Verification Logics for Quantum Programs
WPE-II

Robert Rand
Computer and Information Sciences
University of Pennsylvania
rrand@seas.upenn.edu

Abstract

We survey the landscape of Hoare logics for quantum programs. We review three papers: "Reasoning about imperative quantum programs" by Chadha, Mateus and Sernadas; "A logic for formal verification of quantum programs" by Yoshihiko Kakutani; and "Floyd-hoare logic for quantum programs" by Mingsheng Ying. We compare the mathematic foundations of the logics, their underlying languages and the expressivity of their assertions. We also use the languages to verify the Deutsch-Jozsa Algorithm, and discuss their relative usability in practice.

1 Introduction

Substantial effort has gone into laying the foundations for quantum computing well in advance of the production of scalable quantum computers. This progress is most significant in the areas of quantum complexity and quantum algorithms: Quantum complexity has studied BQP and BQNP, quantum analogues of P and NP, as well as number of more complex classes like QIP (Vazirani, 2002). A number of quantum algorithms have been developed, including the celebrated Shor's Algorithm (Shor, 1994), which efficiently solves the factorization problem, paving the way for the field of post-quantum cryptography (Bernstein et al., 2008).

Considerable contributions have also been made towards the development of quantum programming languages. This began with Peter Knill's *Conventions for Quantum Pseudocode* (1996), which developed the quantum random access machine (QRAM) model for quantum computation, and Selinger's *Towards a Quantum Programming Language* (2004), which gave

semantics for a simple quantum language. Following this, more advanced languages like QML (Altenkirch and Grattage, 2005) and Quipper (Green et al., 2013) were developed for real world quantum programming. With these languages in hand, researchers began to study the formal verification of quantum programs, beginning with the quantum guarded command language (Sanders and Zuliani, 2000) and Quantum Dynamic Logic (Baltag and Smets, 2006).

In this survey we focus on quantum Hoare logics, logics for reasoning about quantum programs in the natural deduction style developed by C.A.R. Hoare (1969). We survey three papers: Chadha, Mateus and Sernadas' *Reasoning about imperative quantum programs* (2006a), Kakutani's *A logic for formal verification of quantum programs* (2009), and Ying's *Floyd–hoare logic for quantum programs* (2011). We look at a number of qualities of the logics focusing on the following four:

- How expressive is the programming language being analyzed?
- What are the predicates of the logic capable of expressing?
- Is the logic mathematically sound and/or complete?
- How usable is the logic for practical verification?

The structure of the paper is as follows: In Section 2 we introduce the basic notions from quantum mechanics and linear algebra necessary to understand the paper, as well as the notations being used. We try to keep the mathematical notation and exposition to the minimum necessary to understand the logical systems presented. In Section 3 we introduce the basic concepts from Hoare logic, and the papers on Hoare logic for probabilistic programs that directly influenced the logics under discussion: Den Hartog and de Vink's *Verifying probabilistic programs using a Hoare like logic* (2002) and Chadha et al.'s *Reasoning about states of probabilistic sequential programs* (2006b).

In Sections 4 through 6 we introduce the three Hoare logics of interest, focusing on the underlying languages, the forms of the assertions and the deductive systems themselves. We then apply the three logics towards verifying the Deutsch–Jozsa algorithm (Deutsch and Jozsa, 1992) in Section 7. In Section 8 we review the logics in light of this comparison, to understand the core differences between these three logics. We end with a summary of our conclusions and a discussion of the further work needed to make quantum verification useful in practice.

2 Preliminaries

2.1 Quantum Computing

We outline the main ideas in quantum computing and linear algebra needed to understand the papers presented[1].

The main subject of interest for the logics we will be presenting is the quantum bit or *qubit*. A qubit may be in one of two states, labeled 0 and 1 with *amplitudes* α and $\beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. The square of the amplitudes here correspond to probabilities. We represent such a qubit with one of the following notations (matrix and *ket* notation):

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \qquad \text{or} \qquad \alpha \left|0\right\rangle + \beta \left|1\right\rangle$$

More precisely, we will be interested in groups or *superpositions* of entangled qubits, which we represent as follows in the case of k qubits:

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{2^k} \end{pmatrix} \qquad \text{or} \qquad \alpha_1 \left|00\ldots00\right\rangle + \alpha_2 \left|00\ldots01\right\rangle + \cdots + \alpha_{2^k} \left|11\ldots11\right\rangle$$

where $\sum_{i=1}^{2^k} |\alpha_i|^2 = 1$

Assuming that our k qubits are independent of one another (called *uncorrelated*), we can also write this superposition as

$$\begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix} \otimes \begin{pmatrix} \alpha_2 \\ \beta_2 \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix}$$

where $\otimes$ is called the *tensor product* (or *Kronecker product*) and is defined as follows:

$$\begin{pmatrix} \alpha_{1,1} & \ldots & \alpha_{1,m} \\ \vdots & \ddots & \vdots \\ \alpha_{n,1} & \ldots & \alpha_{n,m} \end{pmatrix} \otimes B = \begin{pmatrix} \alpha_{1,1}B & \ldots & \alpha_{1,m}B \\ \vdots & \ddots & \vdots \\ \alpha_{n,1}B & \ldots & \alpha_{n,m}B \end{pmatrix}$$

This product is associative and distributes over addition and will be useful throughout this survey. It also has the useful property that $(A \otimes$

[1]The material in this section draws substantially from John Watrous's excellent lecture notes on quantum computing (Watrous, 2006) as well as the developments in the papers studied and their major influences: Selinger (2004); D'Hondt and Panangaden (2006); Chadha et al. (2006a); Kakutani (2009); Ying (2011).

$B)(C \otimes D) = AC \otimes BD$. The tensor product is implicit when multiplying kets, hence $|0\rangle \otimes |0\rangle = |0\rangle\,|0\rangle = |00\rangle$.

Note that we will often represent a complete quantum state by $|\psi\rangle$ even though it doesn't correspond to a single configuration $\left|\{0,1\}^k\right\rangle$. Corresponding to a ket $|\psi\rangle$ there is a *bra* $\langle\psi| = |\psi\rangle^\dagger$, with $\dagger$ representing the conjugate transpose (the transpose with numbers replaced by their complex conjugates) or *adjoint* of a matrix. $\langle\psi|\,|\phi\rangle$ written $\langle\psi|\,\phi\rangle$ is normal matrix multiplication, equal to the inner or dot product in context.

Generally, we modify qubits by multiplying them by *unitary matrices*, matrices where $U^\dagger U = I$ and which therefore preserve the amplitudes summing to one. The following unitary matrices will appear frequently in this survey:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, S = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$$

$$N = \sigma_x = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}, \sigma_y = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}, \sigma_z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$$

where H is called the Hadamard matrix, S the phase matrix, and σ_x, σ_y and σ_z, the Pauli matrices. (N will frequently be used without explanation to flip a qubit or rows of a matrix.) We can also define expanded Hadamard matrices $H_k = \bigotimes_{i=1}^{k} H$.

In the second and third papers under discussion, we will be interested in a more general form for discussing quantum states known as the *density matrix*. We can represent the state $|\psi\rangle$ in density matrix form as $|\psi\rangle\,\langle\psi|$. That is,

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \text{ becomes } \begin{pmatrix}\alpha\\ \beta\end{pmatrix}\begin{pmatrix}\bar{\alpha} & \bar{\beta}\end{pmatrix} = \begin{pmatrix}|\alpha|^2 & \alpha\bar{\beta}\\ \bar{\alpha}\beta & |\beta|^2\end{pmatrix}$$

So far, we've been interested in *pure states*, states that could be represented in either of the two earlier notations. However, this new notation is substantially more general. For example, if the states ψ_i are probabilistically chosen with the corresponding probabilities p_i, we obtain the following *mixed state*:

$$\sum_i p_i\,|\psi_i\rangle\,\langle\psi_i|$$

In general, a density matrix ρ has two important properties:

1. $tr(\rho) = 1$
2. For any vector v of the appropriate length, $v^T \rho v$ is a real number.

The converse is also true: Any matrix satisfying the properties above represents some probabilistic combination of kets. In this survey, we are interested in a broader class of density matrices where the trace may be less than one, used to represent sub-distributions following Selinger (2004).

We represent a unitary operation U applied to a density matrix via $U\rho U^\dagger$. More generally the set of maps Φ that can be applied to density matrices yielding density matrices (even when tensored with the identity matrix) are called *completely positive* and have the following property: For any density matrix ρ,

$$\Phi(\rho) = \sum_i E_i \rho E_i^\dagger$$

for some set of matrices E_i such that $\sum_i E_i^\dagger E_i \leq I$. If $\sum_i E_i^\dagger E_i = I$, the operation preserves the trace of the original density matrix and is called *admissible*. Complete positivity also implies that the map can be represented as a *Hermitian matrix* M, a matrix for which $M = M^\dagger$ and that for any ρ, $0 \leq tr(M\rho) \leq 1$.

Note that measurement and discarding qubits can also be represented as admissible operations on density matrices. For instance, measuring a qubit and forgetting the result can be represented as $\Phi(\rho) = |0\rangle\langle 0|\rho|0\rangle\langle 0| + |1\rangle\langle 1|\rho|1\rangle\langle 1|$ yielding the following (as expected):

$$\Phi \begin{pmatrix} |\alpha|^2 & \alpha\bar{\beta} \\ \bar{\alpha}\beta & |\beta|^2 \end{pmatrix} = \begin{pmatrix} |\alpha|^2 & 0 \\ 0 & |\beta|^2 \end{pmatrix}$$

Admissible operations may also expand or contract the matrix. Consider the following operation for initializing a new qubit to zero from Kakutani's semantics for the QPL programming language: $[\![\texttt{qbit q}]\!](\rho) = |1\rangle\langle 0| \otimes \rho$. This can be rewritten (in the 2×2 case) as:

$$[\![\texttt{qbit q}]\!] \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \alpha & \beta & 0 & 0 \\ \gamma & \delta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

where E and $E^\dagger$ are the matrices in the middle and $E^\dagger E = I$

2.2 Notation

We've tried to use a uniform notation for the diverse systems studied whenever doing so didn't meaningfully impact the interpretation of the language. Hence, we've replaced the e_0, e_1, E_0 and E_1 of Kakutani's paper with their

ket equivalents $|0\rangle, |1\rangle, |1\rangle\langle 0|$ and $|1\rangle\langle 1|$. On the other hand, we've retained both Kakutani's notation $^{\mathbf{q}}|1\rangle\langle 0|$, which involves permuting the context so $\mathbf{q}$ appears first then applying $^{\mathbf{q}}|1\rangle\langle 0| \otimes I$, and Ying's notation $|0\rangle_{\mathbf{q}} \langle 0|$ which is equivalent to $I \otimes |1\rangle\langle 0| \otimes I$ such that $|1\rangle\langle 0|$ lines up with the location of the qubit $\mathbf{q}$ in its density matrix. Note that we use the identity matrix I without specifying its size, which should be assumed to be the necessary size to enable the desired multiplication.

Throughout the paper $\mathbf{b}, \mathbf{n}, \mathbf{q}$ and $\mathbf{qn}$ will be used as variables for booleans and numbers and their quantum analogues, or (in Chadha et al's case) as registers for the given type. $\vec{\mathbf{q}}$ will refer to a sequence of qubits $\mathbf{q}_1, \mathbf{q}_2, \ldots \mathbf{q}_k$. We adopt Kakutani's notation $\vec{\mathbf{q}} *= U$ to represent applying the unitary matrix U to the given qubits. (Note that $\vec{\mathbf{q}} *= U$ is equivalent to $\vec{\mathbf{q}}$ `:=` $U\vec{\mathbf{q}}$ – the matrix always appears on the left.) X and Y will be used to represent predicates on probabilistic states, and $Pr(X)$ to represent the probability of these predicates.

3 Probabilistic Hoare Logics

Hoare logic (Hoare, 1969) (sometimes Floyd-Hoare Logic, after the contributions of Robert Floyd (1967)) is a logical system for reasoning about imperative programs. The atomic propositions of Hoare logic consist of *Hoare triples* of the form $\{P\}\ c\ \{Q\}$, where P and Q are assertions about program states. The triple $\{P\}\ c\ \{Q\}$ says that for any program state σ if P is true of σ and c terminates from σ in the state σ', then Q holds of σ'. We call this type of assertion, where the triple is true for non-terminating programs, a *partial correctness* assertion. Hoare logic can also be modified to ensure *total correctness*, in which the program is guaranteed to terminate, by modifying the rule for `while` loops (Harel, 1979). We show the natural deduction-style rules of classical Hoare logic in Figure 1.

