (\bar{q} :=_{P, \Phi} U[\bar{q}]) \leq pe \quad \text{(Unitary)}
\]

\[
(Q', \lambda') \vdash \bar{P} \leq e' \quad e' \leq \epsilon \quad Q \subseteq Q' \quad \lambda' \leq \lambda
\]

\[
(Q, \lambda) \vdash \bar{P} \leq \epsilon 
\]

\[
(Q/\delta, \lambda/\delta) \vdash \bar{P} \leq \epsilon \\
0 \subseteq Q, Q/\delta \subseteq I \\
0 \leq \lambda, \lambda/\delta \leq 1 
\]

\[
(Q_1, \lambda) \vdash \bar{P}_1 \leq e_1 \\
(Q_2, \lambda) \vdash \bar{P}_2 \leq e_2 \\
\{Q_1 \} P_1 \{Q_2 \} 
\]

\[
\forall m, (Q_m, 1 - \delta) \vdash \bar{P}_m \leq \epsilon \\
t, \delta \in [0, 1] 
\]

\[
(\sum_m M_m^t Q_m M_m, 1 - t \delta) \vdash \text{(case } M[\bar{q}] = m \rightarrow \bar{P}_m \text{ end)} \leq (1 - t)\epsilon + t 
\]

\[
(Q, \lambda) \vdash \bar{P}_1 \leq \epsilon \\
\{Q\} P_1 \{\lambda M_0^t M_0 + M_1^t Q M_1\} 
\]

\[
\bar{P} \equiv \text{while } M[\bar{q}] = 1 \text{ do } \bar{P}_1 \text{ done } \quad P \text{ is } (a, n)-\text{bounded} 
\]

\[
(\lambda M_0^t M_0 + M_1^t Q M_1, \lambda) \vdash \bar{P} \leq ne/(1 - a) \quad \text{(While-Unbounded)}
\]

\[
(Q, \lambda) \vdash (\text{while } M[\bar{q}] = 1 \text{ do } \bar{P}_1 \text{ done}) \leq 1 
\]

Fig. 4. Rules for logic of quantum robustness.

6.3 Logic for Quantum Robustness

A program’s robustness can be proved by working out the (denotational) semantics of programs \( P \) and \( \bar{P} \) and applying Definition 6.2 directly. However, this computation may be difficult. As an alternative, we present a logic for proving judgments of the form \( (Q, \lambda) \vdash \bar{P} \leq \epsilon \), meaning that program \( \bar{P} \) is \( \epsilon \)-robust under \( (Q, \lambda) \). In Section 6.4 we prove the logic is sound. In Section 7 we demonstrate the use of the logic, alongside direct proofs of robustness for some cases (e.g., error correction).

The rules for our logic are given in Figure 4.

6.3.1 Simple Rules. The \( \text{skip} \) and \( \text{Init} \) rules say that the skip and initialization operations are always error-free. These operations will not increase the distance between \( \|P\| \) and \( \|\bar{P}\| \). The \( \text{Unitary} \) rule says that if we can bound the \( (Q, \lambda) \)-diamond norm between the intended operation \( U \) and the noise operation \( \Phi \) by \( \epsilon \), then we can bound the total distance by \( pe \). The \( \text{Weaken} \) rule says that we can always safely make the precondition more restrictive, increase the degree to which an input state must satisfy the predicate, or increase the upper bound on the distance between the noisy and ideal programs. The \( \text{Rescale} \) rule says that equivalent forms of our judgment can be obtained by rescaling \( Q \) and \( \lambda \). Note that the \( \text{Rescale} \) rule does not weaken the judgment, but rather provides some flexibility in choosing \( Q, \lambda \) compatible with other rules; one can scale by \( \delta \) so long as \( Q/\delta \) and \( \lambda/\delta \) are still well defined. The \( \text{Sequence} \) rule allows us to compose two judgments by summing their computed upper bounds. Note that in the \( \text{Sequence} \) and \( \text{While-Bounded} \) rules we define Hoare triples as in Section 4.4.
6.3.2 The Case Rule. The Case rule says that, given appropriate bounds for every branch of a case statement, we can bound the error of the entire case statement. Note that $\sum_m M_m^t Q_m M_m$ is the weakest precondition of the case construct in quantum Hoare logic [Ying 2016]. In a logic for classical programs, one might expect each branch of a case statement to satisfy the precondition perfectly. However, in a quantum logic, as we will discuss in the soundness proof (see Section 6.4), this is not necessarily true. To see this, note that in our rule we start with the precondition $\sum_m M_m^t Q_m M_m$ and $\lambda = 1 - t\delta$ on the input state to the case statement, but we can only guarantee that a weighted fraction of $1 - t$ of the branches satisfy a weaker precondition $Q_m$ and $\lambda' = 1 - \delta$ for some choice of $t \in [0, 1]$. (Note that $1 - \delta \leq 1 - t\delta$ for $t, \delta \in [0, 1]$.)

When applying this rule, one can make $1 - \delta$ and $\epsilon$ the same for every $P_m$ by applying the Weaken and/or Rescale rules. The choice of $t$ will represent a tradeoff between a lower error bound and a more restrictive requirement on the satisfaction of the predicate (see Example 6.2).

Example 6.1 (Simple case statement). Consider the following program.

$$\bar{P} \equiv \text{case } M[\overline{q}] = 0 \rightarrow \overline{q} : \equiv_{p, \Phi} H[\overline{q}]$$

$$1 \rightarrow \text{skip}$$

end

This program performs measurement in the $\{|0\rangle, |1\rangle\}$ basis and either applies a noisy Hadamard $H$ gate or does nothing depending on the measurement outcome. In order to apply the Case rule, we must bound the error of both branches. By the Skip rule, the second branch will have zero error, i.e., $(Q, \lambda) \vdash \text{skip} \leq 0$ for any $Q$ and $\lambda$. We will choose $Q = I$ and $\lambda = 1$. By the Unitary rule, the error of the first branch will depend on $\Phi$.

Consider an error model given by $\Phi(\rho) = HZ\rho ZH$. This means that with probability $1 - p$, $\overline{q} : \equiv_{p, \Phi} H[\overline{q}]$ applies an $H$ gate, and with probability $p$ it applies a $Z$ gate followed by an $H$ gate. Recall that $Z|0\rangle\langle 0|Z = |0\rangle\langle 0|$. This means that a $Z$ error will not affect the state in the first branch (because the first branch corresponds to having measured a 0). So we have that $(|0\rangle\langle 0|, 1) \vdash (\overline{q} : \equiv_{p, \Phi} H[\overline{q}]) \leq 0$, which says that if the input state is the $|0\rangle$ state, then with the error model defined, there will be no error during an application of $H$ (i.e. $\epsilon = 0$). Note that without the precondition $|0\rangle\langle 0|$, we would have $\epsilon > 0$.

Given $(|0\rangle\langle 0|, 1) \vdash (\overline{q} : \equiv_{p, \Phi} H[\overline{q}]) \leq 0$ and $(I, 1) \vdash \text{skip} \leq 0$, we can conclude that $(I, 1) \vdash \bar{P} \leq t$ by the Case rule with the following simplification,

$$M_0^t Q_0 M_0 + M_1^t Q_1 M_1 = |0\rangle\langle 0||0\rangle\langle 0| + |1\rangle\langle 1||1\rangle\langle 1| = |0\rangle\langle 0| + |1\rangle\langle 1| = I.$$  

Because $\delta = 0$, we can choose $t = 0$ to get an overall error bound of zero. However, in general, the choice of $t$ will represent a tradeoff between a lower error bound and a more restrictive requirement on the satisfaction of the predicate (i.e. a larger $\lambda$ value).

Note that the precondition for each branch does not need to directly relate to the measurement basis. For example, consider instead the following program.

$$\text{case } M[\overline{q}] = + \rightarrow \overline{q} : \equiv_{p, \Phi} H[\overline{q}]$$

$$- \rightarrow \text{skip}$$

end

This program performs measurement in the $\{|+, -\rangle\}$ basis. For this program, we can still use the judgments $(|0\rangle\langle 0|, 1) \vdash (\overline{q} : \equiv_{p, \Phi} H[\overline{q}]) \leq 0$ and $(I, 1) \vdash \text{skip} \leq 0$ described in the previous example,
but now our conclusion will be $(\frac{1}{2}|+\rangle\langle+| + |\rangle\langle-|, 1) + \tilde{P} \leq t$, by the following simplification,

$$M_0^t Q_0 M_0 + M_1^t Q_1 M_1 = |+\rangle\langle+|0\rangle\langle0| + |\rangle\langle-|\rangle\langle-| - \frac{1}{2} |+\rangle\langle+| + |\rangle\langle-|\rangle\langle-|.$$

Note that $\frac{1}{2}|+\rangle\langle+| + |\rangle\langle-| \in I$, so we have restricted the set of states for which this error bound will hold. This may be useful in situations where it is difficult to compute an error bound given certain preconditions. For example, it may be difficult to show that $(|+\rangle\langle+|, 1) + (q: \equiv_{p, \Phi} H[\tilde{q}]) \leq \epsilon$ for a sufficiently low $\epsilon$, but it is easy to show that $(0\rangle\langle0|, 1) + (\tilde{q}: \equiv_{p, \Phi} H[\tilde{q}]) \leq 0$.

