Verification Techniques for Quantum Systems

Verification & Validation Techniques Report

Alex Della Schiava

February 2022

1 Outline

The purpose of this report is to illustrate part of the topics discussed in [1]. The topics were selected based on their relevance with the contents of the course of V&V Techniques. The report is structured as follows. Section 2 contextualizes the topics later to be described. Section 3 provides the required notions from both the fields of classical formal verification and quantum mechanics. Section 4 introduces quantum **while**-programs [2]. Section 5 illustrates Quantum Hoare Logic [2] and Quantum Weakest Precondition [2][3]. Section 6 focusses on Quantum CTL and its related Model Checking problem [4]. Finally, Section 7 shows an application of the results previously shown.

2 Introduction

The fast improvement in the construction of quantum computers has plotted a lot of attention towards the field of quantum computing. While discussing concepts of this field, sooner or later, one will very likely stumble upon the issue of error managing in quantum computation. In most cases, this first encounter is due to the probabilistic nature of quantum mechanics.

This observation alone does not automatically link quantum computing to the field of systems verification. First, a further analysis of quantum error is needed. As the authors of [1] point out, in quantum computing it is possible to identify three main causes to error:

1. **Randomness.** This factor is strictly related to quantum measurement. The probabilistic nature of this operation causes many quantum algorithms to fall into the category of *randomized algorithms*. Section 3.3 provides the details relatively to such operation.
2. **Hardware.** When speaking about quantum mechanics one often refers to *closed systems* (*e.g.* a qubit). Though, working with closed systems is easier said than done, as it is very hard to prevent the environment from interacting/interfering with such closed system. Interference is what consequently can lead to error.

3. **Defective software.** This category encloses all kinds of error caused by faulty implementations of quantum software.

This report focusses on cause 3: defective software, as it provides the link between formal verification and quantum computing. Defective software is a common issue in both classical and quantum computing. As for the classical case, successful results have been achieved through a variety of verification techniques. Among the most popular are: Model Checking and Deductive Verification. Their success has led many researchers to try and develop their quantum counterparts. This report shall try to present part of them, more specifically: Quantum CTL, Quantum Hoare Logic, Quantum Weakest Precondition. These techniques are particularly interesting as they fully rely on the quantum framework. Moreover, the choice of these specific techniques is motivated by the fact that they follow a *"program-centered"* approach, while the other techniques shown in [1] follow a *"circuit-centered"* approach.

3 Background

This section provides the needed background for the understanding of the results later described. Section 3.1 is very brief as these topics completely overlap the contents of the course.

3.1 Model Checking for Formal Verification

Among verification techniques, Model Checking has enjoyed particular success. One of its advantages is *exhaustiveness*: Model Checking analyzes all the possible behaviors of a system. More specifically, Model Checking is a formal verification technique which consists into verifying a finite (or finitely abstractable) system against a specific property. The Model Checking problem can be described through the expression $M, s \models \varphi$, where $\mathcal{M}$ is a model of the system, s is a state of $\mathcal{M}$ and φ is a property expressed by means of some logic. Thus, for the verification process to start, two preliminary steps are required: system modeling and property specification.

Kripke structeres are typically the formalism used to model a system. A Kripke structure is a tuple $M = (S, S_0, R, L)$ where S is the set of states of the system, $S_0 \subseteq S$ is the set of initial states, $R \subseteq S \times S$ is the transition relation and $L : S \to 2^{AP}$ is the labelling function with AP set of atomic propositions.

System modeling may lead to trouble: consider a system formed by m different components, each bearing its own set of states. A state of the resulting model should be able to describe any possible combination of states of the m components, thus requiring an exponential number of states with respect to m. This is the so called *state explosion problem*, the main weakness of Model Checking. Although quite daunting at first glance, different successful solutions to mitigate the problem have been proposed.

Computational Tree Logic *Temporal logics* make a fine candidate for the task of property specification. The power of temporal logics lies in their ability to express a time ordering of events without explicitly referring to time-points (or time-intervals).

This report will focus on Computational Tree Logic (CTL) as it will be useful to better introduce its quantum counterpart Quantum CTL. CTL is a branching-time logic, that is, it allows to specify properties over the computational tree starting from a specific state of the model. Briefly illustrating its syntax, CTL allows to quantify over different computations of the tree (*path quantifiers*). Path quantifiers are required to be interleaved by exactly one state quantifier. The syntax of CTL can be illustrated in terms of *state formulae* θ and *path formulae* T:

$$\theta ::= p \parallel p \wedge q \parallel \neg p \parallel A\,T \parallel E\,T$$
$$T ::= X\,\theta \parallel F\,\theta \parallel G\,\theta \parallel \theta\,U\,\theta \parallel ,$$

where $p \in AP$, A, E represent respectively the universal and existential path quantifiers and X, F, G, U are the well known state quantifiers.

The problem of Model Checking on CTL can be again expressed as $\mathcal{M}, s \models \theta$, where s is the initial state of the model $\mathcal{M}$ and θ a path formula. Such definition of the problem is equivalent to the problem finding all states s of $\mathcal{M}$ such that $\mathcal{M}, s \models \theta$. Following the latter definition, an *explicit-state* approach allows for the definition of an algorithm bearing a polynomial time-complexity with respect to the dimension of the set S of $\mathcal{M}$. Such result is indeed remarkable, although CTL does not come without disadvantages. In fact, CTL lacks a way of expressing *fairness* conditions. A minor flaw, which does have solutions, for instance the use of *Fair Kripke Models* allows to bring and surpass the obstacle in the phase of system specification.

3.2 Deductive Verification

Whereas Model Checking works by means of an exhaustive search over the states of a system, deductive verification utilizes *logical inference* to verify specific properties of a program/system. That is, it checks whether the property to be verified can be inferred from the initial assumptions on the program using a given set of inference rules. Although originally deductive verification processes were completely manual, with time different solutions have been proposed in order to automatize them. Nonetheless, deductive verification techniques still suffer the comparison with Model Checking in terms of time efficiency. On the plus side, they are able to provide a higher degree of explainability in their correctness result, though requiring a certain expertise in the field of deductive reasoning.

Floyd-Hoare Logic Floyd and Hoare laid the foundations of deductive verification proposing a set of axioms and inference rules to reason about the correctness of imperative programs. Before going deeper into the workings of Floyd-Hoare (FH) logic, it is useful to introduce the notation of *Hoare triples*. A Hoare

triple is an expression of the form $\{P\}S\{Q\}$, where S is a program and P and Q are *assertions* specified by means of first-order logic. P *(precondition)* is a set of assumptions on the initial state of S. On the other hand, Q *(postcondition)* is a set of properties to be verified against S. The Hoare triple $\{P\}S\{Q\}$ states that, the execution of S at a state s where $s \models P$ leads to a final state s' such that $s' \models Q$. If such statement holds, the Hoare triple is said to be valid and the program correct, formally $\models \{P\}S\{Q\}$.

Hoare Triples are used to define axioms and rules of FH logic over single statements of a programming language. Take for instance the *Composition Rule*:

$$\frac{\{P\}S_1\{R\}, \{R\}S_2\{Q\}}{\{P\}S_1; S_2\{Q\}}$$

where $S_1; S_2$ stands for program concatenation: first execute S_1 then S_2. The rule states that if the execution of $S_1; S_2$ on the assumption P implies Q to hold, then there exists a *midcondition* R, for which the two Hoare triples $\{P\}S_1\{R\}$ and $\{R\}S_2\{Q\}$ are valid. A list of other basic inference rules is provided in [1, Eq. (1)]. The verification process of FH logic fragments a program's instructions and proves their correctness using such inference rules.

Using this kind of machinery, one is able to only prove the *partial correctness* of a program: $\models_{par} \{P\}S\{Q\}$. The reason being these rules, as they are currently given, are not able to prove the termination of S. Thus, an implicit assumption on the termination of S is needed. The stronger notion of *total correctness* ($\models_{tot} \{P\}S\{Q\}$) gets rid of such assumption and requires a slight modification of the *While* rule.

Weakest Precondition As FH logic, the *weakest precondition calculus* is concerned with the correctness of Hoare triples, though its workings follow a different approach.

First, one must define what makes an assertion *weaker* than another. Given two assertions P, Q, P is said to be weaker than Q if $Q \Rightarrow P$. Thus, considering a state s, $s \models Q \Rightarrow s \models P$. These trivial observations are used to show how Q forces a stronger (or equal) condition on a state. With this in mind, given a postcondition Q on a program S, the weakest precondition $\mathsf{wp}.S(Q)$ is definable as the assertion P such that $\models_{tot} \{P\}S\{Q\}$ and for any assertion P' such that $\{P'\}S\{Q\}$, $P' \Rightarrow P$. Given the uniqueness of the weakest precondition, $\mathsf{wp}.S(Q)$ is indeed a computable function. Moreover, given the fact that it acts on the domain of predicates, it can be understood as a *predicate transformer*. It is important to add that such function is built *on top* of FH logic, that is, the function $\mathsf{wp}(\cdot, \cdot)$ is defined on the basic statements of the imperative language at hand in a coherent manner relatively to the inference rules of FH logic.

Having said that, suppose the goal is to prove the correctness of the Hoare triple $\{P\}S\{Q\}$. One can simply compute $\mathsf{wp}.S(Q)$. Finally, it suffices to check whether $P \Rightarrow \mathsf{wp}.S(Q)$. If so, $\models_{tot} \{P\}S\{Q\}$. Note that the last step is made possible by the inference rule of *Consequentiality*, which allows to both strengthen a precondition or weaken a postcondition.