Generalizing Hoare logic to a probabilistic or quantum setting involves, among other considerations, refining the notion of partial correctness. Unlike classical program, probabilistic programs may never terminate, probabilistically terminate (i.e. terminate with some probability between 0 and 1), terminate *almost surely* (with probability one but with non-terminating traces) or deterministically terminate. How the logic treats non-termination determines the kind of Hoare rule that can be applied for `while` statements. Additionally, the meaning of Hoare triples must change in a probabilistic setting, from deterministic assertions about a program state to either probabilistic assertions about a distribution over states or deterministic assertions

$$\text{Skip}\ \frac{}{\{P\}\ \texttt{skip}\ \{P\}} \qquad \frac{}{\{P[z \mapsto e]\}\ z\ \texttt{:=}\ e\ \{P\}}\ \text{Assignment}$$

$$\frac{\{P\}\ c_1\ \{Q\} \qquad \{Q\}\ c_2\ \{R\}}{\{P\}\ c_1;\ c_2\ \{R\}}\ \text{Sequence}$$

$$\frac{\{P \wedge b\}\ c_1\ \{Q\} \qquad \{P \wedge \neg b\}\ c_2\ \{Q\}}{\{P\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{Q\}}\ \text{If}$$

$$\frac{\{P \wedge b\}\ c\ \{P\}}{\{P\}\ \texttt{while}\ b\ \texttt{do}\ c\ \ \{P \wedge \neg b\}}\ \text{While}$$

$$\frac{P' \to P \qquad \{P\}\ c\ \{Q\} \qquad Q \to Q'}{\{P'\}\ c\ \{Q'\}}\ \text{Consequence}$$

Figure 1: The Classical Hoare Logic Rules

that hold of some portion of the possible outcome states.

Lyle Ramshaw first addressed Hoare logics for probabilistic programs in his 1979 PhD thesis (Ramshaw, 1979). The proposed logic reasoned about both distributions and *frequencies*, sub-distributions obtained by conditioning on a given event. Ramshaw's logic was limited in that it could only express a limited set of assertions of the form $Pr(X) = p$ and used a restrictive loop rule that required proving *feasibility* and *closedness* of assertions. It also had little impact on subsequent work: Den Hartog and de Vink (2002) seemed to be unaware of it, and it didn't significantly influence Chadha et al.'s subsequent logic (Chadha et al., 2006b, 2007). These two systems will primarily concern us in this survey as they directly influenced Kakutani (2009) and Chadha et al's (2006a) Hoare logics for quantum programs.

3.1 Den Hartog and De Vink's pH

In 2002, Jerry den Hartog and Eric de Vink's introduced their probabilistic Hoare Logic pH. In the language being analyzed, $\mathcal{L}_{pw}$, commands are transformers between (sub)distributions over states, represented using Θs. $\mathcal{L}_{pw}$ is a modest extension of the simple imperative language of Hoare (Hoare, 1969), with the addition of probabilistic choice between two commands:

$$c ::= \texttt{skip} \mid \mathbf{n}\ \texttt{:=}\ e \mid c\,;c \mid \texttt{if}\ b\ \texttt{then}\ c\ \texttt{else}\ c \mid \texttt{while}\ b\ \texttt{do}\ c\ \mid c\ \oplus_p\ c$$

where $\mathbf{n}$ ranges over arithmetic variables, e over arithmetic expressions, b over boolean expressions and p over the rational open interval $(0, 1)$. $c_1 \oplus_p c_2$

runs command c_1 with probability p, and c_2 with probability $1 - p$.

The $\oplus_p$ operator is overloaded to also combine (sub)distributions. For example $\Theta_1 \ \oplus_p \ \Theta_2 = p\Theta_1 + (1-p)\Theta_2$ combines two subdistributions, scaling the first by p and the second by $1 - p$. The $b?$ operator restricts a (sub)distribution to only the states satisfying b, throwing out the rest of the probability mass.

We can now introduce the deterministic and probabilistic predicates (or assertions) that are used in the Hoare logic pH itself:

$$X, Y ::= b \mid e = e \mid e < e \mid \cdots \mid \neg X \mid X \wedge X \mid \cdots \mid \forall i : X \mid \exists i : X$$
$$P, Q ::= P_r \mid P \wedge P \mid P \vee P \mid \exists j : P \mid \forall j : P \mid p * P \mid P + P \mid P \ \oplus_p \ P \mid b?P$$

where P_r is a proposition over the real numbers which may include $Pr(X)$, that is, the probability of a given predicate being true. $P_1 \ \oplus_p \ P_2$ is once again shorthand for $p * P_1 + (1-p) * P_2$ which is true of Θ whenever $\Theta = p * \Theta_1 + (1-p) * \Theta_2$ such that Θ_1 satisfies P_1 and Θ_2 satisfies P_2. Similarly, $(b?P)(\Theta)$ means that there exists some Θ' such that $b?\Theta' = \Theta$ which in turn satisfies P.

(Note that X can be thought of as a boolean expression, lifted to the status of deterministic proposition. In the original paper, X is written DP and doesn't explicitly include the booleans – however, the logic often puts booleans inside probability terms, as in the While rule. We will use X throughout this presentation for the boolean terms that appear inside probabilities.)

Logic: pH With these probabilistic assertions defined, we can address the Hoare logic pH, summarized in Figure 2.

The Skip, Assign, Seq and Cons rules are all standard Hoare Logic rules. The Toss rule follows directly from the doubly lifted $\oplus$ operator: $c_1 \ \oplus_p \ c_2$ splits the distribution into two subdistributions satisfying $p * Q_1$ and $(1 - p) * Q_2$. The If rule is somewhat more troublesome: It requires us to show that for any Θ_1' that can be split by $b?\Theta_1'$ into Θ_1, that $[\![c_1]\!][\Theta_1']$ satisfies Q_1 (and likewise for Θ_2).

The While rule is a simple generalization of the conventional While rule, using a notion of $\langle b, c\rangle$-closedness for its invariant. The $\langle b, c\rangle$-closedness of P can be interpreted as a requirement that the probability of termination for any state satisfying P is lower bounded by some constant, hence the program terminates *almost surely* (with probability one). If P is an invariant for the loop and P is $\langle b, c\rangle$-closed then we say P is invariant for $\langle b, c\rangle$ and the While rule follows pretty easily.

$$\text{Skip}\ \frac{}{\{P\}\ \texttt{skip}\ \{P\}} \qquad \frac{P' \to P \qquad \{P\}\ c\ \{Q\} \qquad Q \to Q'}{\{P'\}\ c\ \{Q'\}}\ \text{Cons}$$

$$\text{Assign}\ \frac{}{\{P[\mathbf{n} \mapsto e]\}\ \mathbf{n}\ \texttt{:=}\ e\ \{P\}} \qquad \frac{\{P\}\ c_1\ \{Q_1\} \qquad \{P\}\ c_2\ \{Q_2\}}{\{P\}\ c_1\ \oplus_p\ c_2\ \{Q_1\ \oplus_p\ Q_2\}}\ \text{Prob}$$

$$\text{Seq}\ \frac{\{P\}\ c_1\ \{Q\} \qquad \{Q\}\ c_2\ \{R\}}{\{P\}\ c_1;\ c_2\ \{R\}} \qquad \frac{\{b?P\}\ c_1\ \{Q_1\} \qquad \{\neg b?P\}\ c_2\ \{Q_2\}}{\{P\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{Q_1 + Q_2\}}\ \text{If}$$

$$\text{Or}\ \frac{\{P_1\}\ c\ \{Q\} \qquad \{P_2\}\ c\ \{Q\}}{\{P_1 \vee P_2\}\ c\ \{Q\}} \qquad \frac{\{P\}\ c\ \{Q\} \qquad j \text{ not free in } Q}{\{\exists j : P\}\ c\ \{Q\}}\ \text{Exists}$$

$$\text{And}\ \frac{\{P\}\ c\ \{Q_1\} \qquad \{P\}\ c\ \{Q_2\}}{\{P\}\ c\ \{Q_1 \wedge Q_2\}} \qquad \frac{\{P\}\ c\ \{Q\} \qquad j \text{ not free in } P}{\{P\}\ c\ \{\forall j : Q\}}\ \text{Forall}$$

$$\text{Lin}*\ \frac{\{P\}\ c\ \{Q\}}{\{r * P\}\ c\ \{r * Q\}} \qquad \frac{\{P_1\}\ c\ \{Q_1\} \qquad \{P_2\}\ c\ \{Q_2\}}{\{P_1 + P_2\}\ c\ \{Q_1 + Q_2\}}\ \text{Lin}+$$

$$\text{While}\ \frac{P \text{ invariant for } \langle b, c\rangle}{\{P\}\ \texttt{while}\ b\ \texttt{do}\ c\ \ \{P \wedge Pr(b) = 0\}}$$

Figure 2: Den Hartog and De Vink's pH

pH also has to introduce a number of additional rules (Linearity, And, Or, Exists and Forall) for the sake of expressivity. In the absence of the Or rule, for example, we would be unable to prove $\{Pr(X) = 1/2 \vee Pr(Y) = 1/2\}$ `skip` $\oplus_{1/2}$ `skip` $\{Pr(X) = 1/2 \vee Pr(Y) = 1/2\}$. Using the Prob rule we only obtain $\{Pr(X) = 1/2 \vee Pr(Y) = 1/2\}$ `skip` $\oplus_{1/2}$ `skip` $\{1/2 * (Pr(X) = 1/2 \vee Pr(Y) = 1/2) + 1/2 * (Pr(X) = 1/2 \vee Pr(Y) = 1/2)\}$ the post-condition of which doesn't guarantee $Pr(X) = 1/2 \vee Pr(Y) = 1/2$. Instead, we can combine $\{Pr(X) = 1/2\}$ `skip` $\oplus_{1/2}$ `skip` $\{Pr(X) = 1/2\}$ and $\{Pr(Y) = 1/2\}$ `skip` $\oplus_{1/2}$ `skip` $\{Pr(Y) = 1/2\}$ to achieve the desired result.

Soundness and Completeness Den Hartog and de Vink demonstrate that the pH logic is sound in the partial correctness sense, that is, for any derived triple $\{P\}\ c\ \{Q\}$, if P initially holds in Θ and c terminates almost

surely in Θ', then Q holds of Θ'. pH is also complete for the fragment of $\mathcal{L}_{pw}$ that excludes the While rule, when two further restrictions are applied:

1. $Pr(X)$ can only appear in predicates in the form $Pr(X) = r$ for some real number r.

2. $b?P$ cannot appear in any predicate.

The first restriction is shown not to decrease the expressivity of the logic. The same isn't shown for the second condition, and since $b?P$ appears in the form of the If rule, this restriction would seem to confine us to programs without branching.

Completeness isn't shown for the general form of the logic, including While rules. Den Hartog's thesis (den Hartog, 2002) claims to present a completeness proof for the entire pH, but this proof contains flaws acknowledged by the author.

3.2 Chadha, (Cruz-Filipe,) Mateus and Sernadas's EPPL

Chadha, Mateus and Sernadas followed up on Den Hartog's logic pH with their own state assertion logic EPPL and associated Hoare logic, with the aim of producing a complete logic for probabilistic programs at the cost of abandoning iteration (and therefore Turing-completeness). There are actually two versions of this logic: the one set out in *Reasoning About States of Probabilistic Sequential Programs* (Chadha et al., 2006b) and (with Luis Cruz-Felipe) the logic of *Reasoning About Probabilistic Sequential Programs* (Chadha et al., 2007) which achieves the desired completeness result. Here we will focus on first paper, since it forms the basis for Chadha, Mateus and Sernadas' *Reasoning About Imperative Quantum Programs* (Chadha et al., 2006a), and subsequently discuss the differences between the two.

Chadha et al.'s language should look familiar:

$$c ::= \texttt{skip} \mid \mathbf{n} = e \mid \mathbf{b} = b \mid \mathbf{b} \texttt{ := toss}(p) \mid s \,;s \mid \mathbf{b} - \texttt{if } b \texttt{ then } c \texttt{ else } c$$

where the registers $\mathbf{n}$ and $\mathbf{b}$ are restricted to come from some finite set and the arithmetic expressions e are *real numbers from some finite range.* Den Hartog's probabilistic choice is also replaced by a p-biased coin toss – we can recover $c_1 \oplus_p c_2$ via $\mathbf{b}_i \texttt{ := toss}(p) \,;\mathbf{b}_j - \texttt{if } \mathbf{b}_i \texttt{ then } c \texttt{ else } c$ for some fresh registers $\mathbf{b}_i$ and $\mathbf{b}_j$. Note the boolean register attached to the `if` statement: This register is set to true if the `then` branch is taken and otherwise set to false. This provides a somewhat inelegant way of ensuring that the two branches can be reasoned about separately.

$$\text{Skip}\ \frac{}{\{P\}\ \texttt{skip}\ \{P\}} \qquad \frac{\{P\}\ c_1\ \{Q\} \qquad \{Q\}\ c_2\ \{R\}}{\{P\}\ c_1;\ c_2\ \{R\}}\ \text{Seq}$$

$$\text{AsgnA}\ \frac{}{\{P[\mathbf{n} \mapsto e]\}\ \mathbf{n}\ \texttt{:=}\ e\ \{P\}} \qquad \frac{}{\{P[\mathbf{b} \mapsto b]\}\ \mathbf{b}\ \texttt{:=}\ b\ \{P\}}\ \text{AsgnB}$$

$$\frac{P' \to P \qquad \{P\}\ c\ \{Q\} \qquad Q \to Q'}{\{P'\}\ c\ \{Q'\}}\ \text{Cons}$$

$$\frac{}{\{P[Pr(X) \mapsto p * Pr(X[\mathbf{b} \mapsto \texttt{t}]) + (1-p) * Pr(X[\mathbf{b} \mapsto \texttt{f}])]\}\ \mathbf{b}\ \texttt{:=}\ \texttt{toss}(p)\ \{P\}}\ \text{Toss}$$

$$\frac{\{P_1\}\ c_1; \mathbf{b}\ \texttt{:=}\ \texttt{t}\ \{Q_1\} \qquad \{P_2\}\ c_2; \mathbf{b}\ \texttt{:=}\ \texttt{f}\ \{Q_2\}}{\{(P_1/b_0) \wedge (P_2/\neg b_0)\}\ \mathbf{b} - \texttt{if}\ b_0\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{(Q_1/\mathbf{b}) \wedge (Q_2/\neg \mathbf{b})\}}\ \text{If}$$

$$\text{Or}\ \frac{\{P_1\}\ c\ \{Q\} \qquad \{P_2\}\ c\ \{Q\}}{\{P_1 \vee P_2\}\ c\ \{Q\}} \qquad \frac{\{P\}\ c\ \{Q_1\} \qquad \{P\}\ c\ \{Q_2\}}{\{P\}\ c\ \{Q_1 \wedge Q_2\}}\ \text{And}$$

Figure 3: Chadha, Mateus and Sernadas's EPPL Hoare Logic

The assertions of the language also look similar to those of pH with the $b?$ operator removed and a conditional operator added:

$$P, Q ::= P_r \mid P/X \mid \texttt{f} \mid P \to Q$$

where X is again a deterministic predicate (this time without quantification) and P_r is again a proposition over the reals that may contain terms of the form $Pr(X)$. (The paper's propositions also contain terms of the form $\Box X$ – meaning X is true throughout the distribution – but to simplify the presentation we can replace $\Box X$ with $Pr(\neg X) = 0$. The followup paper by the same authors makes this explicit.)