**Example 6.2 (t-value tradeoffs).** Consider the following program.

$$\tilde{P} \equiv \begin{array}{ll}
\text{case } M[\tilde{q}] = 0 & \rightarrow \tilde{P}_0 \\
1 & \rightarrow \tilde{P}_1 
\end{array}$$

Say that we have that $(Q_0, 1 - 0.05) + \tilde{P}_0 \leq 0.01$ and $(Q_1, 1 - 0.25) + \tilde{P}_1 \leq 0.1$ for programs $\tilde{P}_0$ and $\tilde{P}_1$. Now we can use the Case rule to conclude that $(M_0^t Q_0 M_0 + M_1^t Q_1 M_1, 1 - t\delta) + \tilde{P} \leq (1 - t)\epsilon + t$ for $\delta = \min(0.05, 0.25) = 0.05$, $\epsilon = \max(0.01, 0.1) = 0.1$, and $t \in [0, 1]$.

If we choose $t = 0$ then we have the lowest possible error bound, but $1 - t\delta = 1$, so the produced error bound will only apply to states that completely satisfy the predicate $M_0^t Q_0 M_0 + M_1^t Q_1 M_1$. If we choose $t = 1$ then we have the least restrictive condition on input states, but $(1 - t)\epsilon + t = 1$, which is the trivial error bound. By choosing $t$ to be between 0 and 1 we can trade between a low error bound and a less restrictive constraint on the input state. For example, for $t = 0.25$ we have that $1 - t\delta = 0.9875$ and $(1 - t)\epsilon + t = 0.325$. For $t = 0.5$ we have that $1 - t\delta = 0.975$ and $(1 - t)\epsilon + t = 0.55$.

6.3.3 The Loop Rule. Finally, the While-Bounded and While-Unbounded rules allow us to compute an upper bound for the distance between the noisy and ideal versions of a while loop. The While-Unbounded rule is a trivial bound on the distance. The While-Bounded rule, however, demonstrates a non-trivial upper bound with an assumption called (a, n)-boundedness. Such an assumption is necessary for us to get around the potential issue of termination and to be able to reason about interesting programs in our case study.

Intuitively, (a, n)-boundedness is a condition on how fast the ideal loop will converge, which is inherent to the control flow of the program and does not depend on any specific error model. We view this as an advantage as we do not need a new analysis for every possible noise model.

**Definition 6.3 ((a, n)-boundedness).** A while loop $P \equiv \text{while } M[q] = 1 \text{ do } P_1 \text{ done}$ is said to be (a, n)-bounded for $0 \leq a < 1$ and integer $n \geq 1$ if

$$(E^*)^n(M_1^a M_1) \subseteq aM_1^a M_1$$

where the linear map $E(\rho)$ is defined as $\|P_1\| (M_1 \rho M_1^*)$ and $E^*$ is the dual map of $E$.

Intuitively speaking, a loop is (a, n)-bounded if, for every state $\rho$, after $n$ iterations it is guaranteed that at least a $(1 - a)$-fraction of the state has exited the loop. A while loop with this nice property is guaranteed to terminate with probability 1 on all input states, which helps avoid the termination issue. As we will show in the examples that follow and in Section 7, specific (a, n) can be derived analytically or numerically for concrete programs. We also remark that by assuming (a, n)-boundedness, we avoid weakening $\lambda$ like in the Case rule.

---

8This might even be impossible when $M_0^t Q_0 M_0 + M_1^t Q_1 M_1 \subseteq I$. In that case, the choice $t = 0$ becomes useless.

9If $\|P_1\|$ can be written as $\sum_k F_k \circ F_k^*$ for some set of Kraus operators $\{F_k\}_k$, the Kraus form of $E^*$ is $\sum_k M_1^a F_k \circ F_k M_1$.

10The rough idea is to guess (or enumerate) $n$ and prove $a$ either analytically or numerically (with a simple SDP).
In Carbin et al. [2013], loops are assumed to have a bounded number of iterations or a trivial upper bound will be used (i.e., our rule While-Unbounded). Because of the use of \((a, n)\)-boundedness, we can handle more complicated loops. One also has the freedom to choose appropriate values of \(Q\) for different purposes. A simple choice is \(Q = \lambda I\). A less trivial choice of \(Q\) is shown in the following example.

**Example 6.3 (Slow state preparation).** In this example, we consider the following program which prepares the standard basis state \(|1\rangle\):

\[
\begin{align*}
SSP & \equiv q := |0\rangle; \text{while } M[q] = 0 \text{ do } q := H[q]; q := I[q] \text{ done},
\end{align*}
\]

where \(M\) is the standard basis measurement \(\{0\langle 0|, 1\langle 1|\}\). We consider the case where there is a bit flip error with probability 0.01 when applying the \(I\) gate, i.e., the ideal and the noisy loop bodies are \(P_1 \equiv q := H[q]; q := I[q]\); and \(\overline{P}_1 \equiv q := H[q]; q := I_{0.01, X} I[q]\). Then \([P_1] = H \circ H\) and \([\overline{P}_1] = 0.99H \circ H + 0.01XH \circ HX\). Consider \(Q = |0\rangle \langle 0|\) and \(\lambda = 1\). Since \([P_1]|0\rangle \langle 0| = [\overline{P}_1]|0\rangle \langle 0| = |\rangle \langle +|\), we have that \([\|P_1\| - \|\overline{P}_1\||]|0\rangle \langle 0|, 1\rangle = 0\), and by the Unitary rule, \(|0\rangle \langle 0|, 1\rangle \vdash \overline{P}_1 \leq 0\).

We can use this choice of \(Q\) and \(\lambda\) when applying the While-Bounded rule. First, note that the statement \(\{Q\} P_1 \{M_0^1 M_0 + M_1 Q M_1^1\}^1\) holds because \(\lambda M_0^1 M_0 + M_1 Q M_1^1 = I\) for our choice of \(M_0, M_1, Q, \lambda\). Next, we need to show \((a, n)\)-boundedness of the loop. This requires us to consider the behavior of \(E^*\) where \(E^*\) is the dual of \(E = HM_1 \circ M_1 H\). We claim that the while loop is \((1/2, 1)^{-}\)-bounded because

\[
E^*(M_1^1 M_1) = |0\rangle \langle 0| H|0\rangle \langle 0| H|0\rangle \langle 0| = \frac{1}{2}|0\rangle \langle 0| = \frac{1}{2} M_1^1 M_1.
\]

Now by the While-Bounded rule, we have that \((I, 1) \vdash SSP \leq 0\), i.e., the program is perfectly robust.

In Example 6.3, if we use the precondition \(I\) in our judgment of the robustness of the loop body, the best upper bound we can argue is \(\epsilon = 0.01\), i.e., \((I, 1) \vdash \overline{P}_1 \leq 0.01\). Then, applying the While-Bounded rule yields \((I, 1) \vdash SSP \leq 0.02\) since \(\frac{nc}{1-\epsilon} = 2\epsilon = 0.02\). Therefore, restricting the state space with the predicate \(Q = |0\rangle \langle 0|\), as we did in Example 6.3, was a better choice, as it yielded a better bound. Moreover, this choice of \(Q\) is natural since the post-measurement state entering the loop satisfies \(Q\) perfectly. We note that the program \(SSP\) might look contrived for preparing \(|1\rangle\) when compared to the more straightforward program \(SP \equiv q := |0\rangle; q := X[q]\). We argue that \(SSP\) might be preferred over \(SP\) in the presence of noise. Consider the same noise model as above. Namely, let \(\widehat{SP} \equiv q := |0\rangle; q := X[q]; q := I_{0.01, X} I[q]\). We have shown that \((I, 1) \vdash SSP \leq 0\) given this noise model, which says that \(SSP\) is perfectly robust. By directly applying Definition 6.2, we can show that \(\widehat{SP}\) is 0.01-robust given the same noise model,\(^{11}\) which suggests that \(\widehat{SP}\) is less desirable in this situation.

### 6.4 Soundness

In this section, we show that logic given in Figure 4 is sound.

**Theorem 6.4 (Soundness).** If \((Q, \lambda) \vdash \overline{P} \leq \epsilon\) then \(\overline{P}\) is \(\epsilon\)-robust under \((Q, \lambda)\).

**Proof.** The proof proceeds by induction on the derivation \((Q, \lambda) \vdash \overline{P} \leq \epsilon\). We will work primarily from definition of robustness given in (6.9). We note that superoperators \([\overline{P}]\) and \([P]\) apply on the same space. By the definition of the \((Q, \lambda)\)-diamond norm, we need to consider \([P] \otimes I\) and \([P] \otimes I\). However, to simplify the presentation, we will omit "\(\otimes I\)" in the following proof whenever there is no ambiguity.

Now we consider each possible rule used in the final step of the derivation.

