Formal Semantics of a Classical-Quantum Language

Yuxin Deng[a,*], Yuan Feng[b]

[a] *Shanghai Key Laboratory of Trustworthy Computing,*
MOE International Joint Lab of Trustworthy Software,
and International Research Center of Trustworthy Software,
East China Normal University
[b] *University of Technology Sydney, Australia*

Abstract

We investigate the formal semantics of a simple imperative language that has both classical and quantum constructs. More specifically, we provide an operational semantics, a denotational semantics and two Hoare-style proof systems: an abstract one and a concrete one. The two proof systems are satisfaction-based, as inspired by the program logics of Barthe et al for probabilistic programs. The abstract proof system turns out to be sound and relatively complete, while the concrete one is sound only.

Keywords: Classical-quantum language, Formal semantics, Soundness, Completeness

1. Introduction

Programming is the core of software development, but it is also an inherently error-prone activity. The likelihood of errors will even be significantly higher when programming with a quantum computer, as the techniques used for classical programming are, unfortunately, hard to apply to quantum computers because quantum systems are essentially different from classical ones. Thus, there is a pressing need to provide verification and analysis techniques for reasoning about the correctness of quantum programs. Furthermore, these techniques would also be very useful for compiling and optimising quantum programs.

Among other techniques, Hoare logic [11] provides a syntax-oriented proof system to reason about program correctness. After decades of development, Hoare logic has been successfully applied in analysis of programs with non-determinism, recursion, parallel execution, etc [2, 1]. It was also extended to programming languages with probabilistic features. Remarkably, as the program states for probabilistic languages are (sub)distributions over evaluations of program variables, the extension naturally follows two different approaches, depending on how assertions of probabilistic states are defined. The first one takes subsets of distributions as (qualitative) assertions, similar to the non-probabilistic case, and the *satisfaction* relation between distributions and assertions is then just the ordinary membership [20, 7, 5, 3]. In contrast, the other approach takes non-negative functions on evaluations as (quantitative) assertions. Consequently, one is concerned with the *expectation* of a distribution satisfying an assertion [17, 16, 19, 13, 14].

[*] Corresponding author

Email addresses: `yxdeng@sei.ecnu.edu.cn` (Yuxin Deng), `Yuan.Feng@uts.edu.au` (Yuan Feng)

February 20, 2022

In the past two decades, several Hoare-type logic systems for quantum programs (QHL) have been proposed, also following the two approaches as in the probabilistic setting.

Expectation-based QHLs. In the logic systems proposed in [25, 26, 27, 15] for purely quantum programs, the assertions P and Q in a Hoare triple $\{P\}S\{Q\}$ are both positive operators with the eigenvalues lying in $[0, 1]$, and such a triple is valid in the sense of total correctness if for any initial quantum state ρ, $\mathrm{tr}(P\rho) \leq \mathrm{tr}(Q\rho')$ where ρ' is the final state obtained by executing S on ρ, and $\mathrm{tr}(P\rho)$ denotes the expectation/degree of ρ satisfying P (or physically, the average outcome when measuring ρ according to the projective measurement determined by P). This definition captures the idea that the precondition P (on the initial state) provides a lower bound on the degree of satisfaction of the postcondition Q (on the final state). More recently, this type of expectation-based QHL has been extended to quantum programs with classical variables [9] as well as distributed quantum programs with classical communication [8]. These logic systems have proven to be useful in describing and verifying correctness of a wide range of quantum algorithms such as Shor's algorithm [21], Grover's algorithm [10], etc. Moreover, they are theoretically elegant: all of these systems are (relatively) complete in the sense that every semantically valid Hoare triple can be deduced from the corresponding proof system. However, as logic operations such as conjunction and disjunction are difficult, if at all possible, to define for positive operators, complicated properties can only be analysed separately, making the verification process cumbersome. This has been pointed out in [3] for expectation-based probabilistic Hoare logics. The same is obviously true for expectation-based QHLs as well.

Satisfaction-based QHLs. An Ensemble Exogenous Quantum Propositional Logic (EEQPL) was proposed in [6] for a simple quantum language with bounded iteration. The assertions in EEQPL can access amplitudes of quantum states, which makes it very strong in expressiveness but also hinders its use in applications such as debugging, as amplitudes are not physically accessible through measurements. The completeness of EEQPL is only proven in a special case where all real and complex values involved range over a finite set. In contrast, the QHL proposed in [12] takes as the assertion language an extended first-order logic with the primitives of applying a matrix on a set of qubits and computing the probability that a classical predicate is satisfied by the outcome of a quantum measurement. The proof system is shown to be sound, but no completeness result was established.

Another way of defining satisfaction-based QHLs proposed in [28, 22] regards subspaces of the Hilbert space as assertions, and a quantum state ρ satisfies an assertion P iff the support (the image space of linear operators) of ρ is included in P. The subspace assertion makes it easy to describe and determine properties of quantum programs, but the expressive power of the assertions is limited: they only assert if a given quantum state lies completely within a subspace. Consequently, quantum algorithms which succeed with certain probability cannot be verified in their logic systems.

Contribution of the current paper. In this paper, motivated by [3], we propose two Hoare-style proof systems: an abstract one and a concrete one. It is worth noting that the imperative language we consider here involves both classical and quantum constructs. Our work distinguishes itself from the works on QHLs mentioned above in the following aspects:

- *Assertion language*. The assertions used in our logic systems are boolean-typed, so that they can be easily combined using logic operations such as disjunction and conjunction. On the other hand, all information used in the assertions is physically accessible: it can be obtained through quantum measurement applying on the program states. For example, consider the protocol of superdense coding (see Section 6). In order to send a message of two bits stored in two variables x_0x_1, Alice actually sends a qubit to Bob. From the received quantum information Bob can recover the message y_0y_1. The property we would expect is $x_0 = y_0 \wedge x_1 = y_1$. Indeed, by letting SC be a quantum program to implement the protocol, we can prove that the following judgement is derivable:

$$\{\mathbf{true}\}\ SC\ \{\Box(x_0 = y_0 \wedge x_1 = y_1)\}$$

where the precondition **true** is satisfied by any program state. Intuitively, the judgement says that the message received by Bob is always the same as that sent by Alice, no matter what the initial program state is. There is no need to mention the concrete values of x_0 and x_1. This is a natural and concise way of specifying the correctness of the program SC.

- *Satisfaction-based complete QHL.* The existing satisfaction-based QHLs proposed in the literature all lack completeness; the only exception is [28], but as mentioned above, the assertions there are not expressive enough to verify probabilistic correctness. The abstract proof system proposed in the current work is shown to be sound and relatively complete, while the concrete proof system is sound only. By soundness, we mean that if the Hoare triple $\{P\}\ c\ \{Q\}$ is derivable, then for any program state μ that satisfies P, the program state $[\![c]\!]_\mu$ after the execution of command c always satisfies Q. Completeness means the converse. We only have relative completeness because our assertion language allows for first-order logic operators such as implication. Note that in the presence of both classical and quantum variables, we represent each program state μ as a partial density operator valued distribution (POVD), and interpret a program as a transformer of POVDs. We establish a consistence result between the denotational semantics and the small-step operational semantics based on POVDs.

The rest of the paper is structured as follows. In Section 2 we recall some basic notations from linear algebra and quantum mechanics. In Section 3 we define the syntax and operational semantics of a simple classical-quantum imperative language. In Section 4 we present an abstract proof system and show its soundness and relative completeness. In Section 5 we provide a concrete proof system. In Section 6 we use the example of superdense coding to illustrate the concrete proof system. Finally, we conclude in Section 7.

2. Preliminaries

We briefly recall some basic notations from linear algebra and quantum mechanics which are needed in this paper. For more details, we refer to [18].

A *Hilbert space* $\mathcal{H}$ is a complete vector space with an inner product $\langle\cdot|\cdot\rangle : \mathcal{H}\times\mathcal{H} \to \mathbf{C}$ such that

1. $\langle\psi|\psi\rangle \geq 0$ for any $|\psi\rangle \in \mathcal{H}$, with equality if and only if $|\psi\rangle = 0$;
2. $\langle\varphi|\psi\rangle = \langle\psi|\varphi\rangle^*$;
3. $\langle\varphi|\sum_i c_i|\psi_i\rangle = \sum_i c_i\langle\varphi|\psi_i\rangle$,

where $\mathbf{C}$ is the set of complex numbers, and for each $c \in \mathbf{C}$, c^* stands for the complex conjugate of c. For any vector $|\psi\rangle \in \mathcal{H}$, its length $\||\psi\rangle\|$ is defined to be $\sqrt{\langle\psi|\psi\rangle}$, and it is said to be *normalised* if $\||\psi\rangle\| = 1$. Two vectors $|\psi\rangle$ and $|\varphi\rangle$ are *orthogonal* if $\langle\psi|\varphi\rangle = 0$. An *orthonormal basis* of a Hilbert space $\mathcal{H}$ is a basis $\{|i\rangle\}$ where each $|i\rangle$ is normalised and any pair of them are orthogonal.