The interesting addition here is P/X which we can read as "P conditioned on X" – removing the measure of the distribution in which X is false.

Logic: EPPL We present the Hoare logic in Figure 3. The toss rule here is somewhat difficult to read and to use in practice, but takes a deliberate weakest precondition form, pushing the expression to the precondition rather than explicitly including it in the postcondition. For example, to derive $\{Pr(\texttt{t}) = 1\}\ \mathbf{b}\ \texttt{:=}\ \texttt{toss}(\frac{2}{3})\ \{Pr(\mathbf{b}) = \frac{2}{3}\}$ we first weaken $Pr(\texttt{t}) = 1$ to $\frac{2}{3} * Pr(\texttt{t}) + \frac{1}{3} * Pr(\texttt{f}) = \frac{2}{3}$ and then apply the Toss rule.

$$\frac{P \text{ contains no probabilities}}{\{P\}\ c\ \{P\}}\ \text{Pr-Free}$$

$$\frac{P \cap (\mathbf{b} = b_0) \qquad \mathbf{b} \notin vars(P) \cup vars(b_0)}{\{P[\mathbf{b} \mapsto b_0]\}\ c\ \{P\}}\ \text{V-Elim}$$

$$\frac{\{P_1\}\ c_1\ \{P(X) = p_1\} \qquad \{P_2\}\ c_2\ \{P(X) = p_2\}}{\{(P_1/b_0) \wedge (P_2/\neg b_0)\}\ \texttt{if}\ b_0\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{Pr(X) = p_1 + p_2\}}\ \text{If}$$

Figure 4: The Revised EPPL Hoare Logic

The If rule simply says that if P_1 is initially true of the scaled subdistribution satisfying b_0 and we know that $\{P_1\}\ c_1; \mathbf{b}\ \texttt{:=}\ \texttt{t}\ \{Q_1\}$ then Q_1 holds of the same subdistribution, with the extra variable $\mathbf{b}$ now taking the role of b_0 (and similarly for P_2, c_2 and Q_2). This dramatically simplifies the reasoning process.

Note that this logic has substantially fewer additional rules added for reasoning: The linearity rules from pH and the Forall and Exist rules don't belong in light of the absence of universal or existential quantification.

Soundness and Completeness The Hoare logic of Figure 3 is shown to be sound via the Exogenous Probabilistic Propositional Logic (EPPL) introduced in the paper. However, completeness is left for a subsequent work (Chadha et al., 2007). Interestingly, that paper is able to remove some of the crutches used in this one, particularly the requirement that `if` statements be tagged with a boolean variable. However, the new logic (Figure 4) deviates in surprisingly ways from the old one.

At first glance, the If rule is only a simplified version of that in the previous paper. However, the presentation is misleading. The syntax P/X from the previous paper has been repurposed: P/X (and the related syntax $\curlyvee_X$ used in both papers) is simply defined as $P[Pr(Y) \mapsto Pr(Y \wedge X)]$, that is adding the truth of X inside every probability term. Now if the part of the distribution in which b_0 is true is sufficient to guarantee $Pr(X) = p_1$ and the other part guarantees $Pr(X) = p_2$ then the outcome of the branching statement is that $Pr(X) = p_1 + p_2$. Unfortunately, this greatly restrict the form of the postcondition. We require two new rules – Pr-Free (which states that any assertion without probabilities and hence variables is preserved by any command) and ElimV (which allows us to eliminate equalities) to combine multiple derivations and regain full expressivity. The new logic

is shown to be complete and decidable by showing that for any c and Q the Hoare logic can derive the weakest precondition P that guarantees Q. Moreover, these weakest preconditions and their deductions in the logic can be computed algorithmically.

3.3 Other Hoare-like Systems

Substantial additional work has been done in the area of Hoare logic for probabilistic programs. In 1996, Morgan (Morgan, 1996) introduced a Hoare `while` rule for probabilistic programs. More recently, Rand and Zdancewic (Rand and Zdancewic, 2015) introduced a verified Hoare logic for probabilistic programs that treats partial termination like non-termination and demonstrates multiple equivalent If rules. The Easycrypt cryptographic tool is built upon both a probabilistic Hoare logic (Barthe et al., 2014) and a probabilistic *relational* Hoare logic (Barthe et al., 2011), inspired by the relational Hoare logic of Benton (Benton, 2004).

In 2004, Vásquez et al. (2004) compared Den Hartog's PHL with Morgan and McIver's PGCL (1999), a probabilistic variant of Dijkstra's Guarded Command Language (1975). There has been substantial recent interest in PGCL and its variants (McIver and Morgan, 2006; Hurd et al., 2005; Cock, 2012; Jansen et al., 2015; Olmedo et al., 2016), including a Quantum Guarded Command Language (Sanders and Zuliani, 2000), but these lie outside the scope of this survey.

4 Chadha, Mateus and Sernadas' EEQPL

Shortly after the publication of the first EPPL paper, the authors wrote an extension to the realm of quantum programs Chadha et al. (2006a). Their quantum programming language features four kinds of data: booleans, natural numbers, qubits and qunits. *Qunits* generalize natural number numbers in the same way qubits generalize bits: Instead of being unit vectors in the two dimensional Hilbert space $\mathcal{H}_2$, a qunit is a unit vector in $\mathcal{H}_N$. N here is 2^k for some fixed k, which serves as a bound on both qunits and natural numbers – the arithmetic of the language is modular arithmetic. The language further assumes that there are a fixed number M of indexed registers ($\mathbf{b}_i, \mathbf{n}_i, \mathbf{q}_i$ and $\mathbf{qn}_i$) for each type of data.

Instead of states, the commands of the programming language are defined over *ensembles*. An ensemble is a discrete sub-probability measure with finite support over classical valuations and quantum pure states. Classical valuations are simply mappings v from registers to values and pure states

are as defined in the preliminaries. Note that these ensembles are sufficient to express mixed states as well.

The commands of the language can be split up into the quantum commands U and the classical commands c, with U being callable from c. The following quantum commands can all be thought of as unitary transformations:

$$\begin{aligned} U ::= {} & I \mid H : \mathbf{q} \mid H : \mathbf{qn} \mid \sigma_x : \mathbf{q} \mid \sigma_x : \mathbf{qn}(e, e) \mid S(e, b) : \mathbf{q} \mid S(e, e) : \mathbf{qn} \mid \\ & UU \mid \texttt{qif}\ \mathbf{q}\ \texttt{then}\ U\ \texttt{else}\ U \mid \texttt{qcase}\ \mathbf{qn}\ \triangleright\ 0 : U, \ldots, n-1 : U \end{aligned}$$

Here H, σ_x and S refer to the Hadamard, Pauli X, and phase shift operators discussed in the preliminaries, where the phase shift takes two arguments. Each of these can be applied to qubits or bits. The `qif` and `qcase` constructs can be represented as controlled versions of their arguments (and hence unitary operations).

The classical commands of the language are as follows:

$$\begin{aligned} c ::= {} & \texttt{skip} \mid \mathbf{b}\ \texttt{:=}\ b \mid \mathbf{n}\ \texttt{:=}\ e \mid U \mid \mathbf{b} \overset{m}{\texttt{:=}} \mathbf{qn} \mid \mathbf{n} \overset{m}{\texttt{:=}} \mathbf{qn} \mid c\ ; c \mid \\ & \texttt{if}\ \mathbf{b}\ \texttt{then}\ c\ \texttt{else}\ c \mid \texttt{case}\ \mathbf{n}\ \triangleright\ 0 : c, \ldots, n-1 : c \mid \mathbf{n}\ \texttt{repeat}\ c \end{aligned}$$

where $\mathbf{b}_i \overset{m}{\texttt{:=}} \mathbf{q}_i$ denotes measuring $\mathbf{q}_i$ and storing the outcome in $\mathbf{b}_i$. Note that, like in the previous papers by the same authors, the language lacks a loop construct so all programs terminate. Moreover, distinct Hoare logic rules are not given for `case` or `repeat` (they are treated as shorthand) so we can effectively exclude them from the language.

The propositions of the logic take a similar form as those in the authors' EPPL, but in this case we'll make the subexpressions explicit. We first have to introduce quantum valuation terms, represented by ω, which assign boolean and natural number values to all of the qubit and qunit registers. The amplitude of these valuations is denoted $\langle \omega | t \rangle$. We can now introduce the components of assertions and assertions themselves:

$$\begin{aligned} r ::= {} & \mathbb{R} \mid \mathbf{n} \mid r + r \mid Re(z) \mid Im(z) \mid Arg(z) \mid |z| \\ z ::= {} & r + ir \mid re^{i}r \mid \langle \omega | t \rangle \mid b \\ X ::= {} & b \mid r \leq r \mid \texttt{f} \mid X \to X \\ E ::= {} & \mathbb{R} \mid E(r)/X \mid E + E \mid E * E \\ P, Q ::= {} & E \mid P/X \mid E \leq E \mid \texttt{f} \mid P \to Q \end{aligned}$$

$Re(z)$ and $Im(z)$ and the real and imaginary components of the complex number z. $Arg(z)$ is the *argument* of the given complex number – the angle

$$\text{Skip}\ \frac{}{\{P\}\ \texttt{skip}\ \{P\}} \qquad \frac{\{P\}\ c_1\ \{Q\} \qquad \{Q\}\ c_2\ \{R\}}{\{P\}\ c_1;\ c_2\ \{R\}}\ \text{Seq}$$

$$\text{AsgnB}\ \frac{}{\{P[\mathbf{b} \mapsto b]\}\ \mathbf{b}\ \texttt{:=}\ b\ \{P\}} \qquad \frac{}{\{P[\langle\omega|t\rangle \mapsto \langle U\omega|t\rangle]\}\ U\ \{P\}}\ \text{Unit}$$

$$\frac{P' \to P \qquad \{P\}\ c\ \{Q\} \qquad Q \to Q'}{\{P'\}\ c\ \{Q'\}}\ \text{Cons}$$

$$\frac{}{\{P[E(r) \mapsto m_1^{\mathbf{b},\mathbf{q}}(E(r)) + m_0^{\mathbf{b},\mathbf{q}}(E(r))\}\ \mathbf{b} \overset{m}{\texttt{:=}} \mathbf{q}\ \{P\}}\ \text{MeasB}$$

$$\frac{\{P_1\}\ c_1; \mathbf{b}\ \texttt{:=}\ \texttt{t}\ \{Q_1\} \qquad \{P_2\}\ c_2; \mathbf{b}\ \texttt{:=}\ \texttt{f}\ \{Q_2\}}{\{(P_1/\mathbf{b}) \wedge (P_2/\neg\mathbf{b})\}\ \texttt{if}\ \mathbf{b}\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{(Q_1/\mathbf{b}) \wedge (Q_2/\neg\mathbf{b})\}}\ \text{If}$$

$$\text{Or}\ \frac{\{P_1\}\ c\ \{Q\} \qquad \{P_2\}\ c\ \{Q\}}{\{P_1 \vee P_2\}\ c\ \{Q\}} \qquad \frac{\{P\}\ c\ \{Q_1\} \qquad \{P\}\ c\ \{Q_2\}}{\{P\}\ c\ \{Q_1 \wedge Q_2\}}\ \text{And}$$

Figure 5: Chadha, Mateus and Sernadas' EEQPL

in radians of z drawn in the complex plane. As in the first paper by these authors, P/X removes the measure in which X is false. Note that in this paper the E terms includes *expectations* over real expressions of the form $E(r)$ (rather than probabilities). We again exclude necessity operators: As the paper acknowledges, they aren't needed.

Logic: EEQPL Most of the Hoare logic rules (Figure 5) should be familiar from their use in EPPL. The two new ones echo classical rules: Unit essentially substitutes the unitary transformation in the precondition just like assignment does. MeasB echoes Toss with some added complications. To begin with, for $m_1^{\mathbf{b}_i,\mathbf{q}_j}$ it scales all of the expectation terms by the probability of $\mathbf{q}_j$ evaluating to 1, obtained by summing over the satisfying amplitudes. This is the same as in Toss. However, measuring also modifies the state so $m_1^{\mathbf{b}_i,\mathbf{q}_j}$ removes all valuations where the j^{th} qubit is assigned to 0 and scales the remaining ones. Finally, it replaces all instances of $\mathbf{b}_i$ with `t` to reflect the assignment. As elsewhere, this is all put into the precondition.