\(^{11}\) Note that \([\widehat{SP}] = 0.99X \circ X + 0.01I \circ I\), and hence we have \([\|SP\| - \|\widehat{SP}\|]\|_{L, 1} = 0.01\|X \circ X - I \circ I\|_{L, 1} = 0.01\).
(1) **Skip:** This rule holds by observing $\|\tilde{q}\| = \|\tilde{q}\|$, i.e., they refer to the same superoperator. Thus, any $(Q, \lambda)$-diamond norm between them is 0. We choose $Q = I$ and $\lambda = 0$.

(2) **Unitary:** For the same reason as the proof for Skip.

(3) **Weaken:** By induction, the premise $(P'Q') + \tilde{P} \leq \epsilon'$ implies
\[
\forall \rho, \text{tr}(Q'\rho) \geq \lambda' \text{tr}(\rho) \Rightarrow \frac{1}{2} \|\|\tilde{P}\|\|\rho - \|\|P\|\|\rho\|\| \leq \epsilon' \text{tr}(\rho).
\]
For any density matrix $\rho$, constants $0 \leq \lambda' \leq \lambda \leq 1$, and predicates $Q \subseteq Q'$, $\text{tr}(Q\rho) \geq \lambda' \text{tr}(\rho)$ implies that $\text{tr}(Q'\rho) \geq \lambda' \text{tr}(\rho)$. And for $0 \leq \epsilon' \leq \epsilon \leq 1$, $\frac{1}{2} \|\|\tilde{P}\|\|\rho - \|\|P\|\|\rho\|\| \leq \epsilon' \text{tr}(\rho)$ implies that $\frac{1}{2} \|\|\tilde{P}\|\|\rho - \|\|P\|\|\rho\|\| \leq \epsilon \text{tr}(\rho)$. Therefore, we have that
\[
\forall \rho, \text{tr}(Q\rho) \geq \lambda \text{tr}(\rho) \Rightarrow \frac{1}{2} \|\|\tilde{P}\|\|\rho - \|\|P\|\|\rho\|\| \leq \epsilon \text{tr}(\rho).
\]
So $\tilde{P}$ is $\epsilon$-robust under $(Q, \lambda)$.

(5) **Rescale:** This rule follows by observing that the condition $\text{tr}(\rho Q) \geq \lambda$ is equivalent to the condition $\text{tr}(\rho Q/\delta) \geq \lambda/\delta$ for $\delta > 0$. We only require that $Q, Q/\delta$ and $\lambda, \lambda/\delta$ are well defined, namely, $0 \leq Q, Q/\delta \subseteq I$ and $0 \leq \lambda, \lambda/\delta \leq 1$.

(6) **Sequence:** For every state $\rho$,
\[
\|\|\tilde{P}_1; \tilde{P}_2\|\|\rho - \|\|P_1; P_2\|\|\rho\|\| \leq \|\|\tilde{P}_1\|\|\rho - \|\|P_1\|\|\rho\|\| + \|\|\tilde{P}_2\|\|\rho - \|\|P_2\|\|\rho\|\|.
\]
The inequality $\|\|\tilde{P}_2\|\|\tilde{P}_1\|\|\rho - \|\|P_2\|\|\tilde{P}_1\|\|\rho\|\| \leq \|\|\tilde{P}_1\|\|\rho - \|\|P_1\|\|\rho\|\|$ follows because quantum superoperators are contractive.

Now assume that $\text{tr}(Q_1\rho) \geq \lambda \text{tr}(\rho)$. By induction, the premise $(Q_1, \lambda) + \tilde{P}_1 \leq \epsilon_1$ implies that $\frac{1}{2} \|\|\tilde{P}_1\|\|\rho - \|\|P_1\|\|\rho\|\| \leq \epsilon_1 \text{tr}(\rho)$. Also, by the premise $\{Q_1\} P_1 \{Q_2\}$, we have that $\text{tr}(Q_2\tilde{P}_1\rho) \geq \text{tr}(Q_1\rho) \geq \lambda \text{tr}(\rho) \geq \lambda \text{tr}(\tilde{P}_1\rho)$. Now we can use our induction hypothesis and the premise $(Q_2, \lambda) + \tilde{P}_2 \leq \epsilon_2$ to conclude $\frac{1}{2} \|\|\tilde{P}_2\|\|\rho - \|\|P_2\|\|\rho\|\| \leq \epsilon_2 \text{tr}(\tilde{P}_1\rho)$. So, finally, we have that
\[
\frac{1}{2} \|\|\tilde{P}_1; \tilde{P}_2\|\|\rho - \|\|P_1; P_2\|\|\rho\|\| \leq \epsilon_1 \text{tr}(\rho) + \epsilon_2 \text{tr}(\tilde{P}_1\rho) \leq (\epsilon_1 + \epsilon_2) \text{tr}(\rho).
\]

(7) **Case:** Let $\tilde{P} = \text{case } M[\tilde{q}] = m \rightarrow \tilde{P}_m$. Assume the input state $\rho$ to the case statement satisfies $\sum_m \rho M^\dagger M \rho M_m \geq \lambda' \text{tr}(\rho)$. To leverage the premise $(Q_m, \lambda) + \tilde{P}_m \leq \epsilon$ and the induction hypothesis to conclude that $\frac{1}{2} \|\|\tilde{P}_m\|\|\rho - \|\|P_m\|\|\rho\|\| \leq \epsilon \text{tr}(\rho)$, one must show the precondition $(Q_m, \lambda)$ holds for state $\rho$ entering branch $m$. A naive approach is to show that $\text{tr}(M^\dagger M \rho M_m) \geq \lambda \text{tr}(M^\dagger M \rho M_m)$ holds for every branch $m$, which implies that $\text{tr}(Q_m \rho M_m) \geq \lambda \text{tr}(Q_m \rho M_m)$ where $M_m \rho M_m$ is the (sub-normalized) post-measurement state entering branch $m$. We argue that this is in general impossible when $\lambda' = \lambda$. For instance, consider a collection of projective measurement operators $\{M_m\}$. If there exists a branch $i$ and a state $\rho$ supported on $M_i$ such that $\text{tr}(M_i^\dagger Q_i M_i \rho) \geq \lambda \text{tr}(\rho)$ for some $\lambda > 0$, obviously
\[ \sum_{m} M_m^i Q_m M_m^i \rho \geq \lambda tr(\rho), \] but none of the preconditions is satisfied except for the one for branch \( i \) since \( tr(M_j^i Q_j M_j \rho) = 0 \) for each \( j \neq i \).

Instead, we show that for a majority of the clauses \( tr(M_m^i Q_m M_m \rho) \geq \lambda tr(M_m^i M_m^i \rho) \) holds for some \( \lambda \) strictly less than \( \lambda' \). To that end, let \( p_m = tr(M_m^i M_m^i \rho) \) and \( q_m = tr(M_m^i Q_m M_m \rho) \).

Define \( \delta_m \) to be such that \( q_m = (1 - \delta_m) p_m \). Note that \( 0 \leq \delta_m \leq 1 \) because \( 0 \leq q_m \leq p_m \) for every \( m \). Without loss of generality we assume \( tr(\rho) = 1 \). Let \( S(\rho) \) denote the collection of branches such that the precondition \( (Q_m, \lambda) \) holds, i.e., \( S(\rho) = \{ m : q_m \geq \lambda p_m \}, \) i.e., \( \delta_m \leq 1 - \lambda' \). We will determine a lower bound for \( \sum_{m \in S(\rho)} p_m \) for each state \( \rho \) using a probabilistic argument.

First, note that \( \{p_m\} \) is a probability distribution since \( \sum_m p_m = 1 \) and \( p_m \geq 0 \) for each \( m \). Also, since (by our assumption) \( \sum_m q_m \geq \lambda' \sum_m p_m = \lambda' \), we have that \( \sum_m \delta_m p_m \leq (1 - \lambda') \). Now we can define a random variable \( \Delta \) to be such that \( \Pr[\Delta = \delta_m] = p_m \) for each \( m \). The expected value of \( \Delta \) is \( E[\Delta] \leq (1 - \lambda') \), and Markov’s inequality yields

\[ \sum_{m \in S(\rho)} p_m = \Pr[\Delta \geq 1 - \lambda] \leq \frac{E[\Delta]}{1 - \lambda} \leq \frac{1 - \lambda'}{1 - \lambda} =: t. \quad (6.12) \]

This says that a weighted fraction \( t \) of the branches will not satisfy the precondition \( (Q_m, \lambda) \). Note that \( t \in [0, 1] \) since \( \lambda \leq \lambda' \). For \( m \notin S(\rho) \), only the trivial upper bound \( \epsilon_m = 1 \) is guaranteed. Therefore, for each state \( \rho \),

\[
\frac{1}{2} \|\|P\|\rho - \|P\|\rho\|_1 = \frac{1}{2} \sum_m \|\|P_m\|\| (M_m^i (M_m^i \rho M_m^i) - \|\|P_m\|\| (M_m^i \rho M_m^i)) \|_1 \quad (6.13)
\]

\[
\leq \frac{1}{2} \sum_m \|\|P_m\|\| (M_m^i (M_m^i \rho M_m^i) - \|\|P_m\|\| (M_m^i \rho M_m^i)) \|_1 \quad (6.14)
\]

\[
\leq \sum_{m \in S(\rho)} \text{tr}(M_m^i M_m^i \epsilon) + \sum_{m \notin S(\rho)} \text{tr}(M_m^i M_m^i \epsilon) \quad (6.15)
\]

\[
\leq (1 - t) \epsilon + t = ((1 - t) \epsilon + t) \text{tr}(\rho). \quad (6.16)
\]

Rewriting \( \lambda = 1 - \delta \) for ease of notation, we have \( \lambda' = 1 - t \delta \). Finally, we note that the case \( t = 0 \) implies that the precondition of each branch is satisfied, and the error of the case statement can be bounded by \( \epsilon \).