Let $\mathcal{L}(\mathcal{H})$ be the set of linear operators on $\mathcal{H}$. For any $A \in \mathcal{L}(\mathcal{H})$, A is *Hermitian* if $A^\dagger = A$ where $A^\dagger$ is the adjoint operator of A such that $\langle\psi|A^\dagger|\varphi\rangle = \langle\varphi|A|\psi\rangle^*$ for any $|\psi\rangle, |\varphi\rangle \in \mathcal{H}$. A linear operator $A \in \mathcal{L}(\mathcal{H})$ is *unitary* if $A^\dagger A = AA^\dagger = I_\mathcal{H}$ where $I_\mathcal{H}$ is the identity operator on $\mathcal{H}$. The *trace* of A is defined as $\text{tr}(A) = \sum_i\langle i|A|i\rangle$ for some given orthonormal basis $\{|i\rangle\}$ of $\mathcal{H}$. A linear operator

$A \in \mathcal{L}(\mathcal{H})$ is *positive* if $\langle\varphi|A|\varphi\rangle \geq 0$ for any state $|\varphi\rangle \in \mathcal{H}$. The *Löwner order* $\sqsubseteq$ on the set of Hermitian operators on $\mathcal{H}$ is defined by letting $A \sqsubseteq B$ iff $B - A$ is positive.

Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two Hilbert spaces. Their *tensor product* $\mathcal{H}_1 \otimes \mathcal{H}_2$ is defined as a vector space consisting of linear combinations of the vectors $|\psi_1\psi_2\rangle = |\psi_1\rangle|\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$ with $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$. Here the tensor product of two vectors is defined by a new vector such that

$$\left(\sum_i \lambda_i|\psi_i\rangle\right) \otimes \left(\sum_j \mu_j|\varphi_j\rangle\right) = \sum_{i,j} \lambda_i\mu_j|\psi_i\rangle \otimes |\varphi_j\rangle.$$

Then $\mathcal{H}_1 \otimes \mathcal{H}_2$ is also a Hilbert space where the inner product is defined as the following: for any $|\psi_1\rangle, |\varphi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle, |\varphi_2\rangle \in \mathcal{H}_2$,

$$\langle\psi_1 \otimes \psi_2|\varphi_1 \otimes \varphi_2\rangle = \langle\psi_1|\varphi_1\rangle_{\mathcal{H}_1}\langle\psi_2|\varphi_2\rangle_{\mathcal{H}_2}$$

where $\langle\cdot|\cdot\rangle_{\mathcal{H}_i}$ is the inner product of $\mathcal{H}_i$.

By applying quantum gates to qubits, we can change their states. For example, the Hadamard gate (H gate) can be applied on a single qubit, while the CNOT gate can be applied on two qubits. Some commonly used gates and their representation in terms of matrices are as follows.

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

According to von Neumann's formalism of quantum mechanics [23], an isolated physical system is associated with a Hilbert space which is called the *state space* of the system. A *pure state* of a quantum system is a normalised vector in its state space, and a *mixed state* is represented by a density operator on the state space. Here a *density operator* ρ on Hilbert space $\mathcal{H}$ is a positive linear operator such that $\mathrm{tr}(\rho) = 1$. A *partial density operator* ρ is a positive linear operator with $\mathrm{tr}(\rho) \leq 1$.

The evolution of a closed quantum system is described by a unitary operator on its state space: if the states of the system at times t_1 and t_2 are ρ_1 and ρ_2, respectively, then $\rho_2 = U\rho_1U^\dagger$ for some unitary operator U which depends only on t_1 and t_2.

A quantum *measurement* is described by a collection $\{M_m\}$ of measurement operators, where the indices m refer to the measurement outcomes. It is required that the measurement operators satisfy the completeness equation $\sum_m M_m^\dagger M_m = I_{\mathcal{H}}$. If the system is in state ρ, then the probability that measurement result m occurs is given by

$$p(m) = \mathrm{tr}(M_m^\dagger M_m\rho),$$

and the state of the post-measurement system is $M_m\rho M_m^\dagger/p(m)$.

3. QIMP

We define the syntax and operational semantics of a simple classical-quantum imperative language called **QIMP**. The language is essentially extended from **IMP** [24] by adding quantum data and a few operations for manipulating quantum data.

3.1. Syntax

We assume three types of data in our language: `Bool` for booleans, `Int` for integers, and qubits `Qbt` for quantum data. Let $\mathbb{Z}$ be the set of constant integer numbers, ranged over by n. Let **Cvar**, ranged over by $x, y, ...$, be the set of classical variables, and **Qvar**, ranged over by $q, q', ...$, the set of quantum variables. It is assumed that both **Cvar** and **Qvar** are countably infinite. We assume a set **Aexp** of arithmetic expressions over `Int`, which includes **Cvar** as a subset and is ranged over by $a, a', ...$, and a set of boolean-valued expressions **Bexp**, ranged over by $b, b', ...$, with the usual boolean constants **true**, **false** and boolean operators $\neg, \wedge, \vee$. In particular, we let $a = a'$ and $a \leq a'$ be boolean expressions for any $a, a' \in$ **Aexp**. We further assume that only classical variables can occur free in both arithmetic and boolean expressions.

We let U range over unitary operators, which can be user-defined matrices or built in if the language is implemented. For example, a concrete U could be the 1-qubit Hadamard operator H, or the 2-qubit controlled-NOT operator *CNOT*, etc. Similarly, we write M for the measurement described by a collection $\{M_i\}$ of measurement operators, with each index i representing a measurement outcome. For example, to describe the measurement of the qubit referred to by variable q in the computational basis, we can write $M := \{M_0, M_1\}$, where $M_0 = |0\rangle_q\langle 0|$ and $M_1 = |1\rangle_q\langle 1|$.

Sometimes we use metavariables which are primed or subscripted, e.g. x', x_0 for classical variables. We abbreviate a tuple of quantum variables $\langle q_1, ..., q_n\rangle$ as $\bar{q}$ if the length n of the tuple is not important. The formation rules for arithmetic and boolean expressions as well as commands are defined by the following grammar.

- For **Aexp**: $a ::= n \mid x \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$
- For **Bexp**: $b ::= \textbf{true} \mid \textbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \wedge b_1 \mid b_0 \vee b_1$
- For **Com**:

$$\begin{array}{rl} c ::= & \textbf{skip} \mid x := a \mid c_0; c_1 \mid \textbf{if}\ b\ \textbf{then}\ c_0\ \textbf{else}\ c_1 \mid \textbf{while}\ b\ \textbf{do}\ c \\ & \mid q := |0\rangle \mid U[\bar{q}] \mid x := M[\bar{q}] \end{array}$$

An arithmetic expression can be an integer, a variable, or built from other arithmetic expressions by addition, subtraction, or multiplication. A boolean expression can be formed by comparing arithmetic expressions or by using the usual boolean operators. A command can be a skip statement, a classical assignment, a conditional statement, or a while-loop, as in many classical imperative languages. In addition, there are three commands that involve quantum data. The command $q := |0\rangle$ initialises the qubit referred to by variable q to be the basis state $|0\rangle$. The command $U[\bar{q}]$ applies the unitary operator U to the quantum system referred to by $\bar{q}$. The command $x := M[\bar{q}]$ performs a measurement M on $\bar{q}$ and assigns the measurement outcome to x. It differs from a classical assignment because the measurement M may change the quantum state of $\bar{q}$, besides the fact that the value of x is updated.

3.2. Operational Semantics

Since the execution of a **QIMP** program may involve both classical and quantum data, we consider the setting where the CPU that processes the program has two registers: one stores classical data and the other quantum data. Therefore, we will model a machine state as a pair composed of a classical state and a quantum state.