As in the paper's presentation, the rules for numerical expressions and qunits have been elided – they are all slightly more complex versions of the rules presented above.

This logic is shown to be sound (though not complete) in the paper. It also presents a proof of the Deutsch Algorithm in the Hoare logic. We will extend this proof to the Deutsch-Jozsa Algorithm in Section 7 and further discuss EEQPL in the Section 8.

5 Yoshihiko Kakutani's QHL

Kakutani's Hoare logic, QHL, has the desirable property of being based on an existing quantum programming language, Peter Selinger's QPL (Selinger, 2004). QPL is a simple programming language with a well defined denotational semantics, both via a flow chart language defined in that paper and directly through interpreting typing judgments as Scott-continuous functions between density matrices. Noticeably, QPL is a *functional* language, where all commands are functions and no global state exists.

QHL deals with a restricted version of QPL without recursive procedure calls but with measurement and while loops. Kakutani spells out the implicit denotational semantics of QPL as functions between matrices. The matrices take the place of the global state of classical Hoare logic or the distribution over states in probabilistic Hoare logic and can be reasoned about in a similar fashion. In fact, QHL draws heavily upon den Hartog and de Vink's pH, as we shall see.

The subset of QPL used in the paper is as follows:

$$c ::= \texttt{skip} \mid c\,;c \mid \texttt{bit}\ \mathbf{b} \mid \texttt{qbit}\ \mathbf{q} \mid \texttt{discard}\ \mathbf{q} \mid \mathbf{b}\ \texttt{:=}\ 0 \mid \mathbf{b}\ \texttt{:=}\ 1 \mid \vec{\mathbf{q}} *= U \mid$$
$$\texttt{if}\ \mathbf{b}\ \texttt{then}\ c\ \texttt{else}\ c \mid \texttt{while}\ \mathbf{b}\ \texttt{do}\ c \mid \texttt{measure}\ \mathbf{q}\ \texttt{then}\ c\ \texttt{else}\ c$$

In this case, we can think of **b** and **q** as simply variables (rather than registers) referring to a single bit or qubit. Note that **b** and **q** do not come from separate namespaces: The type system guarantees that the variables that appear in the If and While terms are bits and those that appear in Measure are qubits. In fact, bits and qubits even share a representation in QPL's matrices, with the restriction that only certain operations apply to each guaranteed by the type checker.

We won't show the complete type system, but it takes the expected form: `bit` **b** and `qbit` **q** add a bit and qubit, respectively, to the typing context and `discard` **q** removes a bit/qubit. If/While and Measure require the guard **b** to be of type bit and qubit respectively and end with the same context as the commands they run - preserving **b** in the While case.

The denotational semantics, on the other hand, are worth spelling out in full. These semantics are defined on derivations of typing judgments, rather

$$[\![\langle \Gamma\rangle \texttt{ skip } \langle \Gamma\rangle]\!](\rho) = \rho$$

$$[\![\langle \Gamma\rangle\ c_1;\ c_2\ \langle \Gamma''\rangle]\!](\rho) = [\![\langle \Gamma'\rangle\ c_1;\ c_2\ \langle \Gamma''\rangle]\!]([\![\langle \Gamma\rangle\ c_1\ \langle \Gamma'\rangle]\!](\rho))$$

$$[\![\langle \Gamma\rangle \texttt{ bit } \mathbf{b} \langle \mathbf{b} : \texttt{bit }, \Gamma\rangle]\!](\rho) = |1\rangle\langle 0| \otimes \rho$$

$$[\![\langle \Gamma\rangle \texttt{ qbit } \mathbf{q} \langle \mathbf{q} : \texttt{qbit }, \Gamma\rangle]\!](\rho) = |1\rangle\langle 0| \otimes \rho$$

$$[\![\langle \mathbf{b} : \mathrm{T}, \Gamma\rangle \texttt{ discard } \mathbf{b} \langle \Gamma\rangle]\!](\rho) = (\langle 0| \otimes I)\rho(|0\rangle \otimes I) + (\langle 1| \otimes I)\rho(|1\rangle \otimes I)$$

$$[\![\langle \mathbf{b} : \texttt{bit }, \Gamma\rangle\ \mathbf{b} \texttt{ := } 0\ \langle \mathbf{b} : \texttt{bit }, \Gamma\rangle]\!](\rho) = \pi_0(\rho) + \nu(\pi_1(\rho))$$

$$[\![\langle \mathbf{b} : \texttt{bit }, \Gamma\rangle\ \mathbf{b} \texttt{ := } 1\ \langle \mathbf{b} : \texttt{bit }, \Gamma\rangle]\!](\rho) = \nu(\pi_0(\rho)) + \pi_1(\rho)$$

$$[\![\langle \vec{\mathbf{b}} : \vec{\texttt{bit}} , \Gamma\rangle\ \vec{\mathbf{b}} *= U\ \langle \vec{\mathbf{b}} : \vec{\texttt{bit}} , \Gamma\rangle]\!](\rho) = (U \otimes I)\rho(U^\dagger \otimes I)$$

$$[\![\langle \mathbf{b} : \texttt{bit }, \Gamma\rangle \texttt{ if } \mathbf{b} \texttt{ then } c_1 \texttt{ else } c_2\ \langle \Gamma'\rangle]\!](\rho)$$
$$= [\![\langle \mathbf{b} : \texttt{bit }, \Gamma\rangle\ c_1\ \langle \Gamma'\rangle]\!](\pi_0(\rho)) + [\![\langle \mathbf{b} : \texttt{bit }, \Gamma\rangle\ c_2\ \langle \Gamma'\rangle]\!](\pi_1(\rho))$$

$$[\![\langle \mathbf{b} : \texttt{qbit }, \Gamma\rangle \texttt{ measure } \mathbf{b} \texttt{ then } c_1 \texttt{ else } c_2\ \langle \Gamma'\rangle]\!](\rho)$$
$$= [\![\langle \mathbf{b} : \texttt{qbit }, \Gamma\rangle\ c_1\ \langle \Gamma'\rangle]\!](\pi_0(\rho)) + [\![\langle \mathbf{b} : \texttt{qbit }, \Gamma\rangle\ c_2\ \langle \Gamma'\rangle]\!](\pi_1(\rho))$$

$$[\![\langle \mathbf{b} : \texttt{bit }, \Gamma\rangle \texttt{ while } \mathbf{b} \texttt{ do } c\ \ \langle \mathbf{b} : \texttt{bit }, \Gamma\rangle]\!](\rho)$$
$$= \sum_{n=0}^{\infty} \pi_0 \left[\left([\![\langle \mathbf{b} : \texttt{bit }, \Gamma\rangle\ c_1\ \langle \Gamma'\rangle]\!] \circ \pi_1 \right)^n (\rho) \right]$$

where

$$\pi_0(\rho) = (|0\rangle\langle 0| \otimes I)\rho(|0\rangle\langle 0| \otimes I)$$
$$\pi_1(\rho) = (|1\rangle\langle 1| \otimes I)\rho(|1\rangle\langle 1| \otimes I)$$
$$\nu(\rho) = (N \otimes I)\rho(N \otimes I)$$

Figure 6: QPL Semantics

than commands alone.

The commands `bit` **b** and `qbit` **q** add a bit or qubit to the density matrix in the initial state 0 and $|0\rangle$ respectively. Discard requires the bit/qubit to be discarded to be first in the context, and hence in in the first position in the matrix, it then shrinks down the matrix to remove the bit/qubit. Assigning a bit to 0 adds together the half of the matrix in which the bit was zero with the flipped half in which it was 1. (Note that **b** doesn't need to have a deterministic value, as it may be probabilistically in each state based on the outcome of a measurement.) Unitary transformation has the expected result when the acted upon qubits are ordered first. If and Measure are identical for our purposes, performing the relevant operations on

the two projected matrices, and While can be interpreted as an infinite sum of matrices (with decreasing traces in the terminating case).

We can now introduce the assertions of the language:

$$r := \mathbb{R} \mid x \mid Pr(X) \mid f(r, \ldots, r)$$
$$P, Q ::= r \leq r \mid int(r) \mid r * P \mid P + P \mid {}^{\mathbf{b},\ldots,\mathbf{b}}MP \mid \neg P \mid P \wedge P \mid \forall x : P$$

where X stands for an arbitrary predicate that potentially includes quantified over variables x. Similarly, f here is an arbitrary function on real numbers, and M is a $2^n \times 2^n$ matrix where n is its number of arguments. $int(r)$ here is a predicate stating that r is an integer.

Note the substantial similarity to Den Hartog and De Vink's logic of Section 3.1. The addition and multiplication notation are borrowed from pH (though Kakutani employs $\oplus$ in place of $+$ in his presentation) and say that the distribution can be split into parts satisfying the two predicates, or that when scaled to the given size they satisfy the predicate.

But what do these assertions say? To look at a simple case of interest, we say that a triple of typing context, matrix and valuation (Γ, ρ, v) satisfies $Pr(\mathbf{q}_j = x) = 1/2$ if $\sum\{U^\dagger \rho U|, U = e_{\mathbf{q}_1} \otimes \cdots \otimes e_{\mathbf{q}_n} \ \& \ \mathbf{q}_j = v(x)\} = 1/2$, where the $\mathbf{q}$'s are ordered by the context Γ. (Note that the vs are included simply to deal with quantification). That is, the statement is true whenever the total density of the states satisfying $\mathbf{q}_j = 1$ equals $1/2$.

We say that a matrix (Γ, ρ, v) satisfies ${}^{\mathbf{b}_1,\ldots,\mathbf{b}_n}MP$ if (Γ', ρ', v) satisfies P where $\Gamma \cong \mathbf{b}_1 : T_1 \ldots \mathbf{b}_n : T_n, \Gamma'$ and $\rho = (M \otimes I)\rho'(M^\dagger \otimes I)$. Note that the $\mathbf{b}_1, \ldots \mathbf{b}_n$ serves to reorder the bits/qubits of Γ so only those are multiplied by M in the desired order.

Logic: QHL We present the complete rule set of QHL, including the purely logical rules, in Figure 7. Here again, QHL adheres closely to the formula of pH and largely avoids the conventions of Chadha et al.

Many of the rules here are standard, and the logical rules are preserved from pH. The New Bit and New QBit rules are straightforward, though they do require that $Pr(\mathtt{t}) = 1$ which may not always be true. (Like pH, this logic deals with subdistributions, following a method for reasoning about quantum programs where probabilities do not sum to one described in Selinger's QPL paper.) This is generally obtainable via the linearity rules.

The Unit rule is among the few that uses a weakest precondition form, though the author notes that the rule $\vdash \{P\}\ \vec{\mathbf{q}} *= U\ \{{}^{\vec{\mathbf{q}}}U^\dagger P\}$ would be equivalent. This is notably the only type of assignment that doesn't break

$$\text{Skip}\ \frac{}{\{P\}\ \texttt{skip}\ \{P\}} \qquad \frac{\{P\}\ c_1\ \{Q\} \qquad \{Q\}\ c_2\ \{R\}}{\{P\}\ c_1;\ c_2\ \{R\}}\ \text{Seq}$$

$$\frac{}{\{P \wedge Pr(\mathtt{t}) = 1\}\ \texttt{bit}\ \mathbf{b}\ \{P \wedge Pr(\mathbf{b} = 0) = 1\}}\ \text{New-b}$$

$$\frac{}{\{P \wedge Pr(\mathtt{t}) = 1\}\ \texttt{qbit}\ \mathbf{q}\ \{P \wedge Pr(\mathbf{b} = 0) = 1\}}\ \text{New-q}$$

$$\text{Asgn0}\ \frac{}{\{P\}\ \mathbf{b}\ \texttt{:=}\ 0\ \{{}^{\mathbf{b}}|1\rangle\langle 0|P + {}^{\mathbf{b}}|1\rangle\langle 1|P\}} \qquad \frac{}{\{P\}\ \mathbf{b}\ \texttt{:=}\ 1\ \{{}^{\mathbf{b}}|1\rangle\langle 0|P + {}^{\mathbf{b}}|1\rangle\langle 1|P\}}\ \text{Asgn1}$$

$$\text{Discard}\ \frac{\mathbf{b} \notin vars(P)}{\{P\}\ \texttt{discard}\ \mathbf{b}\ \{P\}} \qquad \frac{}{\{{}^{\vec{\mathbf{q}}}U^\dagger P\}\ \vec{\mathbf{q}} *= U\ \{P\}}\ \text{Unit}$$

$$\frac{\{{}^{\mathbf{b}}|1\rangle\langle 0|P\}\ c_1\ \{Q_1\} \qquad \{{}^{\mathbf{b}}|1\rangle\langle 1|P\}\ c_2\ \{Q_2\}}{\{P\}\ \texttt{if}\ \mathbf{b}\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{Q_1 + Q_2\}}\ \text{If}$$

$$\frac{\{{}^{\mathbf{b}}|1\rangle\langle 1|P_n\}\ c\ \{P_{n+1}\}\ \text{for}\ n \in \mathbb{N} \qquad \{{}^{\mathbf{b}}|1\rangle\langle 0|P_n \mid n \in \mathbb{N}\} \models Q}{\{P\}\ \texttt{while}\ \mathbf{b}\ \texttt{do}\ c\ \ \{Q\}}\ \text{While}$$