(8) While-Bounded: Let \( P \equiv \text{while } M[q] = 1 \text{ do } P_1 \text{ done} \). Let \( S_k \) be the bounded while loop of \( k \) iterations. Define the linear maps \( \tilde{\Sigma}(\rho) := [P_1] \| (M_i \rho M_i^i) \| \) and \( \tilde{\Sigma}(\rho) := [P_1] \| (M_i \rho M_i^i) \| \) and let \( \|S_k\| \rho = M_0 \rho M_0^i + \|S_{k-1}\|(\Sigma(\rho)) \) for \( k \geq 1 \) (with \( \|S_0\| \rho = \rho \)). In order to bound the distance between \( \|P\| \) and \( \tilde{\|P\|} \), we first upper bound the distance between \( \|S_k\| \) and \( \|S_k\| \) and then
we have

\[
\frac{1}{2} \|\|S_k\|\|_1 \leq \frac{1}{2} \|\|S_k-1\|\|_1 \leq \frac{1}{2} \|\|E\|\|_1 + \frac{1}{2} \|\|E\|\|_1
\]

(6.19)

which shows that

\[
\lambda \geq \frac{1}{2} \|\|E\|\|_1 \leq \frac{1}{2} \|\|E\|\|_1
\]

(6.20)

The second inequality (6.18) follows from a technique similar to the one used in the proof of the Sequence rule. We bound the first term in (6.18) by \(\epsilon \text{tr}(M_i \rho M_i)\) by applying the premise \((Q, \lambda) \vdash P_i \leq \epsilon\) and the induction hypothesis. We will prove the post-measurement state \(M_i \rho M_i\) indeed satisfies the precondition \((Q, \lambda)\) later. Similarly, each term in (6.19) is bounded above by \(\epsilon \text{tr}(M_i \rho M_i)\) and thus the inequality in (6.20) holds.

To establish the inequality in (6.21), let \(b_i := \text{tr}(M_i \rho M_i)\). Then the sequence \(\{b_k\}\) is non-negative and non-increasing. We now prove an upper bound of the series. Since \(P\) is \((a, n)\)-bounded, we know that

\[
\text{tr}(M_i \rho M_i) = \text{tr}(E^a(M_i \rho M_i)) \leq \text{at}(M_i \rho M_i), \quad a < 1
\]

(6.22)

for every state \(\sigma\), and therefore \(b_i \leq ab_i\) for every \(i\). Since \(b_k\) is non-increasing, we know that

\[
\sum_{i=0}^{k-1} b_i = \sum_{m=0}^{[k/n]-1} (b_{nm} + \ldots + b_{nm+n-1}) \leq n \sum_{m=0}^{[k/n]-1} a^m b_0 = \frac{n(1 - a^{[k/n]})b_0}{1 - a}.
\]

(6.23)

Thus the inequality in (6.21) holds. Since \(b_0 = \text{tr}(M_i \rho M_i)\), we have

\[
\frac{1}{2} \|\|S_k\|\|_1 \leq \frac{1}{2} \|\|E\|\|_1 \leq \frac{ne(1 - a^{[k/n]})}{1 - a} \text{tr}(\rho).
\]

(6.25)

Taking the limit as \(k \to \infty\), we have that \(a^{[k/n]} \to 0\), which shows that \(\frac{1}{2} \|\|S_k\|\|_1 \leq \frac{ne(1 - a^{[k/n]})}{1 - a} \text{tr}(\rho),\) as desired.

In order to apply the premise \((Q, \lambda) \vdash P_i \leq \epsilon\) to states of the form \(M_i \rho M_i\), we need to show that \(\text{tr}(QM_i \rho M_i) \geq \lambda \text{tr}(M_i \rho M_i)\) for each \(i\). We can prove this by induction. For the base case \(i = 0\), by the preconditions \((\lambda M_0\|M_0 + M_i QM_i)\), we have \(\text{tr}(QM_i \rho M_i) \geq \lambda \text{tr}(\rho) - \lambda \text{tr}(M_i M_0\rho) = \lambda \text{tr}(M_i \rho M_i)\). Therefore, \(\text{tr}(QM_i \rho M_i) \geq \lambda \text{tr}(M_i \rho M_i)\).

For the inductive step, observe that the Hoare triple \(\{Q\} P_i \{\sigma\}\) yields \(\text{tr}(R\|P_i\|\sigma) \geq \text{tr}(Q\sigma)\) for \(R = \lambda M_0\|M_0 + M_i QM_i\) and all states \(\sigma\). By the induction hypothesis, we have

\[
\text{tr}(R\|P_i\|\sigma) \geq \text{tr}(QM_i \rho M_i) \geq \lambda \text{tr}(M_i \rho M_i) \geq \lambda \text{tr}(E^{i+1})\).
\]
where the last inequality holds because quantum operations are trace-non-increasing (applied to \([P_1]\)). Note that \([P_1](M_1 \mathcal{E}^i M_1^\dagger) = \mathcal{E}^{i+1}\). Now by substituting \(R\), we have

\[
\text{tr}(M_1^\dagger Q M_1 \mathcal{E}^{i+1}(\rho)) \geq \lambda \text{tr}(\mathcal{E}^{i+1}(\rho)) - \lambda \text{tr}(M_0^\dagger M_0 \mathcal{E}^{i+1}(\rho)) = \lambda \text{tr}(M_1^\dagger M_1 \mathcal{E}^{i+1}(\rho)).
\]

(6.26)

Or equivalently, \(\text{tr}(Q M_1 \mathcal{E}^{i+1}(\rho) M_1^\dagger) \geq \lambda \text{tr}(M_1 \mathcal{E}^{i+1}(\rho) M_1^\dagger)\), which concludes the proof.

(9) \textit{While-Unbounded}: the proof is trivial as we use the trivial upper bound 1.

\[\square\]

7 CASE STUDIES

In this section, we apply our robustness definition and logic to two example quantum programs: the quantum Bernoulli factory and the quantum walk. We also demonstrate the (in)effectiveness of different error correction schemes on single-qubit errors and analyze the robustness of a fault-tolerant version of QBF. Some computational details are deferred to appendices in the supplemental material.

7.1 Quantum Bernoulli Factory

The quantum Bernoulli factory (QBF) [Dale et al. 2015] is the quantum equivalent of the classical Bernoulli factory problem [Keane and O’Brien 1994]. In the classical Bernoulli factory problem, given a function \(f : [0, 1] \mapsto [0, 1]\) and a coin that returns heads with unknown probability \(p\), the goal is to simulate a new coin that returns head with probability \(f(p)\). In QBF, the goal is to generate the state \(|f(p)\rangle\) given a description of \(f\) and a quantum coin described by the state

\[
|p\rangle := \sqrt{p}|0\rangle + \sqrt{1-p}|1\rangle.
\]

(7.1)

QBF is interesting because it can simulate a strictly larger class of functions \(f\) than can be simulated by its classical counterpart. One example of a function that can be simulated by QBF, but not by the classical Bernoulli factory, is the probability amplification function. The key to simulating the probability amplification function is producing the state \(|f(p)\rangle = (2p - 1)|0\rangle + 2\sqrt{p(1-p)}|1\rangle\). This state can be prepared by (the ideal version of) the following program [Li and Ying 2018]:

\[
QBF \equiv q_1 := |1\rangle; q_2 := |1\rangle;
\text{while } M[q_2] = 1 \text{ do}
\]

\[
q_1 := |0\rangle; q_2 := |0\rangle;
\]

\[
q_1 := p_v \Phi_v V[q_1]; q_2 := p_v \Phi_v V[q_2];
\]

\[
q_1, q_2 := u, \Phi_u U[q_1, q_2] \text{ done},
\]

where \(M\) is the standard basis measurement \(|0\rangle, |1\rangle\}. The unitary \(U\) is defined by

\[
U = |01\rangle \langle \phi^+ | + |00\rangle \langle \phi^- | + |10\rangle \langle \psi^+ | + |11\rangle \langle \psi^- |
\]

(7.2)

acting on \(|0\rangle\) generates the state \(|p\rangle\). We denote the error due to noisy unitary application of \(V\) and \(U\) by \(\epsilon_V = p_v \|\Phi_V - V \circ V^\dagger\|_o\) and \(\epsilon_U = p_U \|\Phi_U - U \circ U^\dagger\|_o\) respectively. Note that to simplify our discussion we compute \(\epsilon_V\) and \(\epsilon_U\) using the trivial precondition \((I, 0)\). Recall that \(|\cdot\|_o = \|\cdot\|_o\).