On a last note, one can notice how the weakest precondition calculus shows *total correctness* of a program. Thus, the modification of the *While* rule anticipated in Section 3.2 is required:

$$\frac{\{i \wedge b \wedge e = v_0\} c \{i \wedge e < v_0\}, i \wedge b \Rightarrow e \geq 0}{\{i\} \texttt{while}\, b\, \texttt{do}\, c \{i \wedge \neg b\}} \tag{1}$$

Differently from the rule given in [1, Equation 1], here the termination of the loop is proven thanks to the auxiliary integer variable e and the so called *ghost variable* v_0. The idea behind such rule is that the variable e is decreased by each execution of c. On the other hand, the rule also implies that, during the loop $(i \wedge b)$, e never becomes negative. This guarantees the termination of the loop.

Turning the attention to partial correctness, and consequently to the original *While* rule, one refers to the *weakest liberal precondition* calculus. Here, the termination of the program is made an assumption.

3.3 Quantum Mechanics

This section introduces quantum mechanics illustrating four of its postulates. Reasoning about such postulates can be done in terms of two different notations: *state vectors* and *density operators*. For clarity, this brief introduction attempts to provide a parallel illustration using both notations.

Before tackling the first postulate, it is useful to introduce the Dirac notation. The writing $|\psi\rangle$ denotes a *ket*, which simply represents a vector in some vector space. On the other hand, $\langle\psi|$ denotes a *bra*, that is, the dual vector of $|\psi\rangle$. Simply put, $\langle\psi|$ is the row vector version of $|\psi\rangle$ with all of its elements turned into their respective complex conjugate. As an example:

$$\text{if } |\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \text{ then } \langle\psi| = \begin{pmatrix} \alpha^* & \beta^* \end{pmatrix}$$

where $\alpha, \beta \in \mathbb{C}$ and α^*, β^* represent their complex conjugate.

Postulate 1. A quantum system has an associated Hilbert space. Moreover, a quantum system is completely described by a state vector belonging in such Hilbert space. Such state vector is a unit vector, that is, a vector with norm equal to 1.

A *qubit* is a common example of a quantum system belonging to the Hilbert space $\mathcal{H}^2$. Consider a qubit described by the vector $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where $|0\rangle$ and $|1\rangle$ represent the vectors of the computational basis $(1,0)^T, (0,1)^T$. Its norm is defined as $\sqrt{\| \langle\psi|\psi\rangle \|} = \sqrt{|\alpha|^2 + |\beta|^2} = 1$. Moreover, if $\alpha, \beta \neq 0$, $|\psi\rangle$ is said to be in a *superposition* of states $|0\rangle, |1\rangle$. That is, when measured, $|\psi\rangle$ has probability $|\alpha|^2$ to give result $|0\rangle$ and probability $|\beta|^2$ to give result $|1\rangle$.

Proceeding using such example, the same qubit can be described by a density operator (or density matrix): $\rho = |\psi\rangle\langle\psi|$. A density operator ρ is a positive

operator ($\forall \varphi \, \langle \varphi | \, \rho \, | \varphi \rangle \geq 0$) such that $tr(\rho) = 1$, where $tr(\cdot)$ represents the *trace*. Density operators come in handy when describing *mixed states*, that is, states of which one does not have total knowledge of. A mixed state ρ can be understood as an ensemble of different quantum states $\{p_i, |\psi_i\rangle\}$, where p_i denotes the probability of the system being in state $|\psi_i\rangle$. Then, the system can be described by the density operator $\rho = \sum_i p_i \, |\psi_i\rangle \, \langle \psi_i|$.

Postulate 2. The evolution of a closed quantum system over time is given by means of *unitary operators*.

A more explicit representation of Postulate 2 can be given both in terms of state vectors and density operators. Suppose to have a state $|\psi\rangle$ with corresponding density operator $\rho = |\psi\rangle \, \langle \psi|$. The application of a unitary operator U on such state can be described as follows:

$$|\psi'\rangle = U \, |\psi\rangle \quad \rho' = U \rho U^\dagger$$

where $|\psi'\rangle$ and ρ' both represent the resulting state. Here, $U^\dagger$ represents the adjoint of U, namely, its conjugate transposed version. A unitary operator is such that $UU^\dagger = I$, where I is the identity operator.

Postulate 3. A *quantum measurement* is described by a collection $\{M_m\}$ of *measurement operators*. In such writing, the subscripts m denote the possible outcomes of the measurement.

Given a state $|\psi\rangle$ with corresponding density operator ρ the probability of measuring m is:

$$p(m) = \langle \psi | \, M_m^\dagger M_m \, |\psi\rangle = tr(M_m^\dagger M_m \rho)$$

The state of the system after the measurement of value m is:

$$|\psi'\rangle = \frac{M_m \, |\psi\rangle}{\sqrt{p(m)}} \quad \rho' = \frac{M_m \rho M_m^\dagger}{p(m)}$$

The results concerning density operators can easily be derived from the ones of state vectors. For reasons of brevity, they are not shown here.

Quantum measurements are quite counter-intuitive in the way they affect the closed system being measured. The reason behind this process is given by the inevitable interaction between the measuring apparatus and the system. Such interaction causes the measuring apparatus and the initial system to become part of a larger system.

This report is mainly concerned with the subclass of *projective measurements*, also known as *observables*. Projective measurements are described by collections of *projectors* $\{P_m\}$. A projector P is a particular kind of Hermitian operator ($P^\dagger = P$) for which $P^2 = P$. Projective measurements can be shown to be equivalent to general measurements under specific conditions.

Postulate 4. The state space of a composite quantum system is given by the tensor product ($\otimes$) of the state spaces of its component quantum systems.

A simple example to better illustrate Postulate 4 is that of a composite system of two qubits $|\psi\rangle, |\varphi\rangle$. The state space of such composite system shall take into account both the state spaces of $|\psi\rangle$ and $|\varphi\rangle$, namely $\mathcal{H}^2$. The resulting state space will, thus, be $\mathcal{H}^2 \otimes \mathcal{H}^2 = \mathcal{H}^4$. As for density operators, the result can be directly translated as the tensor product of density matrices.

More interestingly, the notion of composite system allows to introduce *entanglement*. A composite system is said to be in an entangled state if it cannot be expressed by means of a tensor product of states belonging to its components.

The results illustrated in this report are will mostly be given in terms of density operators. Though in some cases a slightly more general notion is needed, that of *partial density operators* introduced by Selinger following [5, Convention 3.3].

Definition 3.1. Given a Hilbert space $\mathcal{H}$, a partial density operator $\rho \in \mathcal{D}^-(\mathcal{H})$ is a density operator such that $tr(\rho) \leq 1$, where $\mathcal{D}^-(\mathcal{H})$ is the set of all partial density operators in $\mathcal{H}$.

Section 4.2 will clarify the importance of these *non-normalized* states in the definition of *non-probabilistic* transition semantics for quantum programs.

4 Quantum while-programs

This section illustrates the imperative language of quantum **while**-programs following the work of Ying in [2]. Such quantum programming language provides a playground for the quantum deductive verification techniques described in Section 5.

Typically, quantum programming languages are classified depending on their use, if any, of classical variables. For instance, the language QLP proposed by Selinger in [5] uses quantum variables for data representation and classical variables for control state. On the other hand, quantum **while**-programs use quantum variables for both purposes.

Without loss of generality, the variable types can be restricted to boolean and integer. The domains of the types are defined by the following Hilbert spaces:

$$\mathcal{H}_{\text{boolean}} = \mathcal{H}_2, \quad \mathcal{H}_{\text{integer}} = \mathcal{H}_\infty$$

Thus, a boolean variable is a qubit. As for integers, the use of an infinite Hilbert space shall not seem daunting as the required tools are generalizable to such case. Having said that, given a quantum variable q, $type(q)$ denotes its type and $\mathcal{H}_q$ its domain/state space.

Finite sequences of quantum variables are called *quantum registers*. A quantum register $\bar{q} = q_1, \ldots, q_n$ belongs to the tensor product of the state spaces of all the variables it represents:

$$\mathcal{H}_{\bar{q}} = \bigotimes_{i=1}^{n} \mathcal{H}_{q_i}$$

A quantum register can, in some sense, be understood as a specific kind of variable.

4.1 Syntax

The syntax of quantum **while**-programs is defined as follows:

$$S ::= \texttt{skip} \,\|\, q := |0\rangle \,\|\, \bar{q} := U\bar{q} \,\|\, S_1; S_2 \,\|\, \texttt{measure}\, M[\bar{q}] : \bar{S} \,\|\, \texttt{while}\, M[\bar{q}] = 1\, \texttt{do}\, S$$

The meaning of each statement will be clarified defining the operational semantics of the language. Nonetheless, a brief discursive introduction might be useful:

- `skip`. This instruction is the same as its classical counterpart.
- **Initialization** $q := |0\rangle$. In quantum programming, it is crucial not to mistake initialization with assignment. An assignment requires the state of a system to be copied onto another system. An impossible procedure as proved by the No-cloning theorem.
- **Evolution** $\bar{q} := U\bar{q}$. U is a unitary operator. Its workings follow Postulate 2.
- **Measurement** $\texttt{measure}\, M[\bar{q}] : \bar{S}$. $\bar{S}$ represents a set of programs $\{S_m\}$. If the measurement M on $\bar{q}$ lays outcome m, S_m program S_m is executed. `measure` can be understood as the quantum counterpart of an `if` statement.
- **While** $\texttt{while}\, M[\bar{q}] = 1\, \texttt{do}\, S$. The `while` statement also performs a measurement M. Differently from the `measure` statement, $M = \{M_0, M_1\}$, that is, only outcomes 0 and 1 are possible.