$$\frac{\{{}^{\mathbf{q}}|1\rangle\langle 0|P\}\ c_1\ \{Q_1\} \qquad \{{}^{\mathbf{q}}|1\rangle\langle 1|P\}\ c_2\ \{Q_2\}}{\{P\}\ \texttt{measure}\ \mathbf{q}\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{Q_1 + Q_2\}}\ \text{Measure}$$

$$\text{Subst}\ \frac{\{P\}\ c\ \{Q\}}{\{P[x \mapsto r]\ c\ \{Q[x \mapsto r]\}} \qquad \frac{P' \to P \qquad \{P\}\ c\ \{Q\} \qquad Q \to Q'}{\{P'\}\ c\ \{Q'\}}\ \text{Cons}$$

$$\text{Or}\ \frac{\{P_1\}\ c\ \{Q\} \qquad \{P_2\}\ c\ \{Q\}}{\{P_1 \vee P_2\}\ c\ \{Q\}} \qquad \frac{\{P\}\ c\ \{Q_1\} \qquad \{P\}\ c\ \{Q_2\}}{\{P\}\ c\ \{Q_1 \wedge Q_2\}}\ \text{And}$$

$$\text{Exists}\ \frac{\{P\}\ c\ \{Q\} \qquad x \notin fv(Q)}{\{\exists x : P\}\ c\ \{Q\}} \qquad \frac{\{P\}\ c\ \{Q\} \qquad x \notin fv(P)}{\{P\}\ c\ \{\forall x : Q\}}\ \text{Forall}$$

$$\text{Lin} *\ \frac{\{P\}\ c\ \{Q\}}{\{r * P\}\ c\ \{r * Q\}} \qquad \frac{\{P_1\}\ c\ \{Q_1\} \qquad \{P_2\}\ c\ \{Q_2\}}{\{P_1 + P_2\}\ c\ \{Q_1 + Q_2\}}\ \text{Lin} +$$

Figure 7: Kakutani's QHL

up the predicates, as it applies a simple unitary transformation to the entire matrix (even if only the mentioned bits are effected).

By contrast, the assignment rules do split the predicates into two parts, simply in order to flip the bit in one case while leaving it unchanged in the other. This is in contrast to Chadha et al's approach, where they could simply remap **b** in the precondition – but this was possible only since that logic separates the classical and quantum bits, where here they are treated together.

The If and Measure rules can be treated together as they're given identical semantics by QPL. As in pH, if P is sufficient to guarantee Q_1 in the `then` branch and Q_2 in the `else` branch, the entire `if` statement has the outcome $Q_1 + Q_2$. QHL avoids needing to use the $c?$ construct since matrix multiplication by E_i suffices to scale down the trace of the matrix (or equivalently, the probabilities of each branch).

Finally we have the While rule. The While rule given is a bit of a departure from previous rules, as it doesn't reason about an invariant. Instead, it states that if the sum of the matrices satisfying the postconditions P_i of each loop satisfies some predicate Q, then the Q holds after the loop terminates. Obviously, this isn't a very useful rule for reasoning about loops: This sum might be expensive or impossible to calculate. The paper offers in addition an alternative, invariant-based While rule, subject to three conditions:

1. The invariant P has no negation, disjunction or existentials
2. The program always terminates
3. The guard is independent of all other variables.

In this case, we have a While rule that resembles den Hartog and de Vink's:

$$\frac{\{P \wedge Pr(b = 1) = 1\}\ c\ \{P\}}{\{P \wedge Pr(\mathtt{t}) = 1\}\ \mathtt{while}\ b\ \mathtt{do}\ c\ \ \{P \wedge Pr(b = 0) = 1\}}$$

Though it makes no claim to completeness – and it almost certainly isn't, being based upon pH and general avoiding weakest precondition based rules – the logic of QHL is sound and and used to verify a surprising number of programs. The author uses QHL to verify a quantum teleportation algorithm, Shor's Algorithm, the Deutsch Algorithm, a case of the Deutsch-Jozsa Algorithm and the Quantum Coin Tossing Protocol.

QHL was also employed to analyze quantum cryptography protocols in a subsequent paper. *A formal approach to unconditional security proofs for quantum key distribution* (Kubota et al., 2011) verifies the security properties of the classic BB84 quantum security protocol (Bennett and Brassard,

1984). It converts BB84 into an entanglement distillation protocol written in an extended QPL and then transforms Shor and Preskill's (2000) proof of its security into a formal QHL deduction. The proof itself is surprisingly concise (taking up one page of Kubota et al.) demonstrating the strength of QHL for formal verification in the area of quantum cryptography.

6 Mingsheng Ying's qPD

In 2011, Ying (2011) proposed a complete quantum Hoare logic. This logic relies heavily on two previous works: Peter Selinger's QPL paper (Selinger, 2004), discussed above, and D'Hondt and Panangaden's formulation of quantum predicates and weakest preconditions (D'Hondt and Panangaden, 2006).

Ying's language assumes that all variables are quantum. As in Kakutani's presentation, this doesn't mean that there is only one type of data. In fact there can be an arbitrary number of datatypes, all generalized to the quantum context. In practice, the paper deals with two types: the quantum booleans (or qubits) and the quantum integers (which we will call qunits by analogy with Chadha's qunit types). The Hilbert spaces corresponding to these types are $\mathcal{H}_2$ and $\mathcal{H}_\infty$, or the space of sequences whose squares sum to one. Note that the basis vectors of each space are precisely the booleans and the integers.

The syntax of the language being analyzed is quite simple:

$$c ::= \texttt{skip} \mid c\,;c \mid \mathbf{q} \texttt{ := } 0 \mid \vec{\mathbf{q}} *= U \mid \texttt{measure } M[\vec{\mathbf{q}}] \texttt{ : } \vec{c} \mid \texttt{while } M[\vec{\mathbf{q}}] \texttt{ do } c$$

Here Measure and While are both quantum measurements, which can operate on either qubits or qunits, similar to Chadha's presentation. M, then, is a measurement with k outcomes and $\vec{c}$ is the k commands associated with those outcomes. In the While loop there are only two measurement outcomes and c is executed for one of them.

In contrast to QPL, this is very much an imperative language with an associated small step operational semantics. The "state" ρ in contexts is a partial density operator over the state space of all the quantum variables $\mathcal{H}_{all} = \bigotimes_{\mathbf{q}} \mathcal{H}_q$. Interestingly, the operational semantics is nondeterministic: Wherever measurement can lead to multiple different states there is a transition for each state, and the probability of the given state being achieved is encoded in the trace of ρ.

We present the operational semantics in Figure 8. We slightly modify Ying's presentation, replacing the "empty command" E with `skip`, removing

$$\vdash \langle \mathbf{q} := 0, \rho\rangle \to \langle \texttt{skip}, \rho_0^{\mathbf{q}}\rangle \qquad \text{(Init)}$$

$$\vdash \langle \vec{\mathbf{q}} *= U, \rho\rangle \to \langle \texttt{skip}, U\rho U^\dagger\rangle \qquad \text{(Unit)}$$

$$\langle c_1, \rho\rangle \to \langle c_1', \rho'\rangle \vdash \langle c_1;\ c_2, \rho\rangle \to \langle c_1'\ ; c_2, \rho'\rangle \qquad \text{(Seq}_1\text{)}$$

$$\vdash \langle \texttt{skip}\ ; c_2, \rho\rangle \to \langle c_2, \rho\rangle \qquad \text{(Seq}_2\text{)}$$

$$M_m \in M \vdash \langle \texttt{measure}\ M[\vec{\mathbf{q}}]\ :\ \vec{c}, \rho\rangle \to \langle c_m, M_m\rho M_m^\dagger\rangle \qquad \text{(Meas)}$$

$$\vdash \langle \texttt{while}\ M[\vec{\mathbf{q}}]\ \texttt{do}\ \vec{c}, \rho\rangle \to \langle \texttt{skip}, M_0\rho M_0^\dagger\rangle \qquad \text{(Loop}_0\text{)}$$

$$\vdash \langle \texttt{while}\ M[\vec{\mathbf{q}}]\ \texttt{do}\ \vec{c}, \rho\rangle \to \langle c\ ; \texttt{while}\ M[\vec{\mathbf{q}}]\ \texttt{do}\ \vec{c}, M_1\rho M_1^\dagger\rangle \qquad \text{(Loop}_1\text{)}$$

Figure 8: The Operational Semantics of Ying's Quantum Programs

the rule that takes $\langle \texttt{skip}, \rho\rangle$ to $\langle E, \rho\rangle$, and making the Seq$_2$ rule explicit, rather than implicit as in that presentation.

$\rho_0^{\mathbf{q}}$ is shorthand for $|0\rangle_q \langle 0| \rho |0\rangle_q \langle 0| + |0\rangle_q \langle 1| \rho |1\rangle_q \langle 0|$ in the qubit case and

$$\sum_{n=-\infty}^{\infty} |0\rangle_q \langle n| \rho |n\rangle_q \langle 0|$$

in the qunit case. The notation $|0\rangle_q$ indicates the state of the qubit $\mathbf{q}$, treated separately from the state of the remaining qubits. This generalizes the QPL assignment semantics above, except that instead of $|1\rangle\langle 0| \otimes I$ we may have $I_1 \otimes |1\rangle\langle 0| \otimes I_2$, since the updated qubit may not be in the first position.

As noted above, the operational semantics is nondeterministic whenever a `measure` or `while` command is encountered. More than that, every `while` command leads to some nonterminating program consisting of applying Loop$_1$ repeatedly. Here the paper makes a misleading claim:

> If [a sequence of transitions] is finite and its last configuration is $\langle \texttt{skip}, \rho'\rangle$, then we say that it terminates in ρ'; and if it is infinite, then we say that it diverges. We say that c can diverge from ρ whenever it has a diverging computation starting in ρ.

But we just observed that every `while` program has a diverging computation! In fact, a program can be said to converge in the given programming language whenever the sum of the traces of the terminating configurations is one. It's interesting to note that this blurs the distinction between terminating programs and *almost surely terminating* programs discussed in Section 3 – both terminate in the same way. (We can distinguish deterministically terminating programs by specifying a "finite sum" in the above

definition.) For the purpose of the Hoare logic, this is immaterial since it doesn't use a notion of divergence, and instead reasons in weakest liberal precondition style.

The denotational semantics of a program is defined as the following function between partial density operators (or states). Note that the we are summing over a multiset since multiple paths may terminate in identical states:

$$[\![c]\!](\rho) = \sum \{\rho' : \langle c, \rho \rightarrow^* \langle \texttt{skip}, \rho' \rangle\}$$

In Proposition 5.1, Ying is able to more succinctly characterize the denotational semantics, in particular for the Measure and While command. However, this definition is sufficient for our purposes.

The assertions of the logic, following D'Hondt and Panangaden (2006), consist of operators P on the Hilbert space H of the quantum variables such that $\forall \rho \in H, tr(P\rho) \in [0,1]$ (these corresponds to the completely positive maps discussed in the preliminaries). We can use these to define two types of Hoare triples: Total correctness triples and partial correctness triples.

$$\begin{aligned} &\models_{tot} \{P\}\ c\ \{Q\} && \text{iff } \forall \rho,\ tr(P\rho) \leq tr(Q[\![c]\!](\rho) \\ &\models_{par} \{P\}\ c\ \{Q\} && \text{iff } \forall \rho,\ tr(P\rho) \leq tr(Q[\![c]\!](\rho)) + tr(\rho) - tr([\![c]\!]\rho) \end{aligned}$$

Note that these two definitions coincide exactly when $tr(\rho) = tr([\![c]\!]\rho)$, justifying our notion of termination above.

These forms of assertions are closer to the *expectations* of Kozen's foundational works on verifying probabilistic programs (Kozen, 1981, 1985), rather than the truth-functional propositions about probabilities that explicitly motivate Chadha et al. (2006b, 2007). D'Hondt and Panangaden (2006) refer to these as *quantum expectation values* while the positive operators are called *observables*. In essence, the triple $\{P\}\ c\ \{Q\}$ says that the probability of terminating satisfying Q (plus the probability of nontermination in the partial correctness case) is at least as great as the probability of satisfying P.