The notion of classical state is standard. Formally, a *classical state* is a function σ : **Cvar** $\rightarrow \mathbb{Z}$ from classical variables to integers. Thus $\sigma(x)$ is the value of variable x in state σ. The notion of quantum state is slightly more complicated. For each quantum variable $q \in \mathbf{Qvar}$, we assume a 2-dimensional Hilbert space $\mathcal{H}_q$ to be the state space of the q-system. For any finite subset V of **Qvar**, we denote

$$\mathcal{H}_V = \bigotimes_{q \in V} \mathcal{H}_q.$$

That is, $\mathcal{H}_V$ is the tensor product of the individual state spaces of all the quantum variables in V. Throughout the paper, when we refer to a subset of **Qvar**, it is always assumed to be finite. Given $V \subseteq \mathbf{Qvar}$, the set of *quantum states* consists of all partial density operators in the space $\mathcal{H}_V$, denoted by $\mathcal{D}^{-}(\mathcal{H}_V)$. A *machine state* is a pair $\langle \sigma, \rho \rangle$ where σ is a classical state and ρ a quantum state. In the presence of measurements, we often need to consider an ensemble of states. For that purpose, we introduce a notion of distribution.

Definition 3.1. Suppose $V \subseteq \mathbf{Qvar}$ and Σ is the set of classical states, i.e., the set of functions of type $\mathbf{Cvar} \to \mathbb{Z}$. A *partial density operator valued distribution (POVD)* is a function $\mu : \Sigma \to \mathcal{D}^{-}(\mathcal{H}_V)$ with $\sum_{\sigma \in \Sigma} \mathrm{tr}(\mu(\sigma)) \leq 1$.

Intuitively, a POVD μ represents a collection of machine states where each classical state σ is associated with a quantum state $\mu(\sigma)$. The notation of POVD is called classical-quantum state in [9]. If the collection has only one element σ, we explicitly write $(\sigma, \mu(\sigma))$ for μ. The support of μ, written $\lceil \mu \rceil$, is the set $\{\sigma \in \Sigma \mid \mu(\sigma) \neq 0\}$. We can also define the addition of two distributions by letting $(\mu_1 + \mu_2)(\sigma) = \mu_1(\sigma) + \mu_2(\sigma)$.

A *configuration* is a pair $\langle e, \sigma, \rho \rangle$, where e is an expression and (σ, ρ) is a POVD. We define the small-step operational semantics of arithmetic and boolean expressions as well as commands in a syntax-directed way by using an evaluation relation $\hookrightarrow$ between configurations. In Figure 1 we list the rules for evaluating integer variables, sums, and expressions of the form $a_0 \leq a_1$; the rules for other arithmetic and boolean expressions are similar. When evaluating an arithmetic or boolean expression, we only rely on the information from the given classical state, therefore we omit the quantum state in the configuration. This is not the case when we execute commands.

Let σ be a classical state and $n \in \mathbb{Z}$. We write $\sigma[n/x]$ for the updated state satisfying

$$\sigma[n/x](y) = \begin{cases} n & \text{if } y = x, \\ \sigma(y) & \text{if } y \neq x. \end{cases}$$

We are going to write $\to$ for the execution of commands. The transition rules are given in Figure 2. Here we introduce a special command **nil** that stands for a successful termination of programs. We follow [25] to define the operational semantics of quantum measurements in a non-deterministic way, and the probabilities of different branches are encoded in the quantum part of the configurations. For that reason we need to take partial density operators instead of the normalised density operators to represent quantum states. After the measurement M defined by some measurement operators M_i, the original state (σ, ρ) may evolve into a new state whose classical part is the updated state $\sigma[i/x]$ and the quantum part is the new quantum state $M_i \rho M_i^{\dagger}$. In all other rules, the execution of a command changes a configuration to another one. Among them, the rules for initialising qubits and unitary transformations only affect the quantum part of the original machine state. On the contrary, the commands for manipulating classical data only update the classical part of a state.

3.3. Denotational Semantics

For the purpose of presenting the denotational semantics, we add an **abort** command that halts the computation with no result. We interpret programs as POVD transformers. We write **POVD** for the set of POVDs called distribution states.

$$\frac{}{\langle x, \sigma\rangle \hookrightarrow \langle \sigma(x), \sigma\rangle}$$

$$\frac{\langle a_0, \sigma\rangle \hookrightarrow \langle a_0', \sigma\rangle}{\langle a_0 + a_1, \sigma\rangle \hookrightarrow \langle a_0' + a_1, \sigma\rangle} \qquad \frac{\langle a_1, \sigma\rangle \hookrightarrow \langle a_1', \sigma\rangle}{\langle n + a_1, \sigma\rangle \hookrightarrow \langle n + a_1', \sigma\rangle}$$

$$\frac{}{\langle n + m, \sigma\rangle \hookrightarrow \langle p, \sigma\rangle} \quad \text{if } p \text{ is the sum of } n \text{ and } m$$

$$\frac{\langle a_0, \sigma\rangle \hookrightarrow \langle a_0', \sigma\rangle}{\langle a_0 \leq a_1, \sigma\rangle \hookrightarrow \langle a_0' \leq a_1, \sigma\rangle} \qquad \frac{\langle a_1, \sigma\rangle \hookrightarrow \langle a_1', \sigma\rangle}{\langle n \leq a_1, \sigma\rangle \hookrightarrow \langle n \leq a_1', \sigma\rangle}$$

$$\frac{}{\langle n \leq m, \sigma\rangle \hookrightarrow \langle \mathbf{true}, \sigma\rangle} \quad \text{if } n \text{ is less than or equal to } m.$$

$$\frac{}{\langle n \leq m, \sigma\rangle \hookrightarrow \langle \mathbf{false}, \sigma\rangle} \quad \text{if } n \text{ is greater than } m.$$

Figure 1: Evaluation of arithmetic and boolean expressions (selected rules)

Lemma 3.2. We impose an order between POVDs by letting $\mu_1 \leq \mu_2$ if for any classical state σ we have $\mu_1(\sigma) \sqsubseteq \mu_2(\sigma)$, where $\sqsubseteq$ is the Löwner order. Let $(\mu_n)_{n\in\mathbb{N}} \in \mathbf{POVD}$ be an increasing sequence of POVDs. This sequence converges to some POVD μ_∞ and $\mu_n \leq \mu_\infty$ for any $n \in \mathbb{N}$.

Given an expression e, we denote its interpretation with respect to machine state (σ,ρ) by $[\![e]\!]_{(\sigma,\rho)}$. The denotational semantics of commands is displayed in Figure 3, where we omit the denotational semantics of arithmetic and boolean expressions such as $[\![a]\!]_\sigma$ and $[\![b]\!]_\sigma$, which is almost the same as in the classical setting because the quantum part plays no role for those expressions. This is an extension of the semantics for probabilistic programs presented in [3]. Instead of probabilistic assignments are measurements of quantum systems. A state evolves into a POVD after some quantum qubits are measured, with the measurement outcomes assigned to a classical variable. Two other quantum commands, initialisation of qubits and unitary operations, are deterministic and only affect the quantum part of a state. As usual, we define the semantics of a loop (**while** b **do** c) as the limit of its lower approximations, where the n-th lower approximation of $[\![\mathbf{while}\ b\ \mathbf{do}\ c]\!]_{(\sigma,\rho)}$ is $[\![(\mathbf{if}\ b\ \mathbf{then}\ c)^n; \mathbf{if}\ b\ \mathbf{then\ abort}]\!]_{(\sigma,\rho)}$, where (**if** b **then** c) is shorthand for (**if** b **then** c **else skip**) and c^n is the command c iterated n times with $c^0 \equiv \mathbf{skip}$. The limit exists because the sequence $([\![(\mathbf{if}\ b\ \mathbf{then}\ c)^n; \mathbf{if}\ b\ \mathbf{then\ abort}]\!]_{(\sigma,\rho)})_{n\in\mathbb{N}}$ is increasing and bounded. We write ε for the special POVD whose support is the empty set.

Proposition 3.3. The semantics $[\![c]\!]_{(\sigma,\rho)}$ of a command c in initial state (σ,ρ) is a POVD. The lifted semantics $[\![c]\!]_\mu$ of a command c in initial POVD μ is a POVD.

The operational and denotational semantics are related by the following theorem.