Given a program S the set of variables $\mathcal{V}(S)$ is defined inductively on its structure. Given the simplicity of such definition, the reader is referred to [2, Definition 3.1]

4.2 Operational semantics

Key in the definition of operational semantics is the concept of *state* of a quantum **while**-program.

Definition 4.1. A state of a program S is represented by a partial density operator ρ belonging to the Hilbert space $\mathcal{H}_S$, where

$$\mathcal{H}_S = \bigotimes_{q \in \mathcal{V}(S)} \mathcal{H}_q$$

Coherently with Postulate 4, a state of a program is thus simply a composite system having all of its variables as its components.

Definition 4.2. A configuration of a quantum **while**-program is a pair $\langle S, \rho \rangle$, where S represents the instructions still to be executed and ρ the current state.

Finally, the operational semantics of a quantum **while**-program are described by a transition relation among quantum configurations. The writing:

$$\langle S, \rho \rangle \to \langle S', \rho' \rangle$$

denotes a computational step through the program S. S' represents S devoid of its first instruction, while ρ' represents the new state of the program. For simplicity $\downarrow$ represents an empty program, thus, the writing $\langle \downarrow, \rho \rangle$ represents a *terminating configuration.*

The transition rules for each statement of the language can be given as follows. Note: from now on the program "$\texttt{while}\, M[\bar{q}] = 1\, \texttt{do}\, S$" shall be shortened as "$\texttt{while}$".

$$\frac{}{\langle \texttt{skip}, \rho \rangle \to \langle \downarrow, \rho \rangle} \quad (\textit{Skip})$$

$$\frac{}{\langle q := 0, \rho \rangle \to \langle \downarrow, \rho_0^q \rangle} \quad (\textit{Init})$$

$$\frac{}{\langle \bar{q} := U\bar{q}, \rho \rangle \to \langle \downarrow, U\rho U^\dagger \rangle} \quad (\textit{Unit})$$

$$\frac{\langle S_1, \rho \rangle \to \langle S_1', \rho' \rangle}{\langle S_1; S_2, \rho \rangle \to \langle S_1'; S_2, \rho' \rangle}, \text{ where } \langle \downarrow; S_2, \rho \rangle = \langle S_2, \rho \rangle, \quad (\textit{Comp})$$

$$\frac{}{\langle \texttt{measure}\, M[\bar{q}] : \bar{S}, \rho \rangle \to \langle S_m, M_m \rho M_m^\dagger \rangle} \quad (\textit{If})$$

$$\frac{}{\langle \texttt{while}, \rho \rangle \to \langle \downarrow, M_0 \rho M_0^\dagger \rangle} \quad (\textit{Loop 0})$$

$$\frac{}{\langle \texttt{while}, \rho \rangle \to \langle S; \texttt{while}, M_1 \rho M_1^\dagger \rangle} \quad (\textit{Loop 1})$$

Here, it should be clarified that *If* describes a collection of rules: one for each outcome m of M. For the sake of brevity, here only the rules *Init*, *Loop 0* and *Loop 1* are covered in depth, as they lay down the most interesting consequences.

Rule *Init* describes two rules, respectively for boolean and integer initialization. Here, the workings of boolean initialization are described, for the integer case is illustrated at [2, Pages 10-11]. Consider the configuration $\langle q := 0, \rho \rangle$. Here, ρ describes the state of all variables of the program, including q. Consider $|0\rangle_q \langle 0| + |0\rangle_q \langle 1|$ to be the operator acting on the state space of ρ but only initializing variable q to 0. That is, the cylindrical extension of $|0\rangle_q \langle 0| + |0\rangle_q + \langle 1|$:

$$|0\rangle_q \langle 0| + |0\rangle_q \langle 1| = \Big[|0\rangle \langle 0| + |0\rangle \langle 1| \Big] \bigotimes_{q' \neq q} I_{q'}$$

Applying $|0\rangle_q \langle 0|$ to ρ gives:

$$|0\rangle_q \langle 0| \rho |0\rangle_q \langle 0| = \Big[|0\rangle \langle 0|q\rangle \langle q|0\rangle \langle 0| + |0\rangle \langle 1|q\rangle \langle q|1\rangle \langle 0| \Big] \bigotimes_{q' \neq q} |q'\rangle \langle q'| \quad (2)$$

$$= |0\rangle \langle 0| \bigotimes_{q' \neq q} |q'\rangle \langle q'| = \rho_0^q \tag{3}$$

Thus, q has been successfully initialized to state $|0\rangle$.

The transition rules *Loop 0* and *Loop 1* define the transition semantics for the `while` statement. As previously stated, the guard of the `while` statement works through a measurement $M = \{M_0, M_1\}$. Coherently with Postulate 3, the state ρ is altered by the measurement process. Now one may argue that, again according to Postulate 3, the post-measurement state should be:

$$\rho_0 = \frac{M_0 \rho M_0^\dagger}{p(m)} = \frac{M_0 \rho M_0^\dagger}{tr(M_0^\dagger M_0 \rho)} \qquad \text{(example on Loop 0)}$$

This is where partial density operators (see Defintion 3.1) come into play. $M_0 \rho M_0^\dagger$ is indeed a partial density operator, thus it has not been normalized to 1. This representation encodes the probability $p(m)$ of being in state $M_0 \rho M_0^\dagger$ into the state itself. Usually, states are normalized so that $tr(\rho') = 1$, in this case though:

$$tr(M_0 \rho M_0^\dagger) = tr(M_0^\dagger M_0 \rho) = p(m) \tag{4}$$

Thus, the probability is easily retrievable. The use of such normalization convention allows to shift probability from transitions into states. Thus avoiding the definition of probabilistic transition semantics such as:

$$\frac{}{\langle \texttt{while}, \rho \rangle \xrightarrow{p_m} \langle S_m, \rho_m \rangle}$$

Of course, avoiding probabilistic transitions introduces non-determinism[1] as two rules are defined for the same configuration. This is also the case for the *If* rules.

Having said that, following are some notational remarks and definitions regarding the transition relation just defined. The writing $\langle S, \rho \rangle \xrightarrow{n} \langle S', \rho' \rangle$ denotes that the configuration $\langle S', \rho' \rangle$ can be reached from configuration $\langle S, \rho \rangle$ in n computational steps. This can be generalized with the writing:

$$\langle S, \rho \rangle \xrightarrow{*} \langle S', \rho' \rangle$$

which states that $\langle S, \rho \rangle \xrightarrow{n} \langle S', \rho' \rangle$ for some n.

Definition 4.3. A sequence $\langle S, \rho \rangle \to \cdots \to \langle S', \rho' \rangle \to \ldots$ is a *computation* of S starting from state ρ if it cannot be extended on the left-hand side. A finite sequence with final configuration of the form $\langle \downarrow, \rho_f \rangle$, is said to be a *terminating computation* in state ρ_f. If a computation does not terminate it is said to be *diverging*.

A program S is said to be *diverging* from ρ if there exists a diverging computation starting from $\langle S, \rho \rangle$.

[1] Ying underlines in different occasions the deterministic nature of quantum **while**-programs. The manifestation of non-determinism is strictly restricted to this way of representing the operational semantics.

4.3 Denotational Semantics

Given a quantum **while**-program S, its denotational semantics is described by the semantic function $[\![S]\!] : \mathcal{D}^-(\mathcal{H}_S) \to \mathcal{D}^-(\mathcal{H}_S)$ that maps the initial state ρ onto the sum of all the final states reached via execution of S. Formally:

$$[\![S]\!](\rho) = \sum \left\{ |\rho' : \langle S, \rho\rangle \xrightarrow{*} \langle \downarrow, \rho'\rangle| \right\} \tag{5}$$

The definition of this function uses the notion of *multiset* ("$|\cdot|$") in order to consider different occurences of the same final configuration.

Proposition 5.1 of [2] defines the denotational semantics of the basic statements of quantum **while**-programs. Avoiding to list all such definitions, this report provides an explanation by example.

Example 4.1. Consider the following program:

$$S \equiv [q := 0;\, q := Hq;\, \texttt{measure}\, M[\bar{q}] : \bar{S}]$$

where $M = \{M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|\}$, $\bar{S} = S_0, S_1$ with $S_0 \equiv S_1 \equiv \texttt{skip};$. It also should be noted that H represents a Hadamard gate. To calculate $[\![S]\!](\rho)$ one first needs to find all terminating states of S. This is done exploiting the transition rules given in Section 4.2.

$$\langle S, \rho\rangle \to \langle q := 0, \rho\rangle \to \langle q := Hq, \rho_1\rangle \to \langle \texttt{measure}, \rho_2\rangle \to \begin{cases} \langle \downarrow, \rho_3^0\rangle, \text{ if } m = 0 \\ \langle \downarrow, \rho_3^1\rangle, \text{ if } m = 1 \end{cases}$$

$$\rho_1 = [\![q := 0]\!](\rho) = |0\rangle\langle 0|$$

$$\rho_2 = [\![q := Hq]\!](\rho_1) = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \frac{\langle 0| + \langle 1|}{\sqrt{2}}$$

$$\rho_3^0 = [\![q := \texttt{measure}\, M[\bar{q}] : \bar{S}]\!](\rho_2) = \frac{1}{2}|0\rangle\langle 0|$$

$$\rho_3^1 = [\![q := \texttt{measure}\, M[\bar{q}] : \bar{S}]\!](\rho_2) = \frac{1}{2}|1\rangle\langle 1|$$

Notice how the `measure` statement gives rise to two different terminating states: ρ_3^0, ρ_3^1. Following the defintion of the semantic function in Eq. 5, one obtains

$$[\![S]\!](\rho) = \rho_3^0 + \rho_3^1 = \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|) = \frac{1}{2}I$$

Having said that, perhaps the most interesting result concerns the effect the semantic function has on the trace of a partial density operator. As Proposition 5.2 in [2] shows, for any program S starting from any state $\rho \in \mathcal{D}^-(\mathcal{H})$, it is always the case that:

$$tr([\![S]\!](\rho)) \leq tr(\rho) \tag{6}$$

More specifically, it is shown that $tr([\![S]\!](\rho)) < tr(\rho)$ if and only if S has a diverging computation. This result allows to easily calculate the probability that program S will not terminate, that is $tr(\rho) - tr([\![S]\!](\rho))$ (Remember that from Eq. 4 $tr(\rho')$ is the probability of reaching state ρ').