Now we prove that \((I, 0) + QBF \leq 4\epsilon_V + 2\epsilon_U\). First, by the \textit{Sequence} and \textit{Unitary} rules, we bound the error in the loop body by \(2\epsilon_V + \epsilon_U\). To show \((a, n)\)-boundedness of the loop, we must consider the behavior of \(\mathcal{E}^*\) where

\[
\mathcal{E}^* = (M_1 \circ M_1^\dagger)^* \circ \left[[q_1 := |p\rangle; q_2 := |p\rangle]\right]^* \circ \left[[U[q_1, q_2]]^\dagger\right]^*.
\]

(7.3)
We evaluate $E^*(M_1^\dagger M_1)$ as follows. Recall that $M_1 = I \otimes |1\rangle\langle 1|$. 
\[
M_1^\dagger M_1 = I \otimes |1\rangle\langle 1| \stackrel{[U|q, q]|^*}{\mapsto} |\phi^+\rangle\langle \phi^+| + |\psi^-\rangle\langle \psi^-| \quad \frac{\|q_1:=|p\rangle; q_2:=|p\rangle\|}{2} \rightarrow I \otimes I \quad (7.4)
\]

This shows that $E^*(M_1^\dagger M_1) = \frac{1}{2} M_1^\dagger M_1$. Therefore, the while loop is $(\frac{1}{2}, 1)$-bounded. Now we can use the While-Bounded rule to conclude that the error bound for the loop is $\frac{2\epsilon_V + \epsilon_U}{1-\frac{1}{2}} = 4\epsilon_V + 2\epsilon_U$, and therefore $(I, 0) \vdash (\text{while } M[q_2] = 1 \text{ do } \bar{P}_1 \text{ done}) \leq 4\epsilon_V + 2\epsilon_U$ where $\bar{P}_1$ is the body of the loop. Two additional applications of the Sequence rule conclude the proof.

We can compute $\epsilon_V$ and $\epsilon_U$ by identifying the error probabilities $p_V, p_U$ and error models $\Phi_V, \Phi_U$, and numerically calculating the diamond norm. For example, consider the case where the state preparation is ideal, i.e., $p_V = 0$, and noise in the application of $U$ is characterized by $p_U = 10^{-5}$ and $\Phi_U = (\frac{X \otimes X + Y \otimes Y + Z \otimes Z + I \otimes I}{2}) \otimes ^2$, the 4-dimensional depolarizing channel. Then $\epsilon_V = 0$ and $\epsilon_U = p_U \parallel \Phi_U - U \circ U^\dagger \parallel_\diamond$. Applying an SDP solver [da Silva 2015; Watrous 2009] for the calculation of the diamond norm, we find that $\epsilon_U = 1 \times 10^{-3} \times 0.9375 = 0.9375 \times 10^{-4}$, and therefore the error bound of $\text{QBF}$ is $1.875 \times 10^{-5}$. More details can be found in Appendix A.2.

### 7.2 Quantum Walk on a Circle

Here we analyze the error of the quantum walk algorithm introduced in Section 4. The noisy quantum walk on a circle with $n$ points can be written as the following program:

\[
\overline{QW}_n \equiv p := |0\rangle; c := |L\rangle; \text{while } M[p] = 1 \text{ do } c := p_h, \Phi_H H[c]; c, p := p_s, \Phi_S S[c, p] \text{ done.} \quad (7.6)
\]

Now we show that $(I, 0) \vdash \overline{QW}_6 \leq 30(\epsilon_H + \epsilon_S)$ where $\epsilon_H = p_H \parallel \Phi_H - H \circ H^\dagger \parallel_\diamond$ and $\epsilon_S = p_S \parallel \Phi_S - S \circ S^\dagger \parallel_\diamond$, are the errors due to noisy application of $H$ and $S$ respectively. First, by the Sequence and Unitary rules, we bound the error in the loop body by $\epsilon_H + \epsilon_S$. Next, we numerically test increasing values of $(a, n)$ until we find a pair that satisfies (6.10). For this program, we find that the pair $(\frac{5}{6}, 5)$ satisfies the inequality, i.e., $(E^*)^3(M_1^\dagger M_1) \subseteq \frac{5}{6} M_1^\dagger M_1$ where $E^* = (M_1^\dagger \circ M_1) \circ \parallel c, p := S[c, p] \parallel^2 \circ \parallel c := H[c] \parallel^2$. This implies that the loop is $(\frac{5}{6}, 5)$-bounded. Note that here, unlike in the previous example, we have computed the $(a, n)$ values numerically. This may be useful in cases where direct deduction of $a$ and $n$ is difficult. Now we can use the While-Bounded rule to conclude that the error bound for the loop is $\frac{3(\epsilon_H + \epsilon_S)}{1-\frac{1}{2}} = 30(\epsilon_H + \epsilon_S)$, and therefore $(I, 0) \vdash (\text{while } M[p] = 1 \text{ do } \bar{P}_1 \text{ done}) \leq 30(\epsilon_H + \epsilon_S)$ where $\bar{P}_1$ is the body of the loop. Two additional applications of the Sequence rule conclude the proof.

As an example, consider the program $\overline{QW}_6$ where only the Hadamard gate may be faulty, i.e., $p_S = 0$. Say that noisy Hadamard application is characterized by $p_H = 5 \times 10^{-5}$ and $\Phi_H(\rho) = \frac{X \otimes X + Y \otimes Y + Z \otimes Z + I \otimes I}{2}$, the 2-dimensional depolarizing channel. Applying an SDP solver [da Silva 2015; Watrous 2009, 2013] for the calculation of the diamond norm, we find that $\epsilon_H = 5 \times 10^{-5} \times 0.75 = 3.75 \times 10^{-5}$. Therefore the error of $\overline{QW}_6$ is $30\epsilon_H = 1.125 \times 10^{-3}$. More details can be found in Appendix A.2.

### 7.3 Error Correction

In this section we use our semantics to show that an error correction scheme that is appropriate for the error model can reduce noise in a program, while an inappropriate error correction scheme may...
do the opposite. We consider three programs: $P_1$, $P_2$, and $P_3$. $P_1$ performs the identity operation, i.e., $P_1 \equiv q := I[q]$. The noisy version of $P_1$, $\tilde{P}_1$, allows error during the application of $I$. $P_2$ and $P_3$ are semantically equivalent to $P_1$ (in the sense that they correspond to the same superoperator\(^{12}\)), but both employ error correction. $P_2$ uses a three-qubit repetition code that can correct a bit flip ($X$ error) on a single qubit and $P_3$ uses a three-qubit repetition code that can correct a phase flip ($Z$ error) on a single qubit. $P_2$ and $P_3$ both have the following form:

$$
\tilde{q} := ENCODE[q]; \\
\bar{q} := \tilde{I}[\tilde{q}]; \\
\bar{q} := CORRECT[\tilde{q}]; \\
q := DECODE[\bar{q}]
$$

where we use $ENCODE$ as a stand-in for the operations associated with turning the qubit $q$ into its encoded counterpart $\tilde{q}$, $CORRECT$ as a stand-in for the operations associated with syndrome detection and error correction, and $DECODE$ as a stand-in for the operations associated with converting the encoded qubit $\bar{q}$ back into $q$. We use $\tilde{I}$ to represent a fault-tolerant version of the identity operation. In the noisy programs $\tilde{P}_2$ and $\tilde{P}_3$, we only allow error during application of $\tilde{I}$. To simplify the calculation, we will assume that encoding, decoding, and error correction are all ideal (i.e. they have an error probability of 0). For a description of $ENCODE$, $CORRECT$, and $DECODE$ see Appendix A.3.

For all three programs, we define the error model acting on a single qubit by $\Phi(\rho) = X\rho X$. With this error model, application of an $I$ gate to a qubit will succeed with probability $1 - p$, and will instead become an application of a $X$ gate with probability $p$. Now we can use our definition of the denotational semantics from Section 5 to directly compute $\|\tilde{P}_1\|$, $\|\tilde{P}_2\|$, and $\|\tilde{P}_3\|$. We find that:

$$
\|\tilde{P}_1\|\rho = (1 - p)I\rho I + pX\rho X \\
\|\tilde{P}_2\|\rho = ((1 - p)^3 + 3p(1 - p)^2)I\rho I + (3p^2(1 - p) + p^3)X\rho X \\
\|\tilde{P}_3\|\rho = ((1 - p)^3 + 3p^2(1 - p)^2)I\rho I + (3p(1 - p) + p^3)Z\rho Z
$$

Now we can directly compute $\|\|P_1\| - \|\tilde{P}_1\|\|_\delta$, $\|\|P_2\| - \|\tilde{P}_2\|\|_\delta$, and $\|\|P_3\| - \|\tilde{P}_3\|\|_\delta$ to determine error rates for these programs. We find that:

$$
\|\|P_1\| - \|\tilde{P}_1\|\|_\delta = p \\
\|\|P_2\| - \|\tilde{P}_2\|\|_\delta = 3p^2 - 2p^3 \\
\|\|P_3\| - \|\tilde{P}_3\|\|_\delta = 3p(1 - p)^2 + p^3
$$

This allows us to conclude that, under $Q = I$ and $\lambda = 0$,

- $\tilde{P}_1$ is $p$ robust
- $\tilde{P}_2$ is $3p^2 - 2p^3$ robust
- $\tilde{P}_3$ is $3p(1 - p)^2 + p^3$ robust

Note that for $0 < p < \frac{1}{2}$, we have that $3p^2 - 2p^3 < p$ and $p < 3p(1 - p)^2 + p^3$, which tells us that, in the presence of $X$ errors, correcting for bit flips will improve the error rate while correcting for phase flips will make the error rate worse. For computational details see Appendix A.4.