These two notions of correctness correctness relate closely to the notion of *weakest preconditions* (wp) and *weakest liberal precondition* (wlp), referred to more precisely as *weakest pre-expectation* and *weakest liberal pre-expectation* by Katoen et al. (2015) and Jansen et al. (2015) who discuss the various possible weakest pre-expectations in depth. Informally, the weakest precondition of P and c is the weakest constraint sufficient to ensure P upon running c. Formally, $wp.c.P$ for a command c and observable P is the largest predicate such that $\models_{tot} \{wp.c.P\}\ c\ \{P\}$ (and similarly for wlp using partial correctness). "Largest" is given by the *Löwner partial order* whereby

$$\text{Skip}\ \frac{}{\{P\}\ \texttt{skip}\ \{P\}} \qquad \frac{}{\{\sum_{n\in\{0,1\}} |n\rangle_q \langle 0| P |0\rangle_q \langle n|\}\ \mathbf{q} := 0\ \{P\}}\ \text{AsgnB}$$

$$\text{Unit}\ \frac{}{\{U^\dagger P U\}\ \vec{\mathbf{q}} *= U\ \{P\}} \qquad \frac{}{\{\sum_{n=-\infty}^{\infty} |n\rangle_q \langle 0| P |0\rangle_q \langle n|\}\ \mathbf{qn} := 0\ \{P\}}\ \text{AsgnN}$$

$$\text{Seq}\ \frac{\{P\}\ c_1\ \{Q\} \qquad \{Q\}\ c_2\ \{R\}}{\{P\}\ c_1;\ c_2\ \{R\}} \qquad \frac{P' \sqsubseteq P \qquad \{P\}\ c\ \{Q\} \qquad Q \sqsubseteq Q'}{\{P'\}\ c\ \{Q'\}}\ \text{Cons}$$

$$\frac{\forall m,\ \{P_m\}\ c_m\ \{Q\}}{\{\sum_m M_m^\dagger P_m M_m\}\ \texttt{measure}\ M[\mathbf{q}]\ :\ \vec{c}\ \{Q\}}\ \text{Measure}$$

$$\frac{\{Q\}\ c\ \{M_0^\dagger P M_0 + M_1^\dagger Q M_1\}}{\{M_0^\dagger P M_0 + M_1^\dagger Q M_1\}\ \texttt{while}\ M[\vec{\mathbf{q}}]\ \texttt{do}\ c\ \{P\}}\ \text{While}$$

Figure 9: Ying's partial correctness logic qPD

$P \sqsubset Q$ if for every state ρ, $tr(P\rho) < tr(Q\rho)$. Proposition 7.1 in Ying's paper establishes the weakest preconditions for every command; these are central in proving the completeness of the Hoare logic.

We present the rules of Ying's partial correctness Hoare logic qPD in Figure 9.

The Cons rule generalizes the standard consequence rule to our setting using the *Löwner partial order* defined above.

The assignment rules use a weakest precondition form, specifying the necessary form of the precondition P. Note that unlike Kakutani's presentation, this is direct: Instead of discussing assertions that bear a given relation to a modified state, our representation of assertions as matrices allows us to modify them directly.

The Measure rule used here is unique, most closely resembling the If rule from Chadha's original probabilistic logic (Chadha et al., 2007). Where the guard is a simple qubit, it *is* an If rule, which says that if P_1 and P_2 both guarantee Q following their respective commands, then the scaled sum of P_1 and P_2 is sufficient to guarantee Q after measurement. Measuring a group of qubits or qunits simply generalizes this branching construct.

The While rule is particularly elegant. It says that if you can split the precondition into two parts, one of which, when scaled, is sufficient to

preserve the precondition upon running c, then the remaining part (scaled) will be preserved at the "end" of the loop. Note that this is a partial correctness rule in the weakest liberal precondition sense, so any probability that fails to terminate is counted as satisfying the postcondition.

The paper also offers a total correctness version of its While rule (which together with the other rules from Figure 9 forms a total correctness logic). For this we need a notion of (P, ϵ)-boundedness, where ϵ bounds the trace of the diverging computation. We can then say that if for any $\epsilon > 0$ there is a $(M_1^\dagger Q M_1, \epsilon)$-bound function of the loop starting in Q then the loop terminates.

Soundness and Completeness The partial correctness logic introduced is both sound and complete, meaning that any partial correctness assertion of the form $\{P\}\ c\ \{Q\}$ is valid if and only if it is derivable in qPD. The soundness proofs are all given directly and tend to follow from simple linear algebra. Consider the derivation of the AsgnB rule.

From the denotational semantics of the language (see Figure 8 and the simple translation from operational to denotational semantics above), we have that

$$[\![q := 0]\!] = |0\rangle_q \langle 0| \rho |0\rangle_q \langle 0| + |0\rangle_q \langle 1| \rho |1\rangle_q \langle 0|$$

Hence, we can do the following simple deduction (modified from the paper's example for AsgnN):

$$\begin{aligned} tr\left[\left(\sum_{n\in\{0,1\}} |0\rangle_q \langle n| P |n\rangle_q \langle 0|\right) \rho\right] &= tr(|0\rangle_q \langle 0| P |0\rangle_q \langle 0| \rho + |0\rangle_q \langle 1| P |1\rangle_q \langle 0| \rho) \\ &= tr(P(|0\rangle_q \langle 0| \rho |0\rangle_q \langle 0| + |0\rangle_q \langle 1| \rho |1\rangle_q \langle 0|)) \\ &= tr(P[\![q := 0]\!](\rho)) \end{aligned}$$

The proofs for Measure and While are naturally somewhat more involved (the proof for Unit is pretty much immediate), but they follow similar principles.

The proof of completeness says that every valid formula $\{P\}\ c\ \{Q\}$ is derivable in qPD. This follows from a proof that the preconditions of the logic are weakest preconditions, which draws on D'Hondt and Panangaden (2006).

Both soundness and completeness are shown for the total correctness variant (qTD) as well.

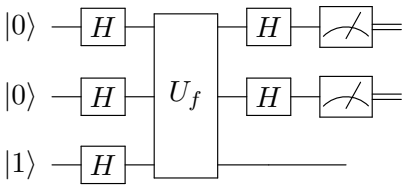

Figure 10: The Deutch-Jozsa Algorithm on $k = 2$ qubits

7 Case Study: The Deutsch-Jozsa Algorithm

We can compare the three logics in terms of their usefulness for verifying quantum programs. A useful case study is using the algorithms to verify the *Deutsch-Jozsa Algorithm* (Deutsch and Jozsa, 1992). Kakutani conveniently provides a QHL derivation of one case of Deutsch-Jozsa for us, we will verify another. Chadha et al. verify a more basic version of the algorithm, the Deutsch Algorithm (Deutsch, 1985), which we expand into our proof of the general algorithm. The proof of Deutsch-Jozsa in Ying's qPD is our own, drawing upon his example of Grover's Algorithm.

The Deutsch-Jozsa problem is quite simple. We have an function f (implemented by an oracle) which takes a number in the range 0 to 2^n, represented in binary, to either zero or one. We also have the following guarantee: Either the function is identical on all inputs, or 0 on exactly 2^{k-1}. Return "constant" in the first case, eitherwise "balanced".

The best possible classical solution to this problem is obvious: Check the first $2^{k-1} + 1$ numbers, if they're all the same return "constant" else return "balanced". Even if we terminate upon seeing distinct numbers, in the worst case this takes $2^{k-1} + 1$ steps and is therefore an exponential algorithm.

In order to express this problem in quantum computing terms, we need to modify it slightly. O can't take arbitrary numbers in superposition with each other to 0 or 1, as this might not represent a unitary transformation. Instead, we use the function

$$U_f(|x\rangle\,|b\rangle) = |x\rangle\,|b \oplus f(x)\rangle$$

where x is the number in binary, b is an extra qubit and $\oplus$ is the standard xor operator. This U_f will always be unitary.

The solution using a quantum computer is quite simple (written out in

quantum pseudocode):

$$
\begin{aligned}
&\vec{\mathbf{q}} \texttt{ := } \left|0^k\right\rangle; \\
&\mathbf{q}_e \texttt{ := } |1\rangle; \\
&\vec{\mathbf{q}} \otimes \mathbf{q}_e \mathrel{*}= H_{k+1}; \\
&\vec{\mathbf{q}} \otimes \mathbf{q}_e \mathrel{*}= U_f; \\
&\vec{\mathbf{q}} \mathrel{*}= H_k; \\
&\texttt{discard } \mathbf{q}_e; \\
&\texttt{measure } \vec{\mathbf{q}}
\end{aligned}
$$

This consists of simply initializing the qubits to $0 \ldots 01$, applying three matrix transformations and measuring the first k qubits. If they all measure 0 then f is constant otherwise it's balanced. See Figure 10 for the corresponding quantum circuit.

A mathematical proof Since our languages don't support vectors or arrays, we'll verify a simplified version of the algorithm, where $f : \{0,1\}^2 \to \{0,1\}$. We first present the mathematical proof of correctness. For readability, we use I_x as shorthand for $(-1)^{f(x)}$:

$$
\begin{aligned}
U_f H_3 |001\rangle &= U_f \frac{1}{2\sqrt{2}} \Big(|000\rangle - |001\rangle + |010\rangle - |011\rangle \\
&\qquad + |100\rangle - |101\rangle + |110\rangle - |111\rangle \Big) \\
&= \frac{1}{2\sqrt{2}} \Big(|00\rangle |f(00)\rangle - |00\rangle |1 - f(00)\rangle + |01\rangle |f(01)\rangle - |01\rangle |1 - f(01)\rangle \\
&\qquad + |10\rangle |f(10)\rangle - |10\rangle |1 - f(10)\rangle + |11\rangle |f(11)\rangle - |11\rangle |1 - f(11)\rangle \Big) \\
&= \frac{1}{2\sqrt{2}} \Big(I_{00} |00\rangle (|0\rangle - |1\rangle) + I_{01} |01\rangle (|0\rangle - |1\rangle) \\
&\qquad + I_{10} |10\rangle (|0\rangle - |1\rangle) + I_{10} |10\rangle (|0\rangle - |1\rangle) \Big) \\
&= \frac{1}{2} \Big(I_{00} |00\rangle + I_{01} |01\rangle + I_{10} |10\rangle + I_{10} |10\rangle \Big) \frac{1}{\sqrt{2}} \Big(|0\rangle - |1\rangle \Big)
\end{aligned}
$$

We can now discard that last qubit and apply H_2:

$$H_2 \frac{1}{2}\Big(I_{00}\left|00\right\rangle + I_{01}\left|01\right\rangle + I_{10}\left|10\right\rangle + I_{10}\left|10\right\rangle \Big)$$
$$= \frac{1}{2} * \frac{1}{2}\Big((I_{00} + I_{01} + I_{10} + I_{11})\left|00\right\rangle + (I_{00} - I_{01} + I_{10} - I_{11})\left|01\right\rangle$$
$$+ (I_{00} + I_{01} - I_{10} - I_{11})\left|10\right\rangle + (I_{00} - I_{01} - I_{10} + I_{11})\left|11\right\rangle \Big)$$

If $\forall x,\ f(x) = k$ for some constant $k \in \{0, 1\}$ we get

$$\frac{1}{4}(4I_k\left|00\right\rangle + 0 + 0 + 0)$$

meaning the probability of measuring $|00\rangle$ is 1.

On the other hand, if $f(x)$ is zero for half the permutations, the coefficients of $|00\rangle$ add up to zero, so the probability of measuring that state is 0. (It's worth noting that this neatly divides the 8 valid functions into four identifiable groups: for instance, $f(b_1b_2) = b_1$ and $f(b_1b_2) = 1 - b_1$ both guarantee measuring $|11\rangle$).

EEQPL Let's try expressing this program in Chadha's language. This language doesn't allow the initial allocation of registers or assignment of qubits, so we'll assume that q_1, q_2 and q_e have the desired form. We omit the discard statement, since the language doesn't allow for discarding qubits and we can simply ignore q_b rather than discard it. Finally, we will introduce the gate U_f that acts on 3 qubits, and use $H_3 : (\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e)$ as shorthand for $H : \mathbf{q}_1;\ H : \mathbf{q}_2;\ H : \mathbf{q}_e$, and likewise for H_2:

$$H_3 : (\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e);$$
$$U_f : (\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e);$$
$$H_2 : (\mathbf{q}_1, \mathbf{q}_2);$$
$$\mathbf{b}_1 \overset{m}{:=} \mathbf{q}_1;$$
$$\mathbf{b}_2 \overset{m}{:=} \mathbf{q}_2;$$

We will verify the case where $\forall x, f(x) = 1$. Using $\square X$ as shorthand for $E(X) = 1$, we want to show that given the precondition $\square(\langle 001|\, t\rangle = 1)$ we can derive the postcondition $\square(\mathbf{b}_1 = 0 \wedge \mathbf{b}_2 = 0)$.

$$
\begin{aligned}
&\{\Box(\langle 001|\, t\rangle = 1)\}\\
&H_3 : (\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e);\\
&\{\Box(\frac{1}{4}(|\, \langle 001|\, t\rangle + \langle 101|\, t\rangle + \langle 011|\, t\rangle + \langle 111|\, t\rangle|^2\\
&+ |\, \langle 000|\, t\rangle + \langle 100|\, t\rangle + \langle 010|\, t\rangle + \langle 010|\, t\rangle|^2 = 1)\}\\
&U_f : (\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e);\\
&\{\Box(\frac{1}{4}(|\, \langle 000|\, t\rangle + \langle 100|\, t\rangle + \langle 010|\, t\rangle + \langle 110|\, t\rangle|^2\\
&+ |\, \langle 001|\, t\rangle + \langle 101|\, t\rangle + \langle 011|\, t\rangle + \langle 011|\, t\rangle|^2 = 1)\}\\
&H : \mathbf{q}_1;\\
&\{\Box(\frac{1}{2}(|\, \langle 000|\, t\rangle + \langle 010|\, t\rangle|^2 + |\, \langle 001|\, t\rangle + \langle 011|\, t\rangle|^2 = 1)\}\\
&H : \mathbf{q}_2;\\
&\{\Box(|\, \langle 000|\, t\rangle|^2 + |\, \langle 001|\, t\rangle|^2 = 1)\} \rightarrow\\
&\{E(p_0^{\mathbf{q}_1} p_0^{\mathbf{q}_2}/(0 = 0) + E(p_1^{\mathbf{q}_1} p_0^{\mathbf{q}_2}/(1 = 0)) = 1\}\\
&\mathbf{b}_1 \overset{m}{:=} \mathbf{q}_1;\\
&\{E(p_0^{\mathbf{q}_2}/(b_1 = 0)) = 1\} \rightarrow\\
&\{E(p_0^{\mathbf{q}_2}/(b_1 = 0 \wedge 0 = 0)) + E(p_1^{\mathbf{q}_2}/(b_1 = 0 \wedge 1 = 0)) = 1\}\\
&\mathbf{b}_2 \overset{m}{:=} \mathbf{q}_2;\\
&\{E(\mathtt{t}/(\mathbf{b}_1 = 0 \wedge \mathbf{b}_2 = 0)) = 1\} \rightarrow\\
&\{\Box(\mathbf{b}_1 = 0 \wedge \mathbf{b}_2 = 0)\}
\end{aligned}
$$

Due to insufficient space to explicitly derive the first deduction of the proof, we note that applying a H_3 to $|001\rangle$ results in the magnitudes of $xx1$ states being $\frac{1}{2\sqrt{2}}$ and the magnitude of $xx0$ states being $-\frac{1}{2\sqrt{2}}$. This is sufficient to guarantee the statement in line 2: $\frac{1}{4}((\frac{4}{2\sqrt{2}})^2 + (\frac{-4}{2\sqrt{2}})^2) = \frac{1}{4}(2 + 2) = 1$.