Theorem 3.4. For any command c and state (σ,ρ), we have

$$[\![c]\!]_{(\sigma,\rho)} = \sum_i \{(\sigma_i,\rho_i) \mid \langle c,\sigma,\rho\rangle \to^* \langle \mathbf{nil},\sigma_i,\rho_i\rangle\}\ .$$

Proof: See Appendix A. □

$$\frac{}{\langle \mathbf{skip}, \sigma, \rho\rangle \to \langle \mathbf{nil}, \sigma, \rho\rangle}$$

$$\frac{\langle a, \sigma\rangle \hookrightarrow \langle a', \sigma'\rangle}{\langle x := a, \sigma, \rho\rangle \to \langle x := a', \sigma', \rho\rangle} \qquad \frac{}{\langle x := n, \sigma, \rho\rangle \to \langle \mathbf{nil}, \sigma[n/x], \rho\rangle}$$

$$\frac{\langle c_0, \sigma, \rho\rangle \to \langle c_0', \sigma', \rho'\rangle}{\langle c_0; c_1, \sigma, \rho\rangle \to \langle c_0'; c_1, \sigma', \rho'\rangle} \qquad \frac{\langle c_1, \sigma, \rho\rangle \to \langle c_1', \sigma', \rho'\rangle}{\langle \mathbf{nil}; c_1, \sigma, \rho\rangle \to \langle c_1', \sigma', \rho'\rangle}$$

$$\frac{\langle b, \sigma\rangle \hookrightarrow \langle b', \sigma'\rangle}{\langle \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma, \rho\rangle \to \langle \mathbf{if}\ b'\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1, \sigma', \rho'\rangle}$$

$$\frac{}{\langle \mathbf{if\ true\ then}\ c_0\ \mathbf{else}\ c_1, \sigma, \rho\rangle \to \langle c_0, \sigma, \rho\rangle} \qquad \frac{}{\langle \mathbf{if\ false\ then}\ c_0\ \mathbf{else}\ c_1, \sigma, \rho\rangle \to \langle c_1, \sigma, \rho\rangle}$$

$$\frac{}{\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma, \rho\rangle \to \langle \mathbf{if}\ b\ \mathbf{then}\ (c; \mathbf{while}\ b\ \mathbf{do}\ c)\ \mathbf{else\ skip}, \sigma, \rho\rangle}$$

$$\frac{}{\langle q := |0\rangle, \sigma, \rho\rangle \to \langle \mathbf{nil}, \sigma, \rho'\rangle} \qquad \text{with } \rho' = |0\rangle_q\langle 0|\rho|0\rangle_q\langle 0| + |0\rangle_q\langle 1|\rho|1\rangle_q\langle 0|$$

$$\frac{}{\langle U[\bar{q}], \sigma, \rho\rangle \to \langle \mathbf{nil}, \sigma, U\rho U^\dagger\rangle}$$

$$\frac{M := \{M_i\}_{i\in I}}{\langle x := M[\bar{q}], \sigma, \rho\rangle \to \langle \mathbf{nil}, \sigma[i/x], M_i\rho M_i^\dagger\rangle}$$

Figure 2: Execution of commands

In [9] the denotational semantics is defined on top of the operational semantics by turning Theorem 3.4 into a definition; then the definition in Figure 3 becomes a property of the denotational semantics. In the current work, the denotational semantics is defined independently from the operational semantics and their consistence is established in the above theorem. For the **while** command, our treatement of both semantics is slightly different from that in [9], hence we provide a detailed proof of Theorem 3.4, though it shares some similarity with the proof outline of Lemma 4.6 in [9].

4. An Abstract Proof System

In this section, we present an abstract proof system, where assertions are arbitrary predicates on POVDs. We show that the proof system is sound and relatively complete. Since the abstract proof system is stated at the semantic level, it is not directly usable in practice, for which we need a concrete proof system to be introduced in Section 5. However, it helps us to factorize the soundness proof of the concrete proof system.

Definition 4.1. The set **Assn** of assertions is defined as $\mathcal{P}(\mathbf{POVD})$, the powerset of **POVD**. We introduce some expressions for assertions in order to concisely specify the abstract proof system.

$$
\begin{array}{rcl}
\llbracket \mathbf{skip} \rrbracket_{(\sigma,\rho)} & = & (\sigma,\rho) \\
\llbracket \mathbf{abort} \rrbracket_{(\sigma,\rho)} & = & \varepsilon \\
\llbracket x := a \rrbracket_{(\sigma,\rho)} & = & (\sigma[\llbracket a \rrbracket_\sigma / x], \rho) \\
\llbracket c_0; c_1 \rrbracket_{(\sigma,\rho)} & = & \llbracket c_1 \rrbracket_{\llbracket c_0 \rrbracket_{(\sigma,\rho)}} \\
\llbracket \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1 \rrbracket_{(\sigma,\rho)} & = & \begin{cases} \llbracket c_0 \rrbracket_{(\sigma,\rho)} & \text{if } \llbracket b \rrbracket_\sigma = \mathbf{true} \\ \llbracket c_1 \rrbracket_{(\sigma,\rho)} & \text{if } \llbracket b \rrbracket_\sigma = \mathbf{false} \end{cases} \\
\llbracket \mathbf{while}\ b\ \mathbf{do}\ c \rrbracket_{(\sigma,\rho)} & = & \lim_{n\to\infty} \llbracket (\mathbf{if}\ b\ \mathbf{then}\ c)^n ; \mathbf{if}\ b\ \mathbf{then\ abort} \rrbracket_{(\sigma,\rho)} \\
\llbracket q := |0\rangle \rrbracket_{(\sigma,\rho)} & = & \langle \sigma, \rho' \rangle \\
 & & \text{where } \rho' := |0\rangle_q\langle 0|\rho|0\rangle_q\langle 0| + |0\rangle_q\langle 1|\rho|1\rangle_q\langle 0| \\
\llbracket U[\bar{q}] \rrbracket_{(\sigma,\rho)} & = & \langle \sigma, U\rho U^\dagger \rangle \\
\llbracket x := M[\bar{q}] \rrbracket_{(\sigma,\rho)} & = & \mu \\
 & & \text{where } M = \{M_i\}_{i\in I} \text{ and } \mu(\sigma') = \sum_i \{M_i \rho M_i^\dagger \mid \sigma[i/x] = \sigma'\} \\
\llbracket c \rrbracket_\mu & = & \sum_{\sigma \in \lceil \mu \rceil} \llbracket c \rrbracket_{(\sigma, \mu(\sigma))}.
\end{array}
$$

Figure 3: Denotational semantics of commands

The expressions, ranged over by P, can be constructed by the following grammar.

$$P \ := \ \mathbf{1}_\mu \mid S \mid \neg P \mid P_1 \wedge P_2 \mid \Box\psi \mid P_1 \oplus P_2 \mid P[f]$$

where $\mu \in \mathbf{POVD}$, $S \subseteq \mathbf{POVD}$, ψ is a predicate over states and f is a function from **POVD** to **POVD**.

Here $\mathbf{1}_\mu$ is also called the characteristic function of the POVD μ, which is a predicate requiring that $\mathbf{1}_\mu$ holds on μ' if and only if $\mu' = \mu$, for any distribution state μ'. The satisfaction relation $\models$ between a POVD and an assertion is defined as follows.

$$
\begin{array}{rll}
\mu \models \mathbf{1}_{\mu'} & \text{iff} & \mu = \mu' \\
\mu \models S & \text{iff} & \mu \in S \\
\mu \models \neg P & \text{iff} & \text{not } \mu \models P \\
\mu \models P_1 \wedge P_2 & \text{iff} & \mu \models P_1 \wedge \mu \models P_2 \\
\mu \models \Box\psi & \text{iff} & \forall \sigma.\ \sigma \in \lceil \mu \rceil \Rightarrow \llbracket \psi \rrbracket_\sigma = \mathbf{true} \\
\mu \models P_1 \oplus P_2 & \text{iff} & \exists \mu_1, \mu_2.\ \mu = \mu_1 + \mu_2 \wedge \mu_1 \models P_1 \wedge \mu_2 \models P_2 \\
\mu \models P[f] & \text{iff} & f(\mu) \models P
\end{array}
$$