Here we invoke the semantics of the program to prove quantum robustness by definition. This is an expensive calculation. However, it is necessary to account for the effect of error correction. Ideally, we imagine a combination of uses of the semantics and the rules to trade off between the cost of the calculation and the accuracy of the bounds.

\(^{12}\)To make this mathematically rigorous, one needs to distinguish between variables and ancillas in the program. The semantics refers to the superoperator that traces out the ancilla part.
7.4 Fault-tolerant Quantum Bernoulli Factory

In this section, we consider a fault-tolerant implementation of the quantum Bernoulli factory. The fault-tolerant QBF, denoted by $\overline{QBF}$, can be written as follows.

$$\overline{QBF} \equiv q_1 := |1\rangle; \ q_2 := |1\rangle;$$

while $M[q_2] = 1$

$$q_1 := |0\rangle; \ q_2 := |0\rangle;$$

$$\overline{q}_1 := ENCODE[q_1]; \ \overline{q}_2 := ENCODE[q_2];$$

$$\overline{q}_1 := CORRECT[\overline{q}_1];$$

$$\overline{q}_2 := CORRECT[\overline{q}_2];$$

$$\overline{q}_1 := CORRECT[\overline{q}_1]; \ \overline{q}_2 := CORRECT[\overline{q}_2];$$

$$q_1 := DECODE[\overline{q}_1]; \ q_2 := DECODE[\overline{q}_2]$$

done

where $ENCODE$, $CORRECT$, and $DECODE$ have the same meanings as in the previous section. As before, we assume that encoding, decoding, and error correction are not affected by noise. $\overline{V}$ and $\overline{U}$ are the fault-tolerant operators that correspond to $V$ and $U$ respectively. We define the error model associated with $\overline{V}$ by $\Phi_\overline{V}(\rho) = \Phi_\overline{V}^\dagger(\overline{V}\rho\overline{V}^\dagger)$ where $\Phi_\overline{V}(\rho) = (1 - p_V)I\rho I + p_V X\rho X$. Similarly, we define the error model associated with $\overline{U}$ by $\Phi_\overline{U}(\rho) = \Phi_\overline{U}^\dagger(\overline{U}\rho\overline{U}^\dagger)$ where $\Phi_\overline{U}(\rho) = (1 - p_U)I\rho I + p_U X\rho X$. Note that because we are using a three-qubit repetition code, $\overline{V}$ and $\overline{U}$ will be 3-qubit and 6-qubit unitaries respectively. These noise models are applied with probability one.

Using the definitions from Section 5 we can show that for $i \in \{1, 2\}$,

$$\|\overline{q}_i \equiv I_{1,\Phi_\overline{V}} \overline{V}[\overline{q}_i]; \overline{q}_i := CORRECT[\overline{q}_i]\|\rho = (1 - q_V)\overline{V}\rho\overline{V}^\dagger + q_V X^{\otimes 3}\overline{V}\rho\overline{V}^\dagger X^{\otimes 3}, \quad (7.13)$$

where $q_V = 3p_V^2(1 - p_V) + p_V^3$. Note that (7.13) is obtained through a calculation similar to the one used to compute $\|P_2\|$ in Section 7.3. This says applying $\overline{V}$ followed by error correction is equivalent to applying $\overline{V}$ with probability $1 - q_V$, and applying $\overline{V}$ followed by $X^{\otimes 3}$ with probability $q_V$. Thus we can use the Unitary rule to compute the error of the noisy application of $\overline{V}$ followed by error correction:

$$\epsilon_{\overline{V}} = q_V \|(X^{\otimes 3} \circ X^{\otimes 3}) \circ (\overline{V} \circ \overline{V}^\dagger) - (\overline{V} \circ \overline{V}^\dagger)\|_\circ = q_V \|(X^{\otimes 3} \circ X^{\otimes 3}) - (I \circ I)\|_\circ = q_V. \quad (7.14)$$

The final equality holds because the unitaries are perfectly distinguishable. Note that $\|(\Phi(U \circ U^\dagger) - U \circ U^\dagger)\|_\circ = \|(\Phi - I \circ I)\|_\circ$ holds for any superoperator $\Phi$ and unitary $U$. Similarly,

$$\|\overline{q}_1, \overline{q}_2 \equiv I_{1,\Phi_\overline{U}} \overline{U}[\overline{q}_1, \overline{q}_2]; \overline{q}_1 := CORRECT[\overline{q}_1]; \overline{q}_2 := CORRECT[\overline{q}_2]\|\rho$$

$$= ((1 - q_U)I \circ I + q_U X^{\otimes 3} \circ X^{\otimes 3}) \circ (\overline{U}\rho\overline{U}^\dagger) \quad (7.15)$$

$$= ((1 - q_U)I \circ I + q_U X^{\otimes 3} \circ X^{\otimes 3} \circ (\overline{U}\rho\overline{U}^\dagger), \quad (7.16)$$
where \( q_U = 3p_U^2(1 - p_U) + p_U^3 \). Thus, by the Unitary rule, the error of the noisy application of \( \overline{U} \) followed by error correction is

\[
\epsilon_T^{\overline{U}} = \frac{1}{2q_U - q_U^2} (q_U^2(X^{\otimes 6} \rho X^{\otimes 6}) + q_U(1 - q_U)(I^{\otimes 3} \otimes X^{\otimes 3})\rho(I^{\otimes 3} \otimes X^{\otimes 3}) + q_U(1 - q_U)(X^{\otimes 3} \otimes I^{\otimes 3}) - (2q_U - q_U^2)(I \circ I)) \quad (7.21)
\]

The superoperator \( E \) is trace-preserving, so \( \epsilon_T^{\overline{U}} = (2q_U - q_U^2)\|E - I \circ I\|_\diamond \leq (2q_U - q_U^2). \) To show \( \epsilon_T^{\overline{U}} \) is also lower bounded by \( 2q_U - q_U^2 \), it suffices to consider the input state \( \rho = (|0\rangle\langle 0|)^{\otimes 6} \) and the projector \( \Pi = I - (|0\rangle\langle 0|)^{\otimes 6} \). By definition, \( \epsilon_T^{\Pi} = (2q_U - q_U^2)\|E - I \circ I\|_\diamond \geq (2q_U - q_U^2)\|\Pi(E(\rho) - \rho)\|_\diamond = 2q_U - q_U^2. \)

Now using the argument given in Section 7.1, we can show that \((I, 0) \vdash \overline{QB} \leq 4\epsilon_V + 2\epsilon_U. \) Note that ENCODE and DECODE do not impact the robustness. Without error correction, as shown in Section 7.1, \((I, 0) \vdash \overline{QB} \leq 4\epsilon_V + 2\epsilon_U \) where \( \epsilon_V = \|\Phi_V - I \circ I\|_\diamond = p_V \) and \( \epsilon_U = \|\Phi_U^2 - I \circ I\|_\diamond = 2p_U - p_U^2. \) In the case where the error rate is constant, i.e., \( p_V = p_U = p \) for some probability \( p \), \((I, 0) \vdash \overline{QB} \leq O(p) \) and \((I, 0) \vdash \overline{QB} \leq O(p^2). \) This shows that the error of \( \overline{QB} \) is suppressed by a factor of \( p \) with a fault-tolerant implementation of the loop body.

Remark 7.1. Throughout our example, we make the assumption that operations like ENCODE, CORRECT and DECODE are noise-free. In the actual setting of fault-tolerant quantum computation, they can also contain noise. In order to suppress the error rate, one may need to use the construction of fault-tolerant gadgets in the proof of the threshold theorem [Aharonov and Ben-Or 1997]. We remark that the assumption we made is just to simplify the example and ease presentation. It would in fact be possible to prove a threshold theorem in our formalism without this assumption, although the calculation could be much more complicated. It is an interesting open question to see whether one can simplify this calculation by adding more rules to our logic.

8 Conclusions and Future Work

We have presented a semantics for describing quantum computation with errors and an analysis that bounds the distance between the result of a noisy program and its corresponding ideal program on the same input. We used our analysis to compute error bounds for noisy versions of the quantum Bernoulli factory and quantum walk programs. We also showed how our analysis can be used to compute the error bounds for small circuits with and without error correction, showing examples of when using error correction is beneficial and when there are tradeoffs between the efficiency of error corrections and related costs.

A natural next step for our work is to encode the rules from Section 6.3 in a proof assistant so that they can be applied in an automated fashion. We have shown that the \((Q, \lambda)\)-diamond norm can be computed by an SDP, and the \((a, n)\) values for loop boundedness can be computed analytically or numerically, so implementing our rules is feasible. Given an implementation, we are interested how our analysis may be used to construct circuits with lower error rates. This application is inspired by work by Misailovic et al. [2014], which uses classical reliability analysis to determine which

operations can be replaced by their noisy counterparts. It could hence be used to inform decisions about which implementations of quantum algorithms are practical for near-term use.