5 Quantum Deductive Verification

This section introduces the verification techniques of Quantum Hoare Logic (QHL) [2] and Quantum Weakest Precondition [2][3]. Thanks to the results discussed in the previous pages, only few notions are left to be introduced. As for the rest, the following work is a natural translation of the classical results into the quantum case.

5.1 Quantum predicates

Section 4.2 gave a natural definition of the state of a program. Now, a way of specifying a *quantum predicate* is needed. For this purpose, the report refers to the definition given by [3].

Definition 5.1. A quantum predicate is a projective measurement with eigenvalues bounded by 1.

A basic definition of projective measurements was given in Section 3.3, Postulate 3. Being Hermitian, a projective measurement M has a spectral decomposition of the form $\sum_m mP_m$. In such representation P_m is a projector on the space of the eigenvalue m. For the case of this report, this result is has the only purpose of emphasizing the corrispondence between eigenvalues and possible outcomes of a projective measurement.

This definition of quantum predicates stems from probability theory. As such, whereas a classical predicate is evaluated in terms of truth, a quantum predicate M relies on its *expectation value* $\langle M \rangle$, which is endowed with a compact representation:

$$\langle M \rangle = \langle \psi | M | \psi \rangle = tr(M | \psi \rangle \langle \psi |) = tr(M\rho) \tag{7}$$

where $|\psi\rangle$ is any state and $\rho = |\psi\rangle \langle \psi|$.

Considering $\mathcal{P}(\mathcal{H})$ to be set of quantum predicates acting on the Hilbert space $\mathcal{H}$ and borrowing symbol $\models$ from the classical case, one obtains the following definition:

Definition 5.2. Given a quantum state $\rho \in \mathcal{D}^-(\mathcal{H})$, a quantum predicate $M \in \mathcal{P}(\mathcal{H})$ and a real number $r \in [0, 1]$ the notion of *quantum satisfiability* is denoted as:

$$\rho \models_r M \iff tr(M\rho) \geq r,$$

stating that ρ satisfies M with probability r.

A reason as to why eigenvalues are to be bounded by 1 is still due. Such condition bounds the expectation value $tr(M\rho)$ to be bounded by 1 too, thus respecting the basic notion of probability. Moreover, this condition allows to organize the elements of $\mathcal{P}(\mathcal{H})$ into a Complete Partial Order (CPO) using the Löwner partial order: $\sqsubseteq$. The Löwner partial order allows for an intuitive comparison of quantum predicates (and also partial density operators) and can be defined as follows:

Definition 5.3. (Löwner partial order) Given $P, Q \in \mathcal{P}(\mathcal{H})$, $P \sqsubseteq Q$ if and only if $Q - P$ is a positive operator.

The CPO will have $I_{\mathcal{H}}$ (Identity) as top element and $0_{\mathcal{H}}$ as bottom. The importance of this result will be clear when desecribing the *quantum predicate transformer semantics*, that is *quantum weakest precondition*.

Another relevant result regarding quantum predicates is illustrated in [3, Proposition 2.2]:

Proposition 5.1. Given two quantum predicates $P, Q \in \mathcal{P}(\mathcal{H})$ and a partial density operator $\rho = |\psi\rangle \langle\psi|$, $P \sqsubseteq Q$ if and only if $tr(P\rho) \leq tr(Q\rho)$.

A lot of theory concerning the poset $(\mathcal{P}, \sqsubseteq)$ is left uncovered in this report. Though the results provided should be sufficient for an understanding of the following sections.

5.2 Quantum Hoare Logic

The purpose of QHL is to reason about the correctness of imperative quantum programs. Aside from a new appropriate set of axioms and rules, the notion of correctness itself needs to be adapted to the quantum case.

Definition 5.4. (Quantum Hoare Triple) Given a program S, quantum predicates $P, Q \in \mathcal{P}(\mathcal{H}_S)$ and a partial density operator $\rho \in \mathcal{D}^-(\mathcal{H}_S)$, the expression $\{P\}S\{Q\}$ is used to denote P, Q as respectively the quantum precondition and postcondition of S. That is,

$$tr(P\rho) \leq tr(Q[\![S]\!](\rho)) \tag{8}$$

To better understand this definition, consider $\rho_f = [\![S]\!](\rho)$, as the final state of the program. Then, Eq. 8 is equivalent to $\rho \models_r P \Rightarrow \rho_f \models_r Q$.

It is now possible to adapt the definitions of total and partial correctness for the quantum case. Thus, given a program S and quantum predicates $P, Q \in \mathcal{P}(\mathcal{H}_S)$:

Definition 5.5. (Quantum Total Correctness) The Hoare triple $\{P\}S\{Q\}$ is valid according to total correctness, formally $\models_{tot} \{P\}S\{Q\}$, if for all $\rho \in \mathcal{D}^-(\mathcal{H}_S)$ it holds that:

$$tr(P\rho) \leq tr(Q[\![S]\!](\rho)) \tag{9}$$

Definition 5.6. (Quantum Partial Correctness) The Hoare triple $\{P\}S\{Q\}$ is valid according to partial correctness, formally $\models_{par} \{P\}S\{Q\}$, if for all $\rho \in \mathcal{D}^-(\mathcal{H}_S)$ it holds that:

$$tr(P\rho) \leq tr(Q[\![S]\!](\rho)) + \big(tr(\rho) - tr([\![S]\!](\rho)\big) \tag{10}$$

Definition 5.6 deserves an explanation. The condition of partial correctness can be rephrased as: "if P is satisfied, then either S does not terminate or Q is satisfied upon termination of S". From basic probability, the disjunction of

two events is equivalent to the sum of their probabilities. Finally, remembering Eq. 6, the probability that S will not terminate is equal to $\big(tr(\rho) - tr([\![S]\!](\rho))\big)$.

Focussing on the axiomatic base and the inference rules of QHL, similarly to the classical case, partial and total correctness differ in their *While* rule. Following, Eq. 11 shows the proof system for partial correctness (qPD):

$$
\begin{array}{ll}
(\textit{Axiom Skip}) & \dfrac{}{\{P\}\texttt{skip}\{P\}} \\[2ex]
(\textit{Axiom Init Boolean}) & \dfrac{}{\left\{ |0\rangle_q \langle 0| P |0\rangle_q \langle 0| + |1\rangle_q \langle 0| P |0\rangle \langle 1| \right\} q \coloneqq 0 \{P\}} \\[2ex]
(\textit{Axiom Init Integer}) & \dfrac{}{\left\{ \sum_{n=-\infty}^{\infty} |n\rangle_q \langle 0| P |0\rangle_q \langle n| \right\} q \coloneqq 0 \{P\}} \\[2ex]
(\textit{Axiom Unitary Transformation}) & \dfrac{}{\{U^\dagger P U\}\, \bar{q} \coloneqq U\bar{q}\, \{P\}} \\[2ex]
(\textit{Rule Composition}) & \dfrac{\{P\}\, S_1\, \{R\}, \{R\}\, S_2\, \{Q\}}{\{P\}\, S_1; S_2\, \{Q\}} \\[2ex]
(\textit{Rule Measurement}) & \dfrac{\{P_m\} S_m \{Q\}}{\left\{ \sum_m M_m^\dagger P_m M_m \right\} \texttt{measure} M[\bar{q}] : \bar{S}\, \{Q\}} \\[2ex]
(\textit{Rule Loop Partial}) & \dfrac{\{Q\}\, S \left\{ M_0^\dagger P M_0 + M_1^\dagger P M_1 \right\}}{\left\{ M_0^\dagger P M_0 + M_1^\dagger P M_1 \right\} \texttt{while}\, M[\bar{q}]\, \texttt{do}\, S\, \{P\}} \\[2ex]
(\textit{Rule Order}) & \dfrac{P \sqsubseteq P',\ \{P'\}\, S\, \{Q'\},\ Q' \sqsubseteq Q}{\{P\}\, S\, \{Q\}}
\end{array}
\tag{11}
$$

The reasoning behind such definitions will be made clearer through the example in Section 7. For the time being, consider the axiom for unitary transformation. Checking whether a state $\bar{q}$ satisfies the precondition means calculating $tr(U^\dagger P U \rho)$. Being the trace equal under rotation one obtains:

$$tr(U^\dagger P U \bar{q}) = tr(P U \bar{q} U^\dagger) = tr(P [\![\bar{q} \coloneqq U\bar{q}]\!]) \tag{12}$$

The right hand side of Eq. 12 is exactly the probability of the final state respecting the postcondition. Since the two probabilities coincide, the Hoare triple is partially correct.