Let $\llbracket P \rrbracket := \{\mu \mid \mu \models P\}$ be the semantic interpretation of assertion P. We see that boolean operations of assertions are represented by set operations. For example, we have $\llbracket \neg P \rrbracket = \mathcal{P}(\mathbf{POVD}) \backslash \llbracket P \rrbracket$ and $\llbracket P_1 \wedge P_2 \rrbracket = \llbracket P_1 \rrbracket \cap \llbracket P_2 \rrbracket$. The predicate $\Box\psi$ is lifted from a state predicate by requiring that $\Box\psi$ holds on the POVD μ when ψ holds on all the states in the support of μ. For example, a particular predicate over states is a boolean expression b with $\sigma \models b$ iff $\llbracket b \rrbracket_\sigma = \mathbf{true}$. Therefore, the predicate $\Box b$ holds on the POVD μ when b evaluates to be true under any state σ in the support of μ. The

$$\frac{}{\{P\}\ \mathbf{skip}\ \{P\}}\,[\text{Skip}] \qquad \frac{}{\{P\}\ \mathbf{abort}\ \{\Box\mathbf{false}\}}\,[\text{Abort}] \qquad \frac{}{\{P[\![\![x := a]\!]\!]\}\ x := a\ \{P\}}\,[\text{Assgn}]$$

$$\frac{\{P_0\}\ c_0\ \{P_1\} \quad \{P_1\}\ c_1\ \{P_2\}}{\{P_0\}\ c_0; c_1\ \{P_2\}}\,[\text{Seq}] \qquad \frac{\{P_0\}\ c\ \{P_0'\} \quad \{P_1\}\ c\ \{P_1'\}}{\{P_0 \oplus P_1\}\ c\ \{P_0' \oplus P_1'\}}\,[\text{Split}]$$

$$\frac{\{P_0 \wedge \Box b\}\ c_0\ \{P_0'\} \quad \{P_1 \wedge \Box\neg b\}\ c_1\ \{P_1'\}}{\{(P_0 \wedge \Box b) \oplus (P_1 \wedge \Box\neg b)\}\ \mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1\ \{P_0' \oplus P_1'\}}\,[\text{Cond}]$$

$$\frac{}{\{\mathbf{false}\}\ c\ \{P\}}\,[\text{Absurd}] \qquad \frac{P_0 \Rightarrow P_1 \quad \{P_1\}\ c\ \{P_2\} \quad P_2 \Rightarrow P_3}{\{P_0\}\ c\ \{P_3\}}\,[\text{Conseq}] \qquad \frac{\forall\mu.\ \{\mathbf{1}_\mu \wedge P\}\ c\ \{P'\}}{\{P\}\ c\ \{P'\}}\,[\text{All}]$$

$$\frac{\begin{array}{c}\text{uclosed}((\text{P}'_\text{n})_{\text{n}\in\mathbb{N}^\infty}) \\ \forall n.\ \{P_n\}\ \mathbf{if}\ b\ \mathbf{then}\ c\ \{P_{n+1}\} \quad \forall n.\ \{P_n\}\ \mathbf{if}\ b\ \mathbf{then\ abort}\ \{P_n'\}\end{array}}{\{P_0\}\ \mathbf{while}\ b\ \mathbf{do}\ c\ \{P_\infty' \wedge \Box\neg b\}}\,[\text{While}]$$

$$\frac{}{\{P[\![\![q := |0\rangle]\!]\!]\}\ q := |0\rangle\ \{P\}}\,[\text{QInit}] \qquad \frac{}{\{P[\![\![U[\bar{q}]]\!]\!]\}\ U[\bar{q}]\ \{P\}}\,[\text{QUnit}]$$

$$\frac{}{\{P[\![\![x := M[\bar{q}]]\!]\!]\}\ x := M[\bar{q}]\ \{P\}}\,[\text{QMeas}]$$

Figure 4: Proof rules for $\mathcal{S}_a$

assertion $P_1 \oplus P_2$ holds on the POVD μ if we can split μ into the sum of two POVDs such that P_1 and P_2 hold on each of them. Lastly, $P[f]$ holds on a POVD μ only when P holds on the image of μ under f.

Definition 4.2. A sequence of assertions $(P_n)_{n\in\mathbb{N}^\infty}$ is *u-closed*, if for each increasing sequence of POVDs $(\mu_n)_{n\in\mathbb{N}}$ such that $\mu_n \models P_n$ for all $n \in \mathbb{N}$, we have $\lim_{n\to\infty} \mu_n \models P_\infty$.

Definition 4.3. A judgement is a triple in the form $\{P\}\ c\ \{P'\}$, where c is a command, P and P' are assertions. It is valid, written $\models \{P\}\ c\ \{P'\}$, if

$$\forall\mu.\ \mu \models P \ \Rightarrow\ [\![c]\!]_\mu \models P'.$$

In Figure 4 we give the rules for an abstract proof system denoted by $\mathcal{S}_a$. It extends the system in [3] with the last three rules to handle the manipulations of quantum systems. In order to show the soundness of $\mathcal{S}_a$, we need a few technical lemmas.

Lemma 4.4. Let P be an assertion and c a command. Then $\models \{P[\![\![c]\!]\!]\}\ c\ \{P\}$.

Proof: Suppose μ is a distribution state and $\mu \models P[\![\![c]\!]\!]$. By the definition of $P[\![\![c]\!]\!]$, this means that $[\![c]\!]_\mu \models P$, which is the desired result. □

Lemma 4.5. Let σ be a classical state, ρ_1, ρ_2 be two quantum states, and μ_1, μ_2 be two POVDs. For any command c, we have

1. $\llbracket c \rrbracket_{(\sigma,\rho_1+\rho_2)} = \llbracket c \rrbracket_{(\sigma,\rho_1)} + \llbracket c \rrbracket_{(\sigma,\rho_2)}$;
2. $\llbracket c \rrbracket_{(\mu_1+\mu_2)} = \llbracket c \rrbracket_{\mu_1} + \llbracket c \rrbracket_{\mu_2}$.

Proof: The two clauses can be proved by a simultaneous induction on the structure of command c. □

Lemma 4.6. For any commands c_0, c_1 and distribution state μ, we have $\llbracket c_0; c_1 \rrbracket_\mu = \llbracket c_1 \rrbracket_{\llbracket c_0 \rrbracket_\mu}$.

Proof:

$$\begin{array}{rcll}
\llbracket c_1 \rrbracket_{\llbracket c_0 \rrbracket_\mu} & = & \sum_\sigma \llbracket c_1 \rrbracket_{(\sigma, \llbracket c_0 \rrbracket_\mu(\sigma))} & \\
& = & \sum_\sigma \llbracket c_1 \rrbracket_{(\sigma, \sum_{\sigma'} \llbracket c_0 \rrbracket_{(\sigma', \mu(\sigma'))}(\sigma))} & \\
& = & \sum_\sigma \sum_{\sigma'} \llbracket c_1 \rrbracket_{(\sigma, \llbracket c_0 \rrbracket_{(\sigma', \mu(\sigma'))}(\sigma))} & \text{by Lemma 4.5(1)} \\
& = & \sum_{\sigma'} \sum_\sigma \llbracket c_1 \rrbracket_{(\sigma, \llbracket c_0 \rrbracket_{(\sigma', \mu(\sigma'))}(\sigma))} & \\
& = & \sum_{\sigma'} \llbracket c_1 \rrbracket_{\llbracket c_0 \rrbracket_{(\sigma', \mu(\sigma'))}} & \\
& = & \sum_{\sigma'} \llbracket c_0; c_1 \rrbracket_{(\sigma', \mu(\sigma'))} & \\
& = & \llbracket c_0; c_1 \rrbracket_\mu &
\end{array}$$

□

Theorem 4.7. (Soundness) Every judgement provable using the proof system $\mathcal{S}_a$ is valid.

Proof: We analyze the cases one by one.

- Rule [Skip]. Suppose $\mu \models P$ for some distribution state μ. Then we have $\llbracket \textbf{skip} \rrbracket_\mu = \mu$ and thus $\llbracket \textbf{skip} \rrbracket_\mu \models P$ as required.

- Rule [Abort]. This case is easy by noting that $\llbracket \textbf{abort} \rrbracket_\mu = \varepsilon$ and $\varepsilon \models \Box\textbf{false}$ for any μ.

- The cases for rules [Assgn], [QInit], [QUnit], and [QMeas] follow from Lemma 4.4.

- Rule [Seq]. Suppose $\mu \models P_0$ for some distribution state μ. By the premises, both $\{P_0\}\ c_0\ \{P_1\}$ and $\{P_1\}\ c_1\ \{P_2\}$ are valid. It follows that $\llbracket c_0 \rrbracket_\mu \models P_1$ and then $\llbracket c_1 \rrbracket_{\llbracket c_0 \rrbracket_\mu} \models P_2$, which is $\llbracket c_0; c_1 \rrbracket_\mu \models P_2$ by Lemma 4.6 as required.