ACKNOWLEDGMENTS
We would like to thank Andrew Childs for helpful discussions on quantum walks. This material is based upon work supported 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, and the Canadian Institute for Advanced Research.

REFERENCES

Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani. 2001. Quantum Walks on Graphs. In STOC.

Dorit Aharonov and Michael Ben-Or. 1997. Fault-tolerant Quantum Computation with Constant Error. In STOC.


James Bornholt, Todd Mytkowicz, and Kathryn S. McKinley. 2014. Uncertain\langle T\rangle: A First-order Type for Uncertain Data. In ASPLOS.

Brett Boston, Adrian Sampson, Dan Grossman, and Luis Ceze. 2015. Probability Type Inference for Flexible Approximate Programming. In OOPSLA.


Michael Carbin, Sasa Misailovic, and Martin C. Rinard. 2013. Verifying Quantitative Reliability for Programs That Execute on Unreliable Hardware. In OOPSLA.


Swarat Chaudhuri, Sumit Gulwani, and Roberto Lublimeran. 2010. Continuity Analysis of Programs. In POPL.


A APPENDIX

A.1 Proof that the $(Q, \lambda)$-diamond Norm is a Seminorm

Here we prove that $(Q, \lambda)$-diamond norm is a seminorm. Recall its definition.

**Definition A.1** $(Q, \lambda)$-diamond norm. Given superoperators $E, E'$, quantum predicate $Q$ over $\mathcal{H}$, and $0 \leq \lambda \leq 1$, the $(Q, \lambda)$-diamond norm between $E$ and $E'$, denoted $\|E - E'\|_{Q, \lambda}$, is defined by

$$\|E - E'\|_{Q, \lambda} = \max_{\rho \in \mathcal{D}(\mathcal{H} \otimes \mathcal{A}) : \text{tr}(\rho) = 1, \text{tr}(Q\rho) \geq \lambda} T(E \otimes I_A, E' \otimes I_A(\rho)), \quad (A.1)$$

where $A$ is any auxiliary space and can be assumed to be $\mathcal{H}$ without loss of generality.

**Proof.** Recall from (6.2) that $0 \leq T(E \otimes I_A(\rho), E' \otimes I_A(\rho)) \leq 1$ for any quantum state $\rho$, so $\|E - E'\|_{Q, \lambda} \geq 0$.

1. **Positive scalability**: This property is inherited directly from the diamond norm. One may also observe that for any $\alpha \in \mathbb{C},$

$$T(E \otimes I_A(\alpha\rho), E' \otimes I_A(\alpha\rho)) = T(\alpha(E \otimes I_A(\rho)), \alpha(E' \otimes I_A(\rho))) = \alpha T(E \otimes I_A(\rho), E' \otimes I_A(\rho)).$$

2. **Triangle Inequality**: For quantum superoperators $E, E', F, F'$, we have

$$\|(E - E') + (F - F')\|_{Q, \lambda} = \max_{\rho \in \mathcal{D}(\mathcal{H} \otimes \mathcal{A}) : \text{tr}(\rho) = 1, \text{tr}(Q\rho) \geq \lambda} T((E + F) \otimes I_A(\rho), (E' + F') \otimes I_A(\rho)).$$

Suppose the maximal value on the right hand side is attained at a state $\rho^* \in \mathcal{D}(\mathcal{H} \otimes \mathcal{A})$. Note that, by definition, this requires $\text{tr}(\rho^*) = 1, \text{tr}(Q\rho^*) \geq \lambda$. Then,

$$\|(E - E') + (F - F')\|_{Q, \lambda} = T((E + F) \otimes I_A(\rho^*), (E' + F') \otimes I_A(\rho^*))$$

$$= \max_{0 \in P_I} \text{tr}(P((E + F) \otimes I_A(\rho^*) - (E' + F') \otimes I_A(\rho^*)))$$

$$= \max_{0 \in P_I} \text{tr}(P((E - E') \otimes I_A(\rho^*) + (F - F') \otimes I_A(\rho^*)))$$

$$\leq \max_{0 \in P_I} \text{tr}(P_1((E - E') \otimes I_A(\rho^*))) + \max_{0 \in P_I} \text{tr}(P_2(F - F') \otimes I_A(\rho^*)))$$

$$\leq \max_{\rho_1 \in \mathcal{D}(\mathcal{H} \otimes \mathcal{A}) : \text{tr}(\rho_1) = 1, \text{tr}(Q\rho_1) \geq \lambda} \left( \max_{0 \in P_I} \text{tr}(P_1((E - E') \otimes I_A(\rho_1))) \right)$$

$$+ \max_{\rho_2 \in \mathcal{D}(\mathcal{H} \otimes \mathcal{A}) : \text{tr}(\rho_2) = 1, \text{tr}(Q\rho_2) \geq \lambda} \left( \max_{0 \in P_I} \text{tr}(P_2((F - F') \otimes I_A(\rho_2))) \right)$$

$$= \max_{\rho_1 \in \mathcal{D}(\mathcal{H} \otimes \mathcal{A}) : \text{tr}(\rho_1) = 1, \text{tr}(Q\rho_1) \geq \lambda} T(E \otimes I_A(\rho_1), E' \otimes I_A(\rho_1))$$

$$+ \max_{\rho_2 \in \mathcal{D}(\mathcal{H} \otimes \mathcal{A}) : \text{tr}(\rho_2) = 1, \text{tr}(Q\rho_2) \geq \lambda} T(F \otimes I_A(\rho_2), F' \otimes I_A(\rho_2))$$

$$= \|E - E'\|_{Q, \lambda} + \|F - F'\|_{Q, \lambda}.$$
A.2 Computation Details for the $QBF$ and $QW_6$ Case Studies

First note that an $n$-qubit depolarizing channel can be written in Kraus form as $\frac{1}{2^n}(I \circ I + X \circ X + Y \circ Y + Z \circ Z)^{\otimes n}$. To numerically compute the error $\epsilon_U$ for $QBF$, one first computes the Choi-Jamiołkowski representation of $\Phi_U - U \circ U^\dagger$:

$$J(\Phi_U - U \circ U^\dagger) = \left( \frac{1}{4} \sum_{i,k \in \{0,1\}} (X|i\rangle\langle k| + Y|i\rangle\langle k| + Z|i\rangle\langle k|) \otimes |i\rangle\langle k| \right)^{\otimes 2} - \sum_{i,j,k,l \in \{0,1\}} U|ij\rangle|kl\rangle|U\rangle\langle j|\langle k|.$$  \hspace{1cm} (A.2)

Using the matrix found in (A.3), we numerically compute the diamond norm [da Silva 2015; Watrous 2009] and obtain $\epsilon_U = p_U \|\Phi_U - U \circ U^\dagger\|_\diamond = 1 \times 10^{-5} \times 0.9375$. Thus the error of the entire loop is $2\epsilon_U = 1.875 \times 10^{-5}$.

For $QW_6$, we compute the Choi-Jamiołkowski representation of $\Phi_H - H \circ H^\dagger$:

$$J(\Phi_H - H \circ H^\dagger) = \sum_{i,j \in \{0,1\}} \left( \frac{1}{4} (X|i\rangle\langle j| + Y|i\rangle\langle j| + Z|i\rangle\langle j|) - H(i\rangle\langle j|)H \right) \otimes |i\rangle\langle j|. \hspace{1cm} (A.3)$$

Using the matrix found in (A.4), we compute the diamond norm [da Silva 2015; Watrous 2009] and obtain $\epsilon_H = p_H \|\Phi_H - H \circ H^\dagger\|_\diamond = 5 \times 10^{-5} \times 0.75 = 3.75 \times 10^{-5}$. Thus the error of the entire loop is $30\epsilon_H = 1.125 \times 10^{-3}$.

A.3 Description of Quantum Repetition Codes

Quantum repetition codes are the quantum equivalent of classical repetition codes. In a classical 3-bit repetition code, 0 is encoded as 000 and 1 is encoded as 111. To do error correction, you can take a majority vote of the three bits (e.g. MAJ(0,0,1) = 0, so 001 will be corrected to 000). Alternatively, you can use two parity checks that indicate whether the first and second, and second and third, bits agree. You can then use this information to do correction (e.g. if the first and second bits do not agree, but the second and third bits do, then the first bit must be in error).

In the quantum case, to construct a code that corrects a single bit flip (i.e. $X$ error), we encode the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ as $|\tilde{\psi}\rangle = \alpha|000\rangle + \beta|111\rangle$. We do error correction using the results of syndrome measurements, which are similar to parity checks. The program in Figure 5 shows how to use the bit flip repetition code to protect an application of the identity gate. Note that this is the program $P_2$ from Section 7.3.