The proof system for total correctness (qTD) is not presented as it requires a rather cumbersome explanation of (P, ϵ)-*bound functions*. To give an idea, bound functions are also employed in the representation of the *While* rule for classical total correctness. Looking back at Eq. 1 in Section 3.2, the bound function is represented by the auxiliary variable e.

Ying concludes his work showing soundness and (relative[2]) completeness for both partial and total proof systems. That is, for any quantum program S and quantum predicates $P, Q \in \mathcal{P}(\mathcal{H}_S)$ it holds that:

$$\models_{par/tot} \{P\}Q\{S\} \iff \vdash_{\text{qPD/qTD}} \{P\}S\{Q\},$$

[2]Relative to the theory of complex numbers.

where $\vdash$ represents the notion of derivation either through the partial proof system (qPD) or the total proof system (qTD).

5.3 Quantum Weakest Precondition

Quantum Weakest Precondition was first proposed in [3] indipendently from any specific quantum language. In [2] it has been adopted for the verification of imperative quantum languages. This report focusses on the latter definition.

The notions of quantum partial and total correctness allow this section to branch out into quantum weakest precondition (QWP) and quantum liberal precondition (QWLP). Given a program S and a quantum predicate $Q \in \mathcal{P}(\mathcal{H}_S)$, it is possible to give their definitions:

Definition 5.7. The quantum weakest precondition of Q with respect to a program S is a quantum predicate $\mathsf{qwp}.S(Q) \in \mathcal{P}(\mathcal{H}_S)$ such that:

1. $\models_{tot} \{\mathsf{qwp}.S(Q)\}\, S\, \{Q\}$;
2. For any quantum predicate $P \in \mathcal{P}(\mathcal{H}_S)$,

$$\models_{tot} \{P\}\, S\, \{Q\} \text{ implies } P \sqsubseteq \mathsf{qwp}.S(Q)$$

Definition 5.8. The quantum weakest liberal precondition of Q with respect to a program S is a quantum predicate $\mathsf{qwlp}.S(Q) \in \mathcal{P}(\mathcal{H}_S)$ such that:

1. $\models_{par} \{\mathsf{qwlp}.S(Q)\}\, S\, \{Q\}$;
2. For any quantum predicate $P \in \mathcal{P}(\mathcal{H}_S)$,

$$\models_{par} \{P\}\, S\, \{Q\} \text{ implies } P \sqsubseteq \mathsf{qwlp}.S(Q)$$

Condition 2 of both definitions deserves attention. Notice how Proposition 5.1 allows to use the relation $\sqsubseteq$ to compare strength of among quantum operators, thus providing a natural translation for logical implication in the quantum case. Suppose for instance $P \sqsubseteq Q$. Thanks to Proposition 5.1, one obtains, for any $\rho \in \mathcal{D}^{-}(\mathcal{H}_S)$, $tr(P\rho) \leq tr(Q\rho)$. Given the definition of the $\models_r$ relation for any $r \in [0,1]$, it is possible to conclude that, for any $\rho \in \mathcal{D}^{-}(\mathcal{H}_S)$,

$$\rho \models_r P \implies \rho \models_r Q$$

Concluding, Q is weaker than P.

Propositions 7.1 and 7.2 of [2] define respectively the QWP and QWLP calculi for the basic statements of quantum **while**-programs. Aligning this section consistently with Section 5.2, the focus is on partial correctness, that is QWLP.

1) $\mathsf{qwlp}.[\texttt{skip}](P) = P$;
2) $\mathsf{qwlp}.[q := 0](P) = |0\rangle_q \langle 0| P |0\rangle_q \langle 0| + |1\rangle_q \langle 0| P |0\rangle_q \langle 1|, \quad type(q) = \text{boolean}$;

3) $\mathsf{qwlp}.[q := 0](P) = \sum_{-\infty}^{\infty} |n\rangle_q \langle 0| P |0\rangle_q \langle n|, \qquad type(q) = \text{integer};$

4) $\mathsf{qwlp}.[\bar{q} := U\bar{q}](P) = U^\dagger P U;$

5) $\mathsf{qwlp}.[S_1; S_2](P) = \mathsf{qwlp}.S_1(\mathsf{qwlp}.S_2(P));$

6) $\mathsf{qwlp}.[\texttt{measure}\, M[\bar{q}] : \bar{S}](P) = \sum_m M_m^\dagger P M_m;$

7) $\mathsf{qwlp}.[\texttt{while}](P) = M_0^\dagger P M_0 + M_1^\dagger(\mathsf{qwlp}.[\texttt{while}])M_1$

These definitions may appear rather complex at first, though one can easily notice how each definition sort of *reverses* the effects of its relative statement on the state. This is indeed possible as, according to Postulate 2, quantum evolutions are described in terms of unitary operators.

Concerning the `while` case, as mentioned in Section 5.1, $(\mathcal{P}(\mathcal{H}_S), \sqsubseteq)$ is a complete partial order. The recursive characterization provided in definition 7) follows from a greatest fixpoint representation.

6 Quantum Model Checking

In [4], Quantum Computational Tree Logic (QCTL) was developed in order to provide a means for temporal reasoning over quantum systems. QCTL is built upon CTL and the *decidable fragment of the Endogenous Quantum Propositional Logic* (dEQPL). This section provides an illustration of a particular version of dEQPL restricted over finite Hilbert spaces, which will be then used in the description of QCTL.

6.1 dEQPL

Whereas classical propositional logic deals with propositional letters, dEQPL deals with qubits. Given a set of n qubits $\mathsf{qB} = \{q_1, \ldots, q_n\}$, each qubit is represented by a propositional symbol q_i, for all $1 \leq i \leq n$. The truth value of a qubit should be understood as the state vector that describes it, for instance, the qubit described by $|q_i\rangle = \alpha|0\rangle + \beta|1\rangle$ has probability $|\alpha|^2$ of being in state $|0\rangle$ (false) and probability $|\beta|^2$ of being in state $|1\rangle$ (true).

Coherently with Postulate 4, a valuation over the set qB is a state $|\psi\rangle$ belonging to the tensor product of the state spaces of the n qubits, that is:

$$\bigotimes_{i=1}^{n} \mathcal{H}^2 = \mathcal{H}^{2^n} = \mathcal{H}_{\mathsf{qB}}$$

The computational basis of $\mathcal{H}_{\mathsf{qB}}$ is denoted as the collection of *"pure valuations"* $\{|v_i\rangle\}_{i=1}^{2^n}$. Given a state $|\psi\rangle \in \mathcal{H}_{\mathsf{qB}}$, the inner product $\langle v_i|\psi\rangle$, is called the *logical amplitude* of $|\psi\rangle$ with respect to $|v_i\rangle$. That is, the square root of the probability of valuation v_i being true in state $|\psi\rangle$. One can easily notice there exists a bijection between the subsets of qB and the set of valuations over qB.

For instance, the subset $A = \{q_1, q_3\}$ is mapped onto the evaluation $|v_A\rangle = |10100\ldots0\rangle$.

Syntax The terms t of dEQPL formulae γ (*quantum formulae*) are real numbers. Terms themselves can be constructed from classical formulae α. Having $m \in \mathbb{Z}$ and $A \subseteq \mathsf{qB}$, the grammar can be defined inductively as follows:

$$\alpha \coloneqq \bot \parallel q \parallel \alpha \Rightarrow \alpha \qquad \text{(Classical f.lae)}$$

$$t \coloneqq x \parallel m \parallel t + t \parallel t * t \parallel \mathsf{Re}(|\top\rangle_A) \parallel \mathsf{Im}(|\top\rangle_A) \parallel \int \alpha \qquad \text{(Terms)}$$

$$\gamma \coloneqq t \leq t \parallel \perp\!\!\!\perp \parallel \gamma \sqsupset \gamma \qquad \text{(Quantum f.lae)}$$

Classical formula are propositional formulae where q stands for a qubit symbol in qB. Terms can range over a countable set of real variables $X = \{x_k : k \in \mathbb{N}\}$. For m an integer number is denoted. Given a quantum state $|\psi\rangle$, terms $\mathsf{Re}(|\top\rangle_A)$ and $\mathsf{Im}(|\top\rangle_A)$ are used to denote respectively the real and imaginary part of the logical amplitude $\langle v_A|\psi\rangle$. Next in order, the term $\int \alpha$ is used to denote the probability of the classical formula α being satisfied. To make this calculation, each qubit occurring in α is supposed to be measured in the computational basis, that is the measurement $M = \sum_{i=1}^{2^n} |v_i\rangle \langle v_i|$

Finally, quantum formulae can be given by comparison formulae over terms, the *quantum falsum term* $\perp\!\!\!\perp$ or connections of them through the operator $\sqsupset$. Operator $\sqsupset$ can be understood as the *quantum implies* operator.