- Rule [Split]. Suppose $\mu \models P_0 \oplus P_1$ for some distribution state μ. Then there exist μ_0 and μ_1 such that $\mu = \mu_0 + \mu_1$, $\mu_0 \models P_0$ and $\mu_1 \models P_1$. By the premises, both $\{P_0\}\ c_0\ \{P'_0\}$ and $\{P_1\}\ c_1\ \{P'_1\}$ are valid. Therefore, we have that $\llbracket c \rrbracket_{\mu_0} \models P'_0$ and $\llbracket c \rrbracket_{\mu_1} \models P'_1$. By Lemma 4.5 we obtain $\llbracket c \rrbracket_\mu = \llbracket c \rrbracket_{\mu_0} + \llbracket c \rrbracket_{\mu_1}$. It follows that $\llbracket c \rrbracket_\mu \models P'_0 \oplus P'_1$ as required.

- Rule [Cond]. We first claim that $\{P_0 \wedge \Box b\}$ **if** b **then** c_0 **else** c_1 $\{P'_0\}$ is valid. To see this, suppose μ is a POVD with $\mu \models P_0 \wedge \Box b$. Obviously, we have $\mu \models \Box b$ and thus $\llbracket b \rrbracket_\sigma = \textbf{true}$ for each $\sigma \in \lceil \mu \rceil$. It follows that

  $$\begin{array}{rcl}
  \llbracket \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 \rrbracket_\mu & = & \sum_\sigma \llbracket \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 \rrbracket_{(\sigma, \mu(\sigma))} \\
  & = & \sum_\sigma \llbracket c_0 \rrbracket_{(\sigma, \mu(\sigma))} \\
  & = & \llbracket c_0 \rrbracket_\mu.
  \end{array}$$

  By the first premise, $\{P_0 \wedge \Box b\}\ c_0\ \{P'_0\}$ is valid. Therefore, we have $\llbracket c_0 \rrbracket_\mu \models P'_0$, and thus

  $$\llbracket \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 \rrbracket_\mu \models P'_0$$

and the above claim is proved. Similarly, we can prove that $\{P_1 \wedge \Box\neg b\}$ **if** b **then** c_0 **else** c_1 $\{P'_1\}$ is valid. By the soundness of [Split], it follows that

$$\{(P_0 \wedge \Box b) \oplus (P_1 \wedge \Box\neg b)\}\ \textbf{if}\ b\ \textbf{then}\ c_0\ \textbf{else}\ c_1\ \{P'_0 \oplus P'_1\}$$

is also valid.

- Rule [Absurd]. There exists no μ with $\mu \models$ **false**. Thus, we always have $\forall\mu.\ \mu \models \textbf{false} \Rightarrow [\![c]\!]_\mu \models P$.
- Rule [Conseq]. Let μ be a distribution state and $\mu \models P_0$. The first premise gives $\mu \models P_1$. The second premise tells us that $[\![c]\!]_\mu \models P_2$. By the third premise, we derive that $[\![c]\!]_\mu \models P_3$. It follows that $\{P_0\}\ c\ \{P_3\}$ is valid.
- Rule [All]. Let μ be a POVD and $\mu \models P$. It is clear that $\mu \models \mathbf{1}_\mu \wedge P$. By the premise, $\{\mathbf{1}_\mu \wedge P\}\ c\ \{P'\}$ is valid. Therefore, we have $[\![c]\!]_\mu \models P'$, and thus $\{P\}\ c\ \{P'\}$ is valid.
- Rule [While]. We first observe that, for any state (σ,ρ),

$$[\![\textbf{if}\ b\ \textbf{then abort}]\!]_{(\sigma,\rho)} = \begin{cases} \varepsilon & \text{if } [\![b]\!]_{(\sigma,\rho)} = \textbf{true} \\ (\sigma,\rho) & \text{if } [\![b]\!]_{(\sigma,\rho)} = \textbf{false}\,. \end{cases}$$

Thus, if a state σ' is in the support of $[\![\textbf{if}\ b\ \textbf{then abort}]\!]_{(\sigma,\rho)}$, it must be the case that $\sigma' \models \neg b$. Furthermore, for any distribution state μ, if a state σ' is in the support of $[\![\textbf{if}\ b\ \textbf{then abort}]\!]_\mu$ then $\sigma' \models \neg b$. It follows that, for any command c' and distribution state μ, we have

$$[\![c'; \textbf{if}\ b\ \textbf{then abort}]\!]_\mu \models \Box\neg b.$$

By definition, $[\![\textbf{while}\ b\ \textbf{do}\ c]\!]_\mu$ is the limit of the sequence

$$([\![(\textbf{if}\ b\ \textbf{then}\ c)^n; \textbf{if}\ b\ \textbf{then abort}]\!]_\mu)_{n\in\mathbb{N}}$$

and so we have that

$$[\![\textbf{while}\ b\ \textbf{do}\ c]\!]_\mu \models \Box\neg b. \tag{1}$$

By the first premise and the soundness of [Seq], it is easy to show by induction that

$$\forall n.\ \{P_0\}\ (\textbf{if}\ b\ \textbf{then}\ c)^n\ \{P_n\}$$

is valid. By the second premise and [Seg] again, the following judgement

$$\forall n.\ \{P_0\}\ (\textbf{if}\ b\ \textbf{then}\ c)^n; \textbf{if}\ b\ \textbf{then abort}\ \{P'_n\}$$

is valid. Let μ be any POVD with $\mu \models P_0$. Then

$$\forall n.\ [\![(\textbf{if}\ b\ \textbf{then}\ c)^n; \textbf{if}\ b\ \textbf{then abort}]\!]_\mu \models P'_n.$$

By assumption, the sequence of assertions $(P'_n)_{n\in\mathbb{N}^\infty}$ is u-closed. Hence, we can infer that

$$[\![\textbf{while}\ b\ \textbf{do}\ c]\!]_\mu \models P'_\infty,$$

which means that the judgement

$$\{P_0\}\ \textbf{while}\ b\ \textbf{do}\ c\ \{P'_\infty\} \tag{2}$$

is valid. Combining (1) and (2), we finally obtain that $\{P_0\}$ **while** b **do** c $\{P'_\infty \wedge \Box\neg b\}$ is valid.

□

Now we turn to the relative completeness of the proof system $\mathcal{S}_a$. Formulas of the form $\mathbf{1}_\mu$ will be helpful for that purpose.

Lemma 4.8. For any distribution state μ and command c,

$$\mathbf{1}_\mu \Rightarrow \mathbf{1}_{[\![c]\!]_\mu}[[\![c]\!]].$$

Proof: Let μ' be any distribution state.

$$\begin{array}{rcl} \mu' \models \mathbf{1}_\mu & \Leftrightarrow & \mu' = \mu \\ & \Rightarrow & [\![c]\!]_{\mu'} = [\![c]\!]_\mu \\ & \Leftrightarrow & [\![c]\!]_{\mu'} \models \mathbf{1}_{[\![c]\!]_\mu} \\ & \Leftrightarrow & \mu' \models \mathbf{1}_{[\![c]\!]_\mu}[[\![c]\!]] \end{array}$$

□

Definition 4.9. Let μ be a distribution state and b a boolean expression. The restriction $\mu_{|b}$ of μ to b is the distribution state such that $\mu_{|b}(\sigma) = \mu(\sigma)$ if $[\![b]\!]_\sigma = \mathbf{true}$ and 0 otherwise.

According to the definition above, it is easy to see that we can split any μ into two parts w.r.t. a boolean expression.

Lemma 4.10. For any distribution state μ and boolean expression b, we have $\mu = \mu_{|b} + \mu_{|\neg b}$.