On the right-hand side of the program in Figure 5, we have labeled every stage of the computation with its purpose. The first line of the program (Initialization) initializes qubits $q_2, q_3, q_4$, and $q_5$ to the $|0\rangle$ state. Assuming that qubit $q_1$ is initially in state $\alpha|0\rangle + \beta|1\rangle$, the state of system after the first line can be written as $(\alpha|000\rangle + \beta|100\rangle)|00\rangle$. The next two lines (Encoding) apply CNOT gates, which are the quantum equivalent of classical XOR gates, to construct the state $(\alpha|000\rangle + \beta|111\rangle)|00\rangle$. Note that at this point, qubits $q_1, q_2,$ and $q_3$ form a codeword. The next line (Fault-tolerant gate application) applies a fault-tolerant version of the identity gate, which in this case is simply $I^\otimes 3$. The next four lines perform Syndrome detection, and the three lines after that perform Error correction based on the detected syndromes. The next line (Syndrome clean-up) resets the syndrome qubits to the $|0\rangle$ state, and the final two lines (Decoding) return the state $(\alpha|000\rangle + \beta|111\rangle)|00\rangle$ to $(\alpha|000\rangle + \beta|100\rangle)|00\rangle$. The ENCODE notation from Section 7.3 and Section 7.4 corresponds to the initialization and encoding.
\[ q_2 := |0\rangle; q_3 := |0\rangle; q_4 := |0\rangle; q_5 := |0\rangle; \]

\[ q_1, q_2 := \text{CNOT}[q_1, q_2]; \]
\[ q_1, q_3 := \text{CNOT}[q_1, q_3]; \]
\[ q_1, q_2, q_3 := I^{\otimes 3}[q_1, q_2, q_3]; \]
\[ q_1, q_4 := \text{CNOT}[q_1, q_4]; \]
\[ q_2, q_4 := \text{CNOT}[q_2, q_4]; \]
\[ q_2, q_5 := \text{CNOT}[q_2, q_5]; \]
\[ q_3, q_5 := \text{CNOT}[q_3, q_5]; \]
\[ q_4, q_5, q_2 := 11 - \text{TOFFOLI}[q_4, q_5, q_2]; \]
\[ q_4, q_5, q_3 := 01 - \text{TOFFOLI}[q_4, q_5, q_3]; \]
\[ q_4, q_5, q_1 := 10 - \text{TOFFOLI}[q_4, q_5, q_1]; \]
\[ q_4 := |0\rangle; q_5 := |0\rangle; \]
\[ q_1, q_3 := \text{CNOT}[q_1, q_3]; \]
\[ q_1, q_2 := \text{CNOT}[q_1, q_2]; \]

**Initialization**

**Encoding**

**Fault-tolerant gate application**

**Syndrome detection**

**Error correction**

**Syndrome clean-up**

**Decoding**

Fig. 5. Identity operation protected by a quantum repetition code that can correct one bit flip (X) error.

Fig. 6. Identity operation protected by a quantum repetition code that can correct one bit flip (X) error. This is program \( P_2 \) from Section 7.3.

stages. The \textit{CORRECT} notation corresponds to the syndrome detection and error correction stages. The \textit{DECODE} notation corresponds to the syndrome clean-up and decoding stages.

The program from Figure 5 can be represented more succinctly by the quantum circuit shown in Figure 6. Note that in quantum circuits each wire corresponds to a qubit, and gates are applied from left to right. Vertical lines between wires represent multi-qubit gates.

The repetition code that corrects phase flip errors instead of bit flip errors is similar to the code already discussed, so we will not describe it in detail. We only present the relevant circuit (Figure 7). We refer interested readers to Nielsen and Chuang [2000, Section 10.1].
where $X$ with probability

$(A.4)$

The reason that the final channel is not the identity is that the three-qubit bit flip repetition code

Denotes an $i$

$\rho$ of $P_3$ are perfectly distinguishable (consider $\|X|0\rangle - |I\rangle\|_1 = 1$).

Now we consider $\|P_2\| - \|\tilde{P}_2\|$. For program $\tilde{P}_2$, the denotation of the noisy version of $\tilde{q} := \tilde{I}[\tilde{q}]$ is given by

$(A.7)$

where $(A.7)$ follows from the fact that $I$ and $X$ are perfectly distinguishable (consider $\|X|0\rangle - |I\rangle\|_1 = 1$).

$\rho$ is the state remains unchanged, while $\tilde{q}$ the state evolves to $\rho|111\rangle + \beta|000\rangle$. After decoding, $q_2$ and $q_3$ are reset to $|0\rangle$ in either case. This means that

$(A.9)$

$X_i$ denotes an $X$ error on qubit $q_i$. After error correction, this channel becomes

$(A.8)$

The reason that the final channel is not the identity is that the three-qubit bit flip repetition code can only correct one $X$ error. If zero or one $X$ errors occurs, then error correction will perfectly restore the state. However, if more than one $X$ error occurs, error correction will “correct” the state $\rho$ to be $X_1X_2X_3\rho X_1X_2X_3$.

Recall from Appendix A.3 that the state before applying the identity operation is $\alpha|000\rangle + \beta|111\rangle$.

Hence, after correction, with probability $(1 - p)^3 + 3p(1 - p)^2$ the state remains unchanged, while with probability $3p^2(1 - p) + p^3$ the state evolves to $\alpha|111\rangle + \beta|000\rangle$. After decoding, $q_2$ and $q_3$ are reset to $|0\rangle$ in either case. This means that

$(A.10)$

In this section, we show how to compute the values for $\|P_1\| - \|\tilde{P}_1\|$, $\|P_2\| - \|\tilde{P}_2\|$, and $\|P_3\| - \|\tilde{P}_3\|$ presented in Section 7.3. To begin, observe that

$(A.5)$

$(A.6)$

$(A.7)$

$(A.8)$

$(A.9)$

$(A.10)$

$X_i$ denotes an $X$ error on qubit $q_i$. After error correction, this channel becomes

$(1 - p)^3I + \sum_{i=1}^3 p(1 - p)^2 X_i \otimes (1 - p)I + pX_1 \otimes X_1 (1 - p)I + pX_2 \otimes X_2 (1 - p)I + pX_3 \otimes X_3$
It follows that
\[
\|\|P_2\|\|\| - \|\|P_2\|\|\| \circ = \|\|(1 - p)^3 + 3p(1 - p)^2)I \circ I + (3p^2(1 - p) + p^3)X_1 \circ X_1\|\| - I \circ I\|\| \circ \\
= (3p^2(1 - p) + p^3)(\|I \circ I - X \circ X\|) \circ \\
= 3p^2(1 - p) + p^3.
\] (A.13)

This shows that \(\|\|P_1\|\| - \|\|P_1\|\|\| \circ \|\|P_2\|\|\| \circ \|\|P_1\|\|\| \circ \) for \(p < \frac{1}{2}\).

Now we consider \(\|\|P_3\|\| - \|\|P_3\|\|\|\circ\). In program \(P_3\), the fault-tolerant identity operation is preceded and followed by \(H\) gates. Because \(HXH = Z\), the \(X\) errors in the previous equations are transformed into \(Z\) errors. So the denotation of the noisy version of \(\bar{q}\) in program \(\bar{P}_3\) is given by
\[
((1 - p)I \circ I + pZ_1 \circ Z_1) \otimes ((1 - p)I \circ I + pZ_2 \circ Z_2) \otimes ((1 - p)I \circ I + pZ_3 \circ Z_3)
\] (A.14)
\[
= (1 - p)^3I \circ I + \sum_{i=1}^{3} p(1 - p)^2Z_i \circ Z_i
\] (A.15)
\[
+ \sum_{i,j \in \{1,2,3\}, i < j} p^2(1 - p)Z_iZ_j \circ Z_iZ_j + p^3Z_iZ_2Z_3 \circ Z_iZ_2Z_3
\] (A.16)

where \(Z_i\) denotes a \(Z\) error on qubit \(q_i\). After error correction, this channel becomes
\[
((1 - p)^3 + 3p^2(1 - p))I \circ I + (3p^2(1 - p)^2 + p^3)Z_1 \circ Z_1
\]

Similar to the previous analysis, the state \(\alpha|000\rangle + \beta|111\rangle\) will remain unchanged in the presence of an even number of \(Z\) errors, so correction will leave the system in the correct state. However, the state \(\alpha|000\rangle + \beta|111\rangle\) will become \(\alpha|000\rangle - \beta|111\rangle\), which is equivalent to \(Z_1(\alpha|000\rangle + \beta|111\rangle)\), in the presence of an odd number of \(Z\) errors. The bit flip repetition code cannot correct (or detect) this type of error, so it will leave the state as-is.

Therefore, we have that
\[
\|\|P_3\|\| = ((1 - p)^3 + 3p^2(1 - p))I \circ I + (3p^2(1 - p)^2 + p^3)Z_1 \circ Z_1.
\] (A.17)

It follows that
\[
\|\|P_3\|\|\| - \|\|P_3\|\|\| \circ = \|\|(1 - p)^3 + 3p^2(1 - p))I \circ I + (3p^2(1 - p)^2 + p^3)Z_1 \circ Z_1\|\| - I \circ I\|\| \circ \\
= (3p(1 - p) + p^3)(\|I \circ I - Z \circ Z\|) \circ \\
= 3p(1 - p)^2 + p^3.
\] (A.20)

This shows that \(\|\|P_1\|\| - \|\|P_1\|\|\| \circ < \|\|P_3\|\| - \|\|P_3\|\|\) for \(p < \frac{1}{2}\).