Semantics A quantum formulae is interpreted over a quantum state $|\psi\rangle \in \mathcal{H}_{\mathsf{qB}}$ and a countable set of real variables $X = \{x_k : k \in \mathbb{N}\}$. Set X is interpreted as an assignment $\lambda : X \to \mathbb{R}$. Given $x \in X$, the writing $\lambda(x)$ denotes the value of variable x. (In [4] the assignment is denoted as ρ which is here replaced by λ to avoid any confusion with density operators). Following are the denotations ($[\![\cdot]\!]$) of the least trivial terms in dEQPL:

$$\begin{aligned}
[\![x]\!]_{|\psi\rangle\lambda} &= \lambda(x) && \text{(Assignment)}\\
[\![\textstyle\int \alpha]\!]_{|\psi\rangle\lambda} &= \mu_{|\psi\rangle}(\mathcal{E}(\alpha)) && \text{(Probability map)}\\
[\![\mathsf{Re}(|\top\rangle_A)]\!]_{|\psi\rangle\lambda} &= \mathsf{Re}(\langle v_A|\psi\rangle) && \text{(Real part)}\\
[\![\mathsf{Im}(|\top\rangle_A)]\!]_{|\psi\rangle\lambda} &= \mathsf{Im}(\langle v_A|\psi\rangle) && \text{(Imaginary part)}
\end{aligned}$$

Focussing on the *probability map*, given a classical formula α the writing $\mathcal{E}(\alpha)$ denotes its *extent*, that is:

$$\mathcal{E}(\alpha) = \{v \in 2^{\mathsf{qB}} : v \Vdash_c \alpha\}$$

where $v \Vdash_c \alpha$ stands for "valuation v satisfies α". Then the probability map is a function[3] $\mu_{|\psi\rangle} : 2^{2^{\mathsf{qB}}} \to \mathbb{R}$:

$$\mu_{|\psi\rangle}(U) = \sum_{v_A \in U} \parallel \langle v_A|\psi\rangle \parallel^2$$

[3]Here [4] defines the function domain as 2^{qB}, though the function takes in input a set of sets of qubits (as is indeed the case of $\mathcal{E}(\alpha)$).

When evaluated on $\mathcal{E}(\alpha)$ this function returns the probability of valuation $|\psi\rangle$ of satisfying α. This is better explained through an example. Suppose $\mathsf{qB} = \{q_1, q_2\}$, $\alpha = q_1 \vee q_2$ and one wants to check $|\psi\rangle = \frac{|01\rangle+|00\rangle}{\sqrt{2}}$ against α.

Then, $\mathcal{E}(\alpha) = \big\{\{q_1\}, \{q_2\}, \{q_1, q_1\}\big\}$. Finally, one obtains:

$$\begin{aligned}\mu_{|\psi\rangle}(\mathcal{E}(\alpha)) &= \| \langle v_{q_1}|\psi\rangle \|^2 + \| \langle v_{q_2}|\psi\rangle \|^2 + \| \langle v_{q_1,q_2}|\psi\rangle \|^2 \\ &= \| \langle 10|\psi\rangle \|^2 + \| \langle 01|\psi\rangle \|^2 + \| \langle 11|\psi\rangle \|^2 = 0 + \frac{1}{2} + 0 = \frac{1}{2}\end{aligned}$$

Finally, the meanings of the terms $\mathsf{Re}(|\top\rangle_A), \mathsf{Im}(|\top\rangle_A)$ simply require to remember the previous definition of a valuation v_A given a subset $A \subseteq \mathsf{qB}$.

Turning now the attention to quantum formulae, their semantics are defined as follows:

$$\begin{aligned}&(|\psi\rangle, \lambda) \Vdash_d (t_1 \leq t_2) \iff [\![t_1]\!]_{(|\psi\rangle,\lambda)} \leq [\![t_2]\!]_{(|\psi\rangle,\lambda)} \\ &(|\psi\rangle, \lambda) \nVdash_d \perp\!\!\!\perp \\ &(|\psi\rangle, \lambda) \Vdash_d (\gamma_1 \sqsupset \gamma_2) \iff (|\psi\rangle, \lambda) \nVdash_d \gamma_1 \vee (|\psi\rangle, \lambda) \Vdash_d \gamma_2\end{aligned}$$

where symbol $\Vdash_d$ is used to replace $\Vdash_{dEQPL}$ used in [4] and denote the satisfaction pairing of a quantum formula. For the sake of a clearer representation, this report uses $(|\psi\rangle, \lambda)$ instead of $|\psi\rangle\, \lambda$.

Shorthands Following are some rather intuitive and useful shorthands (obvious one are omitted):

$$\begin{aligned}
\boxminus \gamma &\iff \gamma \sqsupset \perp\!\!\!\perp && \text{(Negation)} \\
\gamma_1 \sqcup \gamma_2 &\iff \boxminus \gamma_1 \sqsupset \gamma_2 && \text{(Disjunction)} \\
\gamma_1 \sqcap \gamma_2 &\iff \boxminus (\boxminus \gamma_1 \sqcup \boxminus \gamma_2) && \text{(Conjunction)} \\
\gamma_1 \equiv \gamma_2 &\iff (\gamma_1 \sqsupset \gamma_2) \sqcap (\gamma_2 \sqsupset \gamma_1) && \text{(Equivalence)}
\end{aligned}$$

Finally, given a classical formula α, $\Box\alpha \iff \int \alpha = 1$.

Model Checking for dEQPL In [4] a Model Checking algorithm is provided for *closed formulae*, that is, formulae without variables.

In this setting, the Model Checking problem for dEQPL can be described as follows: given a dEQPL formula γ with alphabet $\mathsf{qB} = \{q_1, \ldots, q_n\}$ and a state vector $|\psi\rangle \in \mathcal{H}_{\mathsf{qB}}$, check whether $|\psi\rangle \Vdash_d \gamma$. To this end, $|\psi\rangle$ is modeled by an array ψ of 2^n pairs. Each pair contains the real and imaginary part of a logical amplitude with respect to a vector of the computational basis.

Algorithm 6.1 describes the procedure for dEQPL Model Checking. The procedure works recursively on the subformulae of γ. If γ is a term comparison formula (Line 10), it evaluates the two terms calling the procedure TERM_EVAL described by Algorithm 6.2. Although not shown, this procedure also adopts recursion for the trivial cases of arithmetic operations which, following [4] are assumed to take $\mathcal{O}(1)$ steps. Lines 6-9 retrieve respectively the real and imaginary part of the logical amplitude $\langle v_a|\psi\rangle$. Such procedure takes constant time

$\mathcal{O}(|A|)$ as it simply requires to access $|A|$ positions of ψ. Note that $\pi_i(\psi[j])$ with $i = 1, 2$ stands for the projection over the first or second element of j-th pair of ψ.

The most interesting part is given by Lines 4,5, where the term $\int \alpha$ is evalutated. For this subprocedure, one must first calculate the extent of α: $\mathcal{E}(\alpha)$. This requires to iterate over all possible 2^n *"pure valuations"*. The following step is the calculation of $\mu_{|\psi\rangle}$, which only takes a constant amount of steps for each $v \in \mathcal{E}(\alpha)$. This allows to conclude that the evaluation of $\int \alpha$ takes $\mathcal{O}(2^n)$ steps.

Given $|\gamma|$ the length of the dEQPL formula γ, in the worst case scenario the evaluation of all terms takes $\mathcal{O}(|\gamma| \cdot 2^n)$ steps. Once all terms have been evaluated, the evaluation of the quantum connectives takes $\mathcal{O}(|\gamma|)$.

This allows to conclude that Model Checking on dEQPL has complexity $\mathcal{O}(|\gamma| + |\gamma| \cdot 2^n) = \mathcal{O}(|\gamma| \cdot 2^n)$.

Algorithm 6.1 dEQPL Model Checking

```
1: Input: quantum formula γ, array ψ
2: Output: |ψ⟩ ⊩_d γ
3: procedure dEQPL_MC(γ, ψ)
4:     if φ = γ1 ⊐ γ2 then
5:         if ¬ dEQPL_MC(γ1, ψ) or dEQPL_MC(γ2, ψ) then
6:             return true
7:         return false
8:     if φ = ⫫ then return false
9:     if φ = t1 ≤ t2 then
10:        if TERM_VAL(t1, ψ) ≤ TERM_VAL(t2, ψ) then
11:            return true
12:        return false
```

Algorithm 6.2 dEQPL Term Evaluation

```
1: Input: term t, array ψ
2: Output: value of t
3: procedure TERM_EVAL(t, ψ)
4:     if t = ∫ α then
5:         return  μ_{|ψ⟩}(E(α))
6:     if t = Re(|⊤⟩_A) then                          ▷ A ⊆ qB
7:         return  Σ_{q_i ∈ A} π1(ψ[i])
8:     if t = Im(|⊤⟩_A) then                          ▷ A ⊆ qB
9:         return  Σ_{q_i ∈ A} π2(ψ[i])
10:    ...
```

Concluding on dEQPL, a sound and weakly complete axiomatization can be provided. For these steps the reader is referred to [4, Section 2.3].

6.2 Quantum Computational Tree Logic

Syntax The syntax of QCTL can be easily defined by enriching dEQPL with path quantifiers (A, E) and state quantifiers ($\mathsf{F}, \mathsf{G}, \mathsf{U}$) following the syntactic constraints of CTL (Section 3.1). Following, an abbreviated and recursive definition of QCTL syntax:

$$\theta \coloneqq \gamma \parallel \theta \sqsupset \theta \parallel \mathsf{EX}\theta \parallel \mathsf{AF}\theta \parallel \mathsf{E}[\theta \mathsf{U} \theta]$$

where γ is a dEQPL formula.

Semantics Whereas CTL formulae are interpreted over finite Kripke structures, for QCTL, a model for a quantum transition system is still do be defined. This requirement is perfectly met by *finite quantum Kripke structures*.