Proof: This is straightforward because, at each state σ in the support of μ, the boolean expression b evaluates to either **true** or **false**. □

With Lemmas 4.10 and 4.5, it is easy to see that the denotational semantics of conditional commands can be rewritten as follows.

$$[\![\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1]\!]_\mu \ = \ [\![c_0]\!]_{\mu_{|b}} + [\![c_1]\!]_{\mu_{|\neg b}} \tag{3}$$

The following facts are also easy to show.

$$\begin{array}{rcll} \mathbf{1}_{\mu|b} & \Leftrightarrow & \mathbf{1}_{\mu|b} \wedge \Box b & \\ \mathbf{1}_\mu & \Leftrightarrow & \mathbf{1}_\mu \wedge P & \text{if } \mu \models P \\ \mathbf{1}_\mu & \Leftrightarrow & \mathbf{1}_{\mu|b} \oplus \mathbf{1}_{\mu|\neg b} & \\ \mathbf{1}_{\mu_1+\mu_2} & \Leftrightarrow & \mathbf{1}_{\mu_1} \oplus \mathbf{1}_{\mu_2} & \end{array} \tag{4}$$

Lemma 4.11. For any POVD μ, the following judgement is provable:

$$\{\mathbf{1}_\mu\}\ c\ \{\mathbf{1}_{[\![c]\!]_\mu}\}.$$

Proof: We proceed by induction on the structure of c.

- $c \equiv \mathbf{skip}$. This case is immediate as $[\![\mathbf{skip}]\!]_\mu = \mu$ and by [Skip] we have $\vdash \{\mathbf{1}_\mu\}\ c\ \{\mathbf{1}_\mu\}$.
- $c \equiv \mathbf{abort}$. Then $[\![c]\!]_\mu = \varepsilon$. For any POVD μ', we note that

$$\mu' \models \Box\mathbf{false} \ \Leftrightarrow\ \mu' = \varepsilon \ \Leftrightarrow\ \mu' \models \mathbf{1}_\varepsilon.$$

By rules [Abort] and [Conseq] we can infer $\vdash \{\mathbf{1}_\mu\}\ c\ \{\mathbf{1}_\varepsilon\}$.

- $c \equiv x := a$, $q := |0\rangle$, $U[\bar{q}]$ or $x := M[\bar{q}]$. By the corresponding rules [Assgn], [QInit], [QUnit] or [QMeas], we have
$$\vdash \{\mathbf{1}_{\llbracket c \rrbracket_\mu}[\llbracket c \rrbracket]\}\ c\ \{\mathbf{1}_{\llbracket c \rrbracket_\mu}\}.$$
By Lemma 4.8 and rule [Conseq], we obtain that $\vdash \{\mathbf{1}_\mu\}\ c\ \{\mathbf{1}_{\llbracket c \rrbracket_\mu}\}$.

- $c \equiv c_0; c_1$. By induction, we have $\vdash \{\mathbf{1}_\mu\}\ c_0\ \{\mathbf{1}_{\llbracket c_0 \rrbracket_\mu}\}$ and $\vdash \{\mathbf{1}_{\llbracket c_0 \rrbracket_\mu}\}\ c_1\ \{\mathbf{1}_{\llbracket c_1 \rrbracket_{\llbracket c_0 \rrbracket_\mu}}\}$. Using the rule [Seq], we obtain that $\vdash \{\mathbf{1}_\mu\}\ c\ \{\mathbf{1}_{\llbracket c_1 \rrbracket_{\llbracket c \rrbracket_\mu}}\}$.

- $c \equiv \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1$. By induction, we have $\vdash \{\mathbf{1}_{\mu_{|b}}\}\ c_0\ \{\mathbf{1}_{\llbracket c_0 \rrbracket_{\mu_{|b}}}\}$. By the first clause in (4) and rule [Conseq], we have $\vdash \{\mathbf{1}_{\mu_{|b}} \wedge \Box b\}\ c_0\ \{\mathbf{1}_{\llbracket c_0 \rrbracket_{\mu_{|b}}}\}$. Similarly, $\vdash \{\mathbf{1}_{\mu_{|\neg b}} \wedge \Box\neg b\}\ c_1\ \{\mathbf{1}_{\llbracket c_1 \rrbracket_{\mu_{|\neg b}}}\}$. Using rule [Cond], we infer
$$\vdash \{(\mathbf{1}_{\mu_{|b}} \wedge \Box b) \oplus (\mathbf{1}_{\mu_{|\neg b}} \wedge \Box\neg b)\}\ \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1\ \{\mathbf{1}_{\llbracket c_0 \rrbracket_{\mu_{|b}}} \oplus \mathbf{1}_{\llbracket c_1 \rrbracket_{\mu_{|\neg b}}}\}.$$
Using (3), (4) and rule [Conseq], we finally obtain that
$$\vdash \{\mathbf{1}_\mu\}\ \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1\ \{\mathbf{1}_{\llbracket \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 \rrbracket_\mu}\}.$$

- $c \equiv \textbf{while } b \textbf{ do } c'$. For each $n \in \mathbb{N}$, let
$$\begin{array}{rcl} P_n & = & \mathbf{1}_{\llbracket (\textbf{if } b \textbf{ then } c')^n \rrbracket_\mu} \\ P'_n & = & \mathbf{1}_{\llbracket (\textbf{if } b \textbf{ then } c')^n ; \textbf{if } b \textbf{ then abort} \rrbracket_\mu} \\ P'_\infty & = & \mathbf{1}_{\lim_{n\to\infty} \llbracket (\textbf{if } b \textbf{ then } c')^n ; \textbf{if } b \textbf{ then abort} \rrbracket_\mu} \end{array}$$
Obviously, the sequence of assertions $(P'_n)_{n \in \mathbb{N}^\infty}$ is u-closed. As in the last case, we can show that $\vdash \{P_n\}\ \textbf{if } b \textbf{ then } c'\ \{P_{n+1}\}$ by induction hypothesis and rules [Conseq], [Skip] and [Cond]. It is also easy to see that $\vdash \{P_n\}\ \textbf{if } b \textbf{ then abort}\ \{P'_n\}$ for each $n \in \mathbb{N}$. Therefore, we can use rule [While] to infer that $\vdash \{P_0\}\ \textbf{while } b \textbf{ do } c\ \{P'_\infty \wedge \Box\neg b\}$. Using (1), the second clause of (4), and rule [Conseq], we obtain that $\vdash \{P_0\}\ \textbf{while } b \textbf{ do } c\ \{P'_\infty\}$, which is exactly $\vdash \{\mathbf{1}_\mu\}\ \textbf{while } b \textbf{ do } c'\ \{\mathbf{1}_{\llbracket \textbf{while } b \textbf{ do } c' \rrbracket_\mu}\}$.

□

With the preparations above, we are in the position to show that the proof system $\mathcal{S}_a$ is relatively complete.

Theorem 4.12. (Relative completeness) Every valid judgement is derivable in $\mathcal{S}_a$.

Proof: Let $\{P\}\ c\ \{P'\}$ be a valid judgement. Suppose μ be any POVD. There are two possibilities:

- $\mu \models P$. The validity of the judgement says that $\llbracket c \rrbracket_\mu \models P'$. By Lemma 4.11, we have that $\vdash \{\mathbf{1}_\mu\}\ c\ \{\mathbf{1}_{\llbracket c \rrbracket_\mu}\}$. By the second clause of (4) and rule [Conseq], we can obtain that $\vdash \{\mathbf{1}_\mu \wedge P\}\ c\ \{\mathbf{1}_{\llbracket c \rrbracket_\mu} \wedge P'\}$. Using [Conseq] again gives $\vdash \{\mathbf{1}_\mu \wedge P\}\ c\ \{P'\}$.

- $\mu \not\models P$. Then it is obvious that $\mathbf{1}_\mu \wedge P \Leftrightarrow \textbf{false}$. By rules [Absurd] and [Conseq], we also obtain $\vdash \{\mathbf{1}_\mu \wedge P\}\ c\ \{P'\}$.

Since μ is arbitrarily chosen, the premise of rule [All] is derivable. Therefore, we can use that rule to obtain $\vdash \{P\}\ c\ \{P'\}$. □

5. A Concrete Program Logic

In this section, we present a concrete program logic. We first define the concrete syntax of assertions. Following [3], we define a two-level assertion language in Figure 5. Formally, assertions are divided into two categories: *state assertions* are formulas that describe the properties of machine states and *distribution assertions* are used to describe the properties of POVDs. Distribution assertions are based on comparison of distribution expressions and the connective $\oplus$ mentioned in Section 4. It is easy to extend the syntax to allow more connectives such as first-order quantifiers and connectives. Since the current work does not deal with the implementation of the concrete program logic in a proof assistant, we do not give a very detailed definition of the syntax of assertions. A *distribution expression* is either the expectation $\mathbb{E}[e]$ of a state expression e, the expectation $\mathbb{E}_{\bar{x}\sim M[\bar{q}]}[e]$ of state expression e w.r.t. the measurement M. Again, it is possible to add other constructors for distribution expressions such as an operator applied to distribution expressions. A *state expression* is either a classical variable, the characteristic function $\mathbf{1}_\psi$ of a state assertion ψ, or an operator applied to state expressions. Note that we consider 0-ary operations for state expressions as constants. Finally, a *state assertion* is either a comparison of state expressions, or a first-order formula over state assertions. In particular, the boolean expressions in **Bexp** are included as state assertions. Note that the set of operators is left unspecified but we assume that some basic operators such as addition, subtraction, scalar multiplication for both arithmetic expressions and matrix representation of partial density operators are included. With a slight abuse of notation, when $\leq$ is used to compare matrices, we essentially mean $\sqsubseteq$. Similarly for $<$ and $=$.