We also note that two measurements towards the end were substantially simplified since the measurement's outcome was deterministic. In the more general case, we would have to scale by the probability of each outcome.

QHL We now proceed to Kakutani's logic. We can write the Deutsch-Sojza algorithm in QPL in its complete form:

$\texttt{qbit } \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e;$
$\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_b \texttt{ := } 0, 0, 1;$
$\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e \mathrel{*}= H_3;$
$\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e \mathrel{*}= U_f;$
$\mathbf{q}_1, \mathbf{q}_2 \mathrel{*}= H_2;$
$\texttt{discard } \mathbf{q}_e;$
$\texttt{bit } \mathbf{b}_1, \mathbf{b}_2;$
$\texttt{measure } \mathbf{q}_1 \texttt{ then } \mathbf{b}_1 \texttt{ := } 1 \texttt{ else } \mathbf{b}_1 \texttt{ := } 0;$
$\texttt{measure } \mathbf{q}_2 \texttt{ then } \mathbf{b}_2 \texttt{ := } 1 \texttt{ else } \mathbf{b}_2 \texttt{ := } 0$

And we can proceed to verify it, mostly following Kakutani's own verification sketch.

$\{Pr(\mathtt{t}) = 1\}$
$\texttt{qbit } \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e;$
$\{Pr(\mathbf{q}_1 = 0 \wedge \mathbf{q}_2 = 0 \wedge \mathbf{q}_e = 0) = 1\}$
$\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e \texttt{ := } 0, 0, 1;$
$\{Pr(\mathbf{q}_1 = 0 \wedge \mathbf{q}_2 = 0 \wedge \mathbf{q}_e = 1) = 1\}$
$\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e \mathrel{*}= H_3;$
$\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e \mathrel{*}= U_f;$
$\mathbf{q}_1, \mathbf{q}_2 \mathrel{*}= H_2;$
$\{^{\mathbf{q}_1,\mathbf{q}_2} H_2 {}^{\mathbf{q}_1,\mathbf{q}_2,\mathbf{q}_e} U_f H_3 Pr(\mathbf{q}_1 = 0 \wedge \mathbf{q}_2 = 0 \wedge \mathbf{q}_e = 1) = 1\} \rightarrow$
$\{Pr(\mathbf{q}_1 = 0 \wedge \mathbf{q}_2 = 0) = 1\}$
$\texttt{discard } \mathbf{q}_e;$
$\{Pr(\mathbf{q}_1 = 0 \wedge \mathbf{q}_2 = 0) = 1\}$
$\texttt{bit } \mathbf{b}_1, \mathbf{b}_2;$
$\{Pr(\mathbf{q}_1 = 0 \wedge \mathbf{q}_2 = 0 \wedge \mathbf{b}_1 = 0 \wedge \mathbf{b}_2 = 0) = 1\}$
$\texttt{measure } \mathbf{q}_1 \texttt{ then } \mathbf{b}_1 \texttt{ := } 1 \texttt{ else } \mathbf{b}_1 \texttt{ := } 0;$
$\texttt{measure } \mathbf{q}_2 \texttt{ then } \mathbf{b}_2 \texttt{ := } 1 \texttt{ else } \mathbf{b}_2 \texttt{ := } 0$
$\{Pr(\mathbf{q}_1 = 0 \wedge \mathbf{q}_2 = 0 \wedge \mathbf{b}_1 = 0 \wedge \mathbf{b}_2 = 0) = 1\}$

Note that the consequent step follows from the mathematical deduction early in this section, that applying the three given matrices to $|001\rangle$ in the case where $f(x) = 1$ throughout yields a states where the first and second qubits are guaranteed to be zero. Essentially, this moves the crucial reasoning steps into the consequence rule of the logic. The Measure steps also become trivial when the guard is deterministic since if $\rho \models \Phi$ where $\rho = \mathbf{q} \otimes \rho'$ then $\rho \models {}^{\mathbf{q}}(|1\rangle\langle 1|)\Phi$.

qPD Since the language of qPD doesn't allow for setting classical bits, we will leave out the measurement step and prove that the final quantum state is of the form $\alpha\,|000\rangle + \beta\,|001\rangle$. This is equivalent to saying that given the precondition I_8 (the identity matrix which multiplied by any density matrix yields a trace of 1) we result in the postcondition:

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

meaning that all of the weight is concentrated in the 2x2 square in the top left of the density matrix.

The program, then has the following simple form (note that we take two steps to set $\mathbf{q}_e$ to 1):

$$\begin{aligned} &\mathbf{q}_1 := 0 \\ &\mathbf{q}_2 := 0 \\ &\mathbf{q}_e := 0 \\ &\mathbf{q}_e *= N \\ &\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e *= H_3; \\ &\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e *= U_f; \\ &\mathbf{q}_1, \mathbf{q}_2 *= H_2 \end{aligned}$$

For the sake of the proof, we will need the matrix form of U_f. In the case where $\forall x, f(x) = 1$, $U_f(< a, b, c, d, e, f, g, h >) =< b, a, d, c, f, e, h, g >$ so $U_f = I_4 \otimes N$.

We can now show the proof of the Deutsch-Jozsa algorithm:

$$
\begin{array}{l}
\{I_8\} \rightarrow \\
\{|0\rangle_1 \langle 0| \, |0\rangle_2 \langle 0| \, T \, |0\rangle_2 \langle 0| \, |0\rangle_1 + \dots\} \\
\mathbf{q}_1 := 0; \\
\{|0\rangle_2 \langle 0| \, T \, |0\rangle_2 \langle 0| + |1\rangle_2 \langle 0| \, T \, |0\rangle_2 \langle 1|\} \\
\mathbf{q}_2 := 0; \\
\{T\} \rightarrow \{|0\rangle_e \langle 0| \, T \, |0\rangle_e \langle 0| + |1\rangle_e \langle 0| \, T \, |0\rangle_e \langle 1|\} \\
\mathbf{q}_e := 0; \\
\{T\} \rightarrow \{(I_4 \otimes N)^\dagger T (I_4 \otimes N)\} \\
\mathbf{q}_e *= N; \\
\{T\} \rightarrow \{H_3^\dagger (I_4 \otimes N)^\dagger H_2^\dagger T H_2 (I_4 \otimes N) H_3\} \\
\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e *= H_3; \\
\{(I_4 \otimes N)^\dagger H_2^\dagger T H_2 (I_4 \otimes N)\} \\
\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_e *= U_f; \\
\{(H_2 \otimes I_2)^\dagger T (H_2 \otimes I_2)\} \\
\mathbf{q}_1, \mathbf{q}_2 *= H_2 \\
\{T\}
\end{array}
$$

Note that the uses of the consequence rule are all directly from the (matrix) equality of the two assertions. Some of the intermediate matrices we've elided are actually quite elegant, for example $[(H_2 \otimes I_2)^\dagger T (H_2 \otimes I_2)]_{ab}$ is $1/4$ wherever $a + b$ is even, and zero elsewhere.

8 Hoare Logics Compared

We can now compare the Hoare logics in detail. As noted in the introduction, we are interested in the following properties:

- Language expressivity
- Assertion expressivity
- Completeness
- Usefulness

Some of these blur into one another: The usefulness of a logic relates directly to the expressivity of its language and its assertions. Likewise, language features (like Chadha et al.s iteration construct) that don't have associated Hoare logic rules don't interest us. Nevertheless, we look at the four categories, referencing the limitations of the language and assertions where necessary.

Languages The language of (Chadha et al., 2006a) is the most limited of those analyzed. The limitation to a finite set of registers and even to a maximum size for natural numbers and qunits can be dealt with – such a maximum exists for many practical programs. On the other hand, the absence of a While loop or any form of recursion is highly limiting. The If statement is also restricted such that the guard cannot be modified in either of the branches. Additionally, it would be helpful to be able to initialize qubits rather than assuming that some number of qubits already exist in a given form.

The measure constructor also takes an odd form: Measuring a single qubit and storing its value in **b** recalls the **b** `:=` `toss`(p) construct in Chadha et al. (2006b) in place of Den Hartogs $c_1 \oplus_p c_2$. However, in contrast to that paper, where toss has an elegant rule associated with it, the measurement rule here is remarkably complex. We will discuss this further on; the choice of measure operator doesn't ultimately impact the expressivity of the language.

One of the nice features of EQPL's target language is its distinction between classical operations and quantum operations, with a sub-language for unitary transformations on matrices. This neatly reflects the popular QRAM model for quantum computation (Knill, 1996), in which quantum computation is run by a separate machine and the classical machine may interact with the measured output. It is the only language studied which maintains a distinct classical state, the traditional object of Hoare logic verification.

Kakutani's paper uses a fragment of QPL, a small but expressive quantum programming language. However, it also strips that language of its procedure calls. Since QPL doesn't have any sort of stack, this strictly costs expressivity. QPL is also missing natural numbers or integers and their quantum analogues, though we may be able to add these at a low cost.

The treatment of bits is convenient: Bits occupy the same density matrix that qubits occupy, rendering the entire quantum program a transformation on density matrices. Bits and qubits can also be introduced, named, assigned (in the bit case) and discarded – operations sorely missed in the

other languages under investigation.

Ying's language is essentially a smaller subset of QPL with the addition of quantum integers. It drops the ability to allocate and discard qubits meaning that all of its programs should be treated as an $N \times N$ square matrix, where N is two to the power of the number of qubits referenced in the program. It also drops the bit/qubit distinction - the language features only qubits, though some of those may be treated as bits through limiting their use to certain contexts. Finally the measure rule is slightly more general, as it allows us to specify a set of outcomes and their associated subprograms. Note that these outcomes must be disjoint and cover all possibilities for the given measurement.

Assertions Chadha et al.'s EEQPL is unique in that its language manipulates classical and quantum variables and hence its logic must deal with ensembles of classical and quantum states. In response, the logic reasons probabilistically about classical states and pure quantum states, represented as kets. These type of assertions can be bulky and often difficult to manipulate. Like its predecessor logic EPPL, EEQPL puts relies heavily upon scaling assertions from reasoning about sub-distributions to reasoning about a complete distribution.

The assertion logic for is also quite restricted. It only allows reasoning about real number equalities, which may contain expectation terms. The absence of any form of quantification limits what we can express. However, the paper implies that the assertion language can be substantially expanded, which is often the case.

By contrast Kakutani's QHL allows for arbitrary expressions inside its probability construct, and takes specific care to allow for quantification. It even adds a valuation function to the interpretation of assertion satisfaction to deal with open variables that appear in both a precondition and postcondition.

To an even greater extent then Chadha et al., QHL reasons about quantum systems through the prism of probability theory. Instead of describing a density matrix, the logic may say that the probability that $\mathbf{q}_3$ returns 0 upon measurement is equal to $\frac{1}{2}$. This describes a large set of pure and mixed quantum states, many of which can be distinguished from one another via unitary transformation. QHL does include a construct that mentions matrices: MP, where M is a matrix and P a proposition, describes a state that is equal to a unitary transformation M applied to a state satisfying P. However, P itself is still a probabilistic expression that doesn't refer to a

quantum system directly.

Ying's assertions in qPD are a substantial departure from those of Chadha and Kakutani, as they take the form of completely positive matrices P such that for any density matrix ρ the trace of $P\rho$ is in the unit interval. These don't fully characterize a density matrices but, as D'Hondt and Panangaden (2006) argue, they shouldn't be able to: It's possible to have two distinct mixed states that are physically indistinguishable from each other, these should be treated identically by the logic.

The assertion $\{P\}\ c\ \{Q\}$ is interpreted to mean that the probability of terminating satisfying Q (that is the trace of $Q[\![c]\!]\rho$) is at least as great as the probability of P in ρ, for any quantum state ρ of the appropriate dimensions. In the partial correctness case, we modify that to the probability of satisfying Q plus the probability of non-termination. These types of assertions are substantially different than the assertions in EEQPL and QHL: Chadha et al. (2006b) refers to these as *arithmetical* assertions to be contrasted with *truth functional* assertions. Truth functional assertions tend to be more expressive: It's easy to represent an arbitrary arithmetic triple $\{P\}\ c\ \{Q\}$ in a truth functional manner as $\forall p, \{Pr(P) = p\}\ c\ \{Pr(Q) \geq p\}$ but it's difficult to express arbitrary truth functional assertions as arithmetic ones. Moreover, the specific form of the Hoare triples demands that in a multistage proof, our predicates are monotonically non-decreasing, which is a considerable limitation.