Definition 6.1. Given a finite set of qubits qB and a set of variables $\mathcal{X} \subseteq \mathbb{R}^{|\mathcal{X}|}$, a quantum Kripke structure is a pair $\mathcal{T} = (S, R)$ where:

- $S \subset \mathcal{H}_{\mathsf{qB}} \times \mathbb{R}^{|\mathcal{X}|}$ is the set of *states*. Each state is a pair $s = (|\psi\rangle, \lambda)$ where $|\psi\rangle$ is a unit vector in $\mathcal{H}_{\mathsf{qB}}$ and λ an assignment $\mathcal{X} \to \mathbb{R}^{|\mathcal{X}|}$
- $R \subseteq S \times S$ is a *transition relation* such that for all $(|\psi\rangle, \lambda) \in S$, there exists $(|\psi\rangle', \lambda') \in S$ such that $\big((|\psi\rangle, \lambda), (|\psi\rangle', \lambda')\big) \in R$.

$\mathcal{T}$ is said to be *finite* if S is finite. Moreover, the infinite sequence $\pi = s_1 s_2 \dots$ is said to be a *path* of $\mathcal{T}$ starting from s_1 if $(s_i, s_{i+1}) \in R$ for all $i \geq 1$.

Notice how this definition provides no explicit representation for a labelling function. The reason being that the component $|\psi\rangle$ of a state represents itself a valuation of all the propositionl symbols (qubits), as previously shown.

A QCTL formula can thus be interpreted over a quantum Kripke structure $\mathcal{T} = (S, R)$ and a state $s_i = |\psi\rangle, \lambda \in S$. Following, the semantics of QCTL:

$$\begin{aligned}
&\mathcal{T}, s_i \Vdash_{\mathsf{QCTL}} \gamma && \iff && s \Vdash_d \gamma \\
&\mathcal{T}, s_i \Vdash_{\mathsf{QCTL}} \theta_1 \sqsupset \theta_2 && \iff && \mathcal{T}, s_i \nVdash_{\mathsf{QCTL}} \theta_1 \vee \mathcal{T}, s \Vdash_{\mathsf{QCTL}} \theta_2 \\
&\mathcal{T}, s_i \Vdash_{\mathsf{QCTL}} \mathsf{EX}\theta && \iff && \exists\, s' \in S,\ (s_i, s') \in R \text{ and } \mathcal{T}, s' \Vdash_{\mathsf{QCTL}} \theta \\
&\mathcal{T}, s_i \Vdash_{\mathsf{QCTL}} \mathsf{AF}\theta && \iff && \forall \pi = s_i s_{i+1} s_{i+2} \dots, \exists\, j \geq i, (\mathcal{T}, s_j \Vdash_{\mathsf{QCTL}} \theta) \\
&\mathcal{T}, s_i \Vdash_{\mathsf{QCTL}} \mathsf{E}[\theta_1 \,\mathsf{U}\, \theta_2] && \iff && \exists \pi = s_i s_{i+1} s_{i+2} \dots, \exists\, j \geq i\, (\mathcal{T}, s_j \Vdash_{\mathsf{QCTL}} \theta_2, \\
& && && \forall\, k,\, i \leq k < j,\ \mathcal{T}, s_k \Vdash_{\mathsf{QCTL}} \theta_1)
\end{aligned}$$

One immediately notices how temporal modalities extend dEQPL in the same way they extend classical propositional logic.

Model Checking for QCTL [4] provides the main idea of an algorithm for Model Checking on closed QCTL formulae. This report shall try to elaborate on this idea applying the concepts learned during the lectures, with the goal of providing a slightly more rigorous illustration of the problem.

The idea follows the *explicit-state* approach mentioned in Section 3.1. That is, given a QCTL closed formula θ and a finite quantum Kripke structure $\mathcal{T} = (S, R)$ compute the set of all states of $\mathcal{T}$ that satisfy θ:

$$Sat_{\mathcal{T}}(\theta) \coloneqq \{|\psi\rangle \in S : \mathcal{T}, |\psi\rangle \Vdash_{\mathsf{QCTL}} \theta\}$$

Note that, being in the context of closed formulae, states are now only represented by state vectors in $\mathcal{H}_{\mathsf{qb}}$.

Having said that, the authors in [4] draw inspiration from *symbolic* Model-Checking, proposing a fixpoint characterization of temporal formulae. Adopting such approach in the quantum case requires to organize the elements of $\wp(S)$ via set inclusion ordering, obtaining the complete lattice $(\wp(S), \subseteq)$. The top and bottom of the complete lattice are given respectively by the QCTL formulae $\top\!\!\top$ and $\bot\!\!\bot$. QCTL formulae can thus be represented by sets of states given by least or greatest fixpoints of monotonic predicate transformers $\tau : \wp(S) \to \wp(S)$. Any such transformer τ is proven to have a least and greatest fixpoint over $\wp(S)$. Being S finite, τ can also be proven to be $\cup$-continuous and $\cap$-continuous, these results allow respectively for the following representations:

$$\mu Z.\tau(Z) = \bigcup_i \tau^i(\bot\!\!\bot) \qquad \text{(Least fixpoint)}$$

$$\nu Z.\tau(Z) = \bigcap_i \tau^i(\top\!\!\top) \qquad \text{(Greatest fixpoint)}$$

where $\tau^i(Z)$ denotes i applications of τ. In these conditions, both least and greatest fixpoints are proven to be computable in $\mathcal{O}(S)$.

Having said that, the set $Sat_{\mathcal{T}}$ of a closed QCTL formula can be defined as follows:

$$\begin{aligned}
Sat_{\mathcal{T}}(\gamma) &= \{|\psi\rangle \in S : \psi \Vdash_{\mathsf{QCTL}} \gamma\} \\
Sat_{\mathcal{T}}(\theta_1 \sqsupset \theta_2) &= \big(S \setminus Sat_{\mathcal{T}}(\theta_1)\big) \cup Sat_{\mathcal{T}}(\theta_2) \\
Sat_{\mathcal{T}}(\mathsf{EX}\,\theta) &= \{|\psi\rangle \in S : \exists\, |\psi'\rangle \left((|\psi\rangle, |\psi'\rangle) \in R \wedge |\psi'\rangle \in Sat_{\mathcal{T}}(\theta)\right)\} \\
Sat_{\mathcal{T}}(\mathsf{AF}\,\theta) &= \mu Z.(\theta \vee \mathsf{AX} Z) \\
Sat_{\mathcal{T}}(\mathsf{E}[\theta_1 \mathsf{U} \theta_2]) &= \mu Z.(\theta_2 \vee (\theta_1 \wedge \mathsf{EX} Z))
\end{aligned}$$

The Model Checking algorithms for CTL and QCTL are thus analogous. Motivated by such similarity, the authors of [4] then proceed to analyse the complexity of the algorithm starting from the complexity of Model Checking on CTL: $\mathcal{O}(|\theta| \cdot (|S| + |R|))$. The key difference between the two algorithms is that whereas a classical atom is checked in constant time, a quantum atom γ requires time $\mathcal{O}(|\gamma| \cdot 2^n)$ where n is the number of qubits of the quantum system (see Section 6.1). Putting these two results together one obtains $\mathcal{O}(|\theta|^2 \cdot (|S| + |R|) \cdot 2^n)$ as the complexity of Model Checking on QCTL.

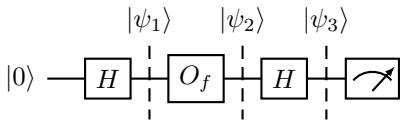
On a last note, the case of the simulation of quantum Kripke structures on classical machinery deserves to be mentioned. The algorithm has just been shown to take polynomial time with respect to the dimension of the quantum

Kripke structure. Though to classically simulate such quantum Kripke structure, exponential space with respect to the number of qubits is required. The added space is needed to encode all possible 2^{2^n} state superpositions.

7 Case Study: Deutsch's Algorithm

The authors of [1] provide two useful applications of the quantum verification techniques just described. Example 3.1 and Example 3.3 of [1] apply respectively the quantum weakest precondition calculus and QCTL to analyze Deutsch's algorithm.

Suppose f to be a function such that $f : \{0,1\} \to \{0,1\}$. Deutsch's algorithm computes $f(0) \oplus f(1)$ in a single computation exploiting quantum parallelism. For clarity, the quantum circuit for the algorithm is here described:



Note that this is a simplified variant using a single qubit rather than two. H denotes a Hadamard gate. O_f is a quantum oracle implementing a call to function f. The last gate consists of a measurement $M = \{ |0\rangle\langle 0|, |1\rangle\langle 1| \}$ over the computational basis. A matrix representation of the gates can be given:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}; \quad O_f = \begin{pmatrix} (-1)^{f(0)} & 0 \\ 0 & (-1)^{f(1)} \end{pmatrix}$$

Finally, the labels $|\psi_i\rangle$ represent the state over different stages of the circuit:

$$|\psi_1\rangle = H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$
$$|\psi_2\rangle = O_f|\psi_1\rangle = \frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}$$

Reasoning on $|\psi_2\rangle$, one notices two possible outcomes based on the behavior of f, thus proceeding as follows:

$$|\psi_2\rangle = \begin{cases} \pm\frac{|0\rangle+|1\rangle}{\sqrt{2}}, \text{ if } f(0) = f(1) \\ \pm\frac{|0\rangle-|1\rangle}{\sqrt{2}}, \text{ if } f(0) \neq f(1) \end{cases} ; |\psi_3\rangle = H|\psi_2\rangle = \begin{cases} \pm|0\rangle, \text{ if } f(0) = f(1) \\ \pm|1\rangle, \text{ if } f(0) \neq f(1) \end{cases}$$

Thus the last measurement will give outcome 0 if $f(0) \oplus f(1) = 0$ and 1 otherwise.