Completeness of the Logics Though throughout this paper we've used EEQPL to refer to both the Hoare logic of Chadha et al. as well as the underlying state logic. In discussing completeness, it's important to make the distinction between the two. The state logic EEQPL derives from two systems: the Exogenous Probabilistic Propositional Logic (EPPL) of Chadha et al. (2006b) (expanded upon in Chadha et al. (2007)), and the Exogenous Quantum Propositional Logic (EQPL) of Mateus and Sernadas (2006). The language of EPPL is explicitly restricted to deal with real numbers drawn from some finite range, this is necessary for the proof of completeness. EPPL is shown to be complete in Chadha et al. (2006b), this is extended to the completeness of the Hoare logic with respect to EPPL in Chadha et al. (2007). On the other hand, EQPL only has a form of completeness called *bounded weak completeness*: EQPL can derive a formula only if it ranges over a finite set of qubit symbols and quantum formulae. The authors show that the EEQPL logic is weakly complete when we restrict the real and complex values to a finite set. They then claim that the Hoare logic based

on EEQPL is complete under the assumption of a finite number of possible quantum and classical valuations, but this claim isn't substantiated in detail.

In contrast to Chadha et al and Ying, Kakutani's QHL paper makes no completeness claims. While it discards the problematic $c?$ construct of the Den Hartog's logic pH, the rules for while loops would presumably cause difficulty for any attempt to prove completeness - in particular, the deterministic while rule doesn't help in this regard. It's equally clear from the logic and its presentation that the logic was designed to be usable rather than complete, a claim we'll evaluate shortly.

Ying's logic qPD, and its total correctness counterpart qTD, are shown to be complete relative to their partial/total correctness semantics. That is, any valid partial correctness triple $\{P\}\ c\ \{Q\}$ can be derived in qPD, and similarly for total correctness triples and qTD. The author qualifies this statement by noting that qPD's completeness is only relative to the theory of complex numbers, since the consequence rule references the the Löwner partial order, but this is to be expected when verifying quantum programs.

$$\wr x, y, z \int$$

Applying the logics EEQPL is a difficult logic to use in practice. Since the language lacks a loop construct, the most difficult part of the program to reason about is measurement, and measurement proves very difficult to tackle. Like EPPL's toss rule, the MeasB rule pushes a lot of complexity into the precondition, where we have to replace every expectation term with the sum of two terms, representing the two possible measurements. Unlike EPPL's toss, this isn't enough. Measurement has three side effects: The boolean register $\mathbf{b}$ is set to $\mathtt{t}$, the qubit $\mathbf{q}$ is set to $|0\rangle$ or $|1\rangle$ and the pure state has to be renormalized to add up to one. All of these effects are pushed into precondition, leading to very complex assertions that have none of the elegance of the Hermitian matrices of qPD. The If rule rests again upon EPPL's combination of scaling and annotating each branch with the probability of the guard – which we then need to know to use the rule at all.

EEQPL's Hoare logic does, however, have the advantage of a weakest precondition form which, while not sufficient to guarantee completeness, does allow reasoning to proceed straightforwardly from the conclusion back.

Kakutani's QHL lacks this feature. Most of the rules seem to be designed for forward reasoning, with the notable exception of the rule for unitary

application. (An alternative, forward-reasoning, version of this rule is also given in the paper.) The lack of directed reasoning makes proving program properties difficult.

Based on Den Hartog and De Vink's pH, QHL also suffers from it's non-constructive form. The frequent form MP holds of ρ whenever some matrix ρ' satisfies P and $\rho = (M \otimes I)\rho'(M^\dagger \otimes I)$. Worse, we have the additive form $P + Q$ which holds of ρ only if ρ can somehow be split into two matrices satisfying P and Q respectively.

Finally, we have two versions of the While rule. Both have their problems: The first involves taking a potentially infinite sum rather than proving an invariant. The second does involve an invariant, but is tremendously limited: It requires guaranteed termination and a guard that is independent of all other program variables, beyond its restriction on the types of postconditions it can prove.

The difficulty in using QHL is somewhat surprising when we consider that the paper presenting it has no fewer than four examples of the logic's applications and a subsequent paper (Kubota et al., 2011) uses it to verify quantum cryptography protocols. However, if we look closely at these derivations, they resemble proof sketches more closely than they resemble actual proofs – most of the details are elided. Worse, the relevant details to proving program correctness aren't contained in the derivation but rather take place through the consequence rules, which often entail reasoning about the entire quantum program.

Finally, we have Ying's qPD logic. This logic benefits substantially from the simplicity of the underlying language and the assertion language. The assertion language, after all, consists purely of matrices satisfying a few equational properties. This allows for a weakest-precondition based logic which then leads to a logic that can be largely automated, starting from the desired conclusion. Surprisingly, even the measurement rule is directed in this manner.

The only hiccup in attempting to automate qPD proofs is the While rule. Using the while rule, we have to divide the Hermetian matrix into two such matrices, one that acts as the invariant and the other as the termination condition. However, even this seems directed: Assuming that the postcondition is derivable, we can find $M_0^\dagger P M_0$ directly, and thereby deduce $M_1^\dagger Q M_1$. Any remaining difficulty lies in expressing desired program conditions through the use of bounded positive operators. (Unfortunately, this interesting aspect of the work receives little attention in the paper itself.)

9 Conclusion and Future Work

Of the three logics studied, Ying's qPD demonstrates both the strongest mathematical grounding and (in our minds) the most potential for further work. As argued by D'Hondt and Panangaden (2006), Baltag and Smets (2006) and others, the language of probability theory alone is insufficient for the rigorous verification of quantum algorithms. This manifests itself in the difficulty that EEQPL and QHL have in precisely characterizing quantum mixed states, and in handling the effects of measurement.

However, qPD also rests upon the simplest language of the logics presented and, in its current state, struggles to verify interesting quantum programs. Ying's proof of Grover's Algorithm requires five pages of dense exposition; the proof of Deutsch-Jozsa above proved easy only because we assumed the presence of a given set of qubits (being unable to introduce them) and elided the measurement step (since it isn't useful in the absence of classical bits).

By contrast, QHL provides easy derivations of a number of quantum proofs in Kakutani (2009) and Kubota et al. (2011), even if these derivations are less informative than we might desire.

In this light, there are three direction in which qPD might be improved. The first involves the language: What additional language constructs can we add to qPD while providing sound deductive rules and without losing completeness? The second involves the assertions: Are there general principles for formulating assertions as completely positive maps? And the last concerns automation: Can we automate proofs of correctness in qPD? It seems from the paper's results that this should be possible. But actually implementing automation in practice would show the enduring value of this work.

References

T. Altenkirch and J. Grattage. A functional quantum programming language. In *Logic in Computer Science, 2005. LICS 2005. Proceedings. 20th Annual IEEE Symposium on*, pages 249–258. IEEE, 2005.

A. Baltag and S. Smets. Lqp: the dynamic logic of quantum information. *Mathematical structures in computer science*, 16(03):491–525, 2006.

G. Barthe, B. Grégoire, S. Heraud, and S. Z. Béguelin. Computer-aided

security proofs for the working cryptographer. In *Advances in Cryptology–CRYPTO 2011*, pages 71–90. Springer, 2011.

G. Barthe, F. Dupressoir, B. Grégoire, C. Kunz, B. Schmidt, and P.-Y. Strub. Easycrypt: A tutorial. In *Foundations of Security Analysis and Design VII*, pages 146–166. Springer, 2014.

C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *IEEE International Conference on Computers, Systems and Signal Processing*, volume 175, page 8, 1984.

N. Benton. Simple relational correctness proofs for static analyses and program transformations. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '04, pages 14–25, New York, NY, USA, 2004. ACM. ISBN 1-58113-729-X. doi: 10.1145/964001.964003.

D. J. Bernstein, J. Buchmann, and E. Dahmen. *Post Quantum Cryptography*. Springer, 2008. ISBN 3540887016, 9783540887010.

R. Chadha, P. Mateus, and A. Sernadas. Reasoning about imperative quantum programs. *Electronic Notes in Theoretical Computer Science*, 158: 19–39, 2006a.

R. Chadha, P. Mateus, and A. Sernadas. Reasoning about states of probabilistic sequential programs. In *Computer Science Logic*, pages 240–255. Springer, 2006b.

R. Chadha, L. Cruz-Filipe, P. Mateus, and A. Sernadas. Reasoning about probabilistic sequential programs. *Theoretical Computer Science*, 379(1): 142–165, 2007.

D. Cock. Verifying probabilistic correctness in Isabelle with pGCL. In *Proceedings of the 7th Systems Software Verification*, pages 1–10, Sydney, Australia, 2012.

J. den Hartog. *Probabilistic extensions of semantical models*. PhD thesis, PhD thesis, Vrije Universiteit Amsterdam, 2002.

J. Den Hartog and E. P. de Vink. Verifying probabilistic programs using a Hoare like logic. *International Journal of Foundations of Computer Science*, 13(03):315–340, 2002.

D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London, Series A*, 400(1818):97–117, 1985.

D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 439, pages 553–558. The Royal Society, 1992.

E. D'Hondt and P. Panangaden. Quantum weakest preconditions. *Mathematical Structures in Computer Science*, 16(03):429–451, 2006.

E. W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18(8):453–457, 1975.

R. W. Floyd. Assigning meanings to programs. *Mathematical aspects of computer science*, 19(19-32):1, 1967.

A. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron. Quipper: A scalable quantum programming language. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2013, pages 333–342, 2013.

D. Harel. First-order dynamic logic, volume 68 of lecture notes in computer science, 1979.

C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

J. Hurd, A. McIver, and C. Morgan. Probabilistic guarded commands mechanized in HOL. *Theoretical Computer Science*, 346(1):96–112, 2005.

N. Jansen, B. L. Kaminski, J.-P. Katoen, F. Olmedo, F. Gretz, and A. McIver. Conditioning in probabilistic programming. *Electronic Notes in Theoretical Computer Science*, 319:199–216, 2015.

Y. Kakutani. A logic for formal verification of quantum programs. In *Advances in Computer Science-ASIAN 2009. Information Security and Privacy*, pages 79–93. Springer, 2009.

J.-P. Katoen, F. Gretz, N. Jansen, B. L. Kaminski, and F. Olmedo. Understanding probabilistic programs. In *Correct System Design*, pages 15–32. Springer, 2015.

E. H. Knill. Conventions for quantum pseudocode. Technical Report LAUR-96-2724, Los Alamos National Laboratory, 1996.

D. Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 22(3):328–350, 1981.

D. Kozen. A probabilistic pdl. *Journal of Computer and System Sciences*, 30(2):162–178, 1985.

T. Kubota, Y. Kakutani, G. Kato, and Y. Kawano. A formal approach to unconditional security proofs for quantum key distribution. In *Unconventional Computation*, pages 125–137. Springer, 2011.

P. Mateus and A. Sernadas. Weakly complete axiomatization of exogenous quantum propositional logic. *Information and Computation*, 204(5):771–794, 2006.

A. McIver and C. Morgan. Developing and reasoning about probabilistic programs in pGCL. In *Refinement Techniques in Software Engineering*, pages 123–155. Springer, 2006.

C. Morgan. Proof rules for probabilistic loops. In *Proceedings of the BCS-FACS 7th Refinement Workshop, Workshops in Computing*, 1996.

C. Morgan and A. McIver. pGCL: Formal reasoning for random algorithms. *South African Computer Journal*, pages 14–27, 1999.

F. Olmedo, B. L. Kaminski, J.-P. Katoen, and C. Matheja. Reasoning about recursive probabilistic programs. *arXiv preprint arXiv:1603.02922*, 2016.

L. H. Ramshaw. *Formalizing the Analysis of Algorithms*. PhD thesis, Stanford University, 1979.

R. Rand and S. Zdancewic. VPHL: A verified partial-correctness logic for probabilistic programs. *Electronic Notes in Theoretical Computer Science*, 319:351–367, 2015.

J. W. Sanders and P. Zuliani. Quantum programming. In *Proceedings of the 5th International Conference on Mathematics of Program Construction*, volume 1837 of *Lecture Notes in Computer Science*, pages 80–99, 2000.

P. Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(04):527–586, 2004.

P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. IEEE, 1994.

P. W. Shor and J. Preskill. Simple proof of security of the bb84 quantum key distribution protocol. *Physical review letters*, 85(2):441, 2000.

M. D. Vásquez, N. Wolovick, and P. R. D'Argenio. Probabilistic Hoare-like logics in comparison. Technical report, Tech. rep. Universidad Nacional de Córdoba, 2004.

U. Vazirani. A survey of quantum complexity theory. In *Proceedings of Symposia in Applied Mathematics*, volume 58, pages 193–220, 2002.

J. Watrous. Lecture notes in introduction to quantum computing, 2006. URL `https://cs.uwaterloo.ca/~watrous/LectureNotes.html`.

M. Ying. Floyd–hoare logic for quantum programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 33(6):19, 2011.