Quantum Weakest Precondition Deutsch's algorithm can be translated into the quantum while-language as follows:

$$Deutsch \equiv [q := 0;\ q := Hq;\ q := O_f q;\ q := Hq;\ \texttt{measure}\, M[q] : \texttt{skip};\ \texttt{skip}]$$

The postcondition is defined as $Post = (1-f(0)\oplus f(1))\,|0\rangle\langle 0| + f(0)\oplus f(1)\,|1\rangle\langle 1|$, coherently with the results just shown: if upon termination $f(0)\oplus f(1) = 0$ then state $|0\rangle\langle 0|$ should be reached. For short consider $c = 1 - f(0)\oplus f(1)$, $b = f(0)\oplus f(1)$. Now, function qwlp as defined in Section 5.3 is computed running backwards through the statements of *Deutsch*. Note that, in absence of `while` statements, $\mathsf{qwlp} = \mathsf{qwp}$.

$$\begin{aligned}&\mathsf{qwp}.[\texttt{measure}\,M[q] : \texttt{skip}; \texttt{skip}](Post) = M_0^\dagger Post M_0 + M_1^\dagger Post M_1\\ &= |0\rangle\langle 0|\,c\,|0\rangle\langle 0|0\rangle\langle 0| + |1\rangle\langle 1|\,b\,|1\rangle\langle 1|1\rangle\,|1\rangle\\ &= c\,|0\rangle\langle 0| + b\,|1\rangle\langle 1| = Post\end{aligned}$$

The obtained Hoare triple $\{Post\}\texttt{measure}\,M[q] : \texttt{skip}; \texttt{skip}\{Post\}$ is obviously correct. Then:

$$\begin{aligned}&\mathsf{qwp}.[q \coloneqq Hq](Post) = H^\dagger Post H = HPostH\\ &= Hc\,|0\rangle\langle 0|\,H + Hb\,|1\rangle\langle 1|\,H\\ &= c\,|+\rangle\langle +| + b\,|-\rangle\langle -| = Post_1\end{aligned}$$

where $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Proceeding:

$$\begin{aligned}&\mathsf{qwp}.[q \coloneqq O_f q](Post_1) = O_f^\dagger Post_1 O_f = O_f Post_1 O_f\\ &= \frac{|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| + |1\rangle\langle 1|}{2} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}\frac{\langle 0| + \langle 1|}{\sqrt{2}} = |+\rangle\langle +| = Post_2\end{aligned}$$

The calculations for this last step are simple though quite tedious and have thus been skipped. Now, onto the first Hadamard application:

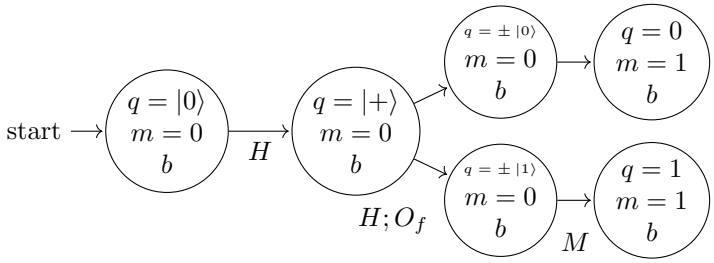
$$\mathsf{qwp}.[q \coloneqq Hq](Post_2) = H^\dagger\,|+\rangle\langle +|\,H = |0\rangle\langle 0| = Post_3$$

And finally, onto the initialization:

$$\begin{aligned}&\mathsf{qwp}.[q \coloneqq 0](Post_3) = |0\rangle_q\langle 0|\,Post_3\,|0\rangle_q\langle 0| + |1\rangle_q\langle 0|\,Post_3\,|0\rangle_q\langle 1|\\ &= |0\rangle_q\langle 0|0\rangle_q\langle 0|0\rangle_q\langle 0| + |1\rangle_q\langle 0|0\rangle\langle 0|0\rangle_q\langle 1|\\ &= |0\rangle\langle 0| + |1\rangle\langle 1| = I\end{aligned}$$

Using definition 5) of function qwp one obtains $\mathsf{qwp}.[Deutsch](Post) = I$ and, consequently, the Hoare triple $\{I\}\,Deutsch\,\{Post\}$. Quite interestingly, *Rule Consequentiality* from Eq. 11 allows to replace I with any precondition P, since for any P, $P \sqsubseteq I$. This leads to the conclusion that *Deutsch's algorithm is correct no matter the assumptions on which it is executed.*

QCTL Following the example, here is an attempt in providing the quantum Kripke structure for Deutsch's algorithm:



Here, b and m are classical bits such that $b = f(0) \oplus f(1)$ and m takes value 1 or 0 depending on whether q has been measured or not. Now consider the QCTL formula proposed in [1]:

$$\theta = \mathsf{A}\big[(\boxminus(\Box m))\mathsf{U}\big(\Box m \sqcap (\Box b \equiv (\int q = 1))\big)\big]$$

Which says: qubit q is never measured *until* it is measured ($\Box(m)$) and $b = 1$ if and only if q is measured to 1. Adding the temporal operator EX fits the model specification just given. From the graphical representation, it is straightforward to notice how the initial state of the quantum Kripke structure satisfies the property.

8 Conclusion

As previously mentioned, this report covers only part of the topics discussed in [1]. Among the other techniques are *Path sums* and the *ZX-Calculus*.

Path sums represent a characterization of unitary operators. In [6], path sums were used to verify Clifford group circuits. A Clifford group is a set of gates which is universal under some specific extensions. The path sum come in useful as classical mathematical functions can be applied to them, and thus, be verified. On the other hand, the ZX-Calculus [7] provides a graphical framework for the verification of quantum circuits. This technique is used to simplify the circuits and checking their equivalence.

Bringing back the attention on the techniques presented earlier, effort has been put to extend these ideas. As happened in the classical case, theorem provers have been developed to automatize quantum deductive verification. For instance, in [8] an implementation of Quantum Hoare Logic was proposed for the proof assistant Isabelle/HOL. Although only partially automatic, the implementation (known as QHLProver) was successful in proving the correctness Grover's algorithm and soundness/relative completeness of qPD (see Eq. 11). Having said that, effort has been put to further develop QHL itself. To this day, one of its main weaknesses concerns the exponential increase in dimension of quantum predicates with respect to the number of qubits.

Moving to Quantum CTL, as mentioned in [4], future work could focus on developing a more general formalism. As of right now, QCTL is restricted to measurements over the computational basis, which is a rather limiting assumption. On the other hand, Model Checking for QCTL deserves a more rigorous

analysis and a more explicit definition of its algorithms. Finally, what could be understood as the *quantum state explosion problem* should be addressed. The problem becomes even more daunting than in the classical case. Whereas the classical problem is concerned with cartesian products of sets of states, the quantum case has to deal with tensor products of entire Hilbert spaces [9]. Mitigation techniques for the quantum case are still to be developed.

References

[1] Marco Lewis, Sadegh Soudjani, and Paolo Zuliani. "Formal Verification of Quantum Programs: Theory, Tools and Challenges". In: *CoRR* abs/2110.01320 (2021). arXiv: 2110.01320. URL: https://arxiv.org/abs/2110.01320.

[2] Mingsheng Ying. "Floyd–Hoare Logic for Quantum Programs". In: *ACM Trans. Program. Lang. Syst.* 33.6 (Jan. 2012). ISSN: 0164-0925. DOI: 10.1145/2049706.2049708. URL: https://doi.org/10.1145/2049706.2049708.

[3] Ellie D'Hondt and Prakash Panangaden. "Quantum weakest preconditions". In: *Mathematical Structures in Computer Science* 16.3 (2006), pp. 429–451. DOI: 10.1017/S0960129506005251.

[4] P. Baltazar, R. Chadha, and P. Mateus. "Quantum Computation Tree Logic — Model checking and complete calculus". In: *International Journal of Quantum Information* 06.02 (2008), pp. 219–236. DOI: 10.1142/S0219749908003530. URL: https://doi.org/10.1142/S0219749908003530.

[5] Peter Selinger. "Towards a quantum programming language". In: *Mathematical Structures in Computer Science* 14.4 (2004), pp. 527–586. DOI: 10.1017/S0960129504004256.

[6] Matthew Amy. "Towards Large-scale Functional Verification of Universal Quantum Circuits". In: *Electronic Proceedings in Theoretical Computer Science* 287 (Jan. 2019), pp. 1–21. ISSN: 2075-2180. DOI: 10.4204/eptcs.287.1. URL: http://dx.doi.org/10.4204/EPTCS.287.1.

[7] Bob Coecke and Ross Duncan. "Interacting quantum observables: categorical algebra and diagrammatics". In: *New Journal of Physics* 13.4 (Apr. 2011), p. 043016. ISSN: 1367-2630. DOI: 10.1088/1367-2630/13/4/043016. URL: http://dx.doi.org/10.1088/1367-2630/13/4/043016.

[8] Junyi Liu et al. "Formal Verification of Quantum Algorithms Using Quantum Hoare Logic". In: *Computer Aided Verification*. Ed. by Isil Dillig and Serdar Tasiran. Cham: Springer International Publishing, 2019, pp. 187–207. ISBN: 978-3-030-25543-5.

[9] Mingsheng Ying and Yuan Feng. *Model Checking Quantum Systems: Principles and Algorithms*. Cambridge University Press, 2021. Chap. 7. DOI: 10.1017/9781108613323.