For convenience of presentation, in this section we consider a general form of quantum measurement.

Definition 5.1. A *general measurement* M is a pair $\langle\{M_i\}_{i\in I}, l\rangle$, where each M_i is a measurement operator as usual, and $l : I \mapsto J$ is a labelling function that maps each measurement outcome i to some some label $l(i)$.

If the state of a quantum system is specified by density operator ρ immediately before the measurement M, then the probability with which those results with label j occur is given by

$$p(j) = \sum_{i:l(i)=j} \mathrm{tr}(M_i^\dagger M_i \rho),$$

and the state of the system after the measurement is

$$\frac{\sum_{i:l(i)=j} M_i \rho M_i^\dagger}{p(j)}.$$

General measurements are convenient to describe the situation where we would like to group some measurement outcomes. For example, if $i_1, i_2 \in I$ are two different outcomes, but for some reasons we would not like to distinguish them, then we simply give them the same label by letting $l(l_1) = l(l_2)$. In the special case that l is the identity function Id, then the labelling function has no effect and we degenerate to the usual notion of measurements.

The interpretation of assertions is given in Figure 6. Comparing the interpretation with that in [3], we see that the main difference is the introduction of a distribution expression related to a quantum measurement. The meaning of $\mathbb{E}_{\bar{x}\sim M[\bar{q}]}[e]$ is the expected Hermitian operator weighted by the value of e after a measurement entailed by M.

Note that the formula $\Box\psi$, where ψ is a state assertion, can now be considered as a syntactic sugar in view of the following lemma.

$$
\begin{array}{rcll}
e & ::= & x \mid \mathbf{1}_{\psi} \mid o(\mathbf{e}) & \text{(State expressions)} \\
\psi & ::= & e \bowtie e \mid FO(\psi) & \text{(State assertions)} \\
r & ::= & \mathbb{E}[e] \mid \mathbb{E}_{\bar{x} \sim M[\bar{q}]}[e] \mid \cdots & \text{(Distribution expressions)} \\
P & ::= & r \bowtie r \mid P \oplus P \mid \cdots & \text{(Distribution assertions)} \\
\bowtie & \in & \{=,\ <,\ \leq\} \qquad o \in Ops & \text{(Operations)}
\end{array}
$$

Figure 5: Syntax of assertions

$$
\begin{array}{rcl}
[\![x]\!]_{\sigma} & := & \sigma(x) \\
[\![\mathbf{1}_{\psi}]\!]_{\sigma} & := & \mathbf{1}_{[\![\psi]\!]_{\sigma}} \\
[\![o(e)]\!]_{\sigma} & := & o([\![e]\!]_{\sigma}) \\
\hline
[\![e_1 \bowtie e_2]\!]_{\sigma} & := & [\![e_1]\!]_{\sigma} \bowtie [\![e_2]\!]_{\sigma} \\
[\![FO(\psi)]\!]_{\sigma} & := & FO([\![\psi]\!]_{\sigma}) \\
\hline
[\![\mathbb{E}[e]]\!]_{\mu} & := & \sum_{\sigma} \mu(\sigma) \cdot [\![e]\!]_{\sigma} \\
[\![\mathbb{E}_{\bar{x} \sim M[\bar{q}]}[e]]\!]_{\mu} & := & \sum_{\sigma} \sum_{i} M_i \mu(\sigma) M_i^{\dagger} \cdot [\![e]\!]_{\sigma[l(i)/\bar{x}]} \\
 & & \text{where } M = \langle \{M_i\}_{i \in I}, l \rangle \\
\hline
[\![r_1 \bowtie r_2]\!]_{\mu} & := & [\![r_1]\!]_{\mu} \bowtie [\![r_2]\!]_{\mu} \\
[\![P_1 \oplus P_2]\!]_{\mu} & := & \exists \mu_1, \mu_2.\ \mu = \mu_1 + \mu_2 \wedge [\![P_1]\!]_{\mu_1} \wedge [\![P_2]\!]_{\mu_2}
\end{array}
$$

Figure 6: Semantics of assertions

Lemma 5.2. 1. $\square\psi \Leftrightarrow \mathbb{E}[\mathbf{1}_{\psi}] = \mathbb{E}[\mathbf{1}_{\mathbf{true}}]$

2. $\square\psi \Leftrightarrow \mathbb{E}_{\bar{x} \sim M[\bar{q}]}[\mathbf{1}_{\psi}] = \mathbb{E}_{\bar{x} \sim M[\bar{q}]}[\mathbf{1}_{\mathbf{true}}]$

3. $\square\psi \Leftrightarrow \square(\psi \wedge b) \oplus \square(\psi \wedge \neg b)$

Proof: Let us consider the first clause; the second one is similar and the third one is easier.

$$
\begin{array}{rll}
\mu \models \square\psi & \text{iff} & \forall \sigma \in \lceil \mu \rceil. [\![\psi]\!]_{\sigma} = \mathbf{true} \\
 & \text{iff} & \sum_{\sigma} \mu(\sigma) \cdot [\![\mathbf{1}_{\psi}]\!]_{\sigma} = \sum_{\sigma} \mu(\sigma) \cdot [\![\mathbf{1}_{\mathbf{true}}]\!]_{\sigma} \\
 & \text{iff} & [\![\mathbb{E}[\mathbf{1}_{\psi}]]\!]_{\mu} = [\![\mathbb{E}[\mathbf{1}_{\mathbf{true}}]]\!]_{\mu} \\
 & \text{iff} & \mu \models (\mathbb{E}[\mathbf{1}_{\psi}] = \mathbb{E}[\mathbf{1}_{\mathbf{true}}])
\end{array}
$$

□

Using the concrete syntax for assertions, we propose a syntactic version of the existing proof rules by avoiding the semantics of commands. We call the concrete proof system $\mathcal{S}_c$. Specifically, we keep all proof rules in Figure 4 but replace [Assgn], [QInit], [QUnit], and [QMeas] with the four rules in Figure 7.

In rule [QInit'] we use the notation $h(P)$ for a syntactic substitution. It changes all $\mathbb{E}_{\bar{x} \sim M[\bar{q}]}[e]$ in P into $\mathbb{E}_{\bar{x} \sim M'[\bar{q}]}[e]$ and distributes over most other syntactic constructors of assertions, where M' is obtained from $M = \langle \{M_i\}_i, l \rangle$ by constructing two measurement operators M_{i0}, M_{i1} for each M_i in M

$$\frac{}{\{P[a/x]\}\ x := a\ \{P\}}\,[\text{Assgn}'] \qquad \frac{}{\{h(P)\}\ q := |0\rangle\ \{P\}}\,[\text{QInit}']$$

$$\frac{}{\{g^U(P)\}\ U[\bar{q}]\ \{P\}}\,[\text{QUnit}'] \qquad \frac{}{\{f^M_{x,\bar{q}}(P)\}\ x := M[\bar{q}]\ \{P\}}\,[\text{QMeas}']$$

Figure 7: Selected syntactic proof rules

with the mapping l' given by $l'(i0) = l'(i1) = l(i)$. A formal definition is given below.

$$\begin{array}{rcl}
h(\mathbb{E}[e]) & := & \mathbb{E}_{x\sim M[q]}[e] \quad \text{where } M = \langle\{M_0, M_1\}, Id\rangle \text{ with } M_0 = |0\rangle\langle 0|,\ M_1 = |0\rangle\langle 1|, \\
 & & \hfill x \text{ is fresh} \\
h(\mathbb{E}_{\bar{x}\sim M[\bar{q}]}[e]) & := & \mathbb{E}_{\bar{x}\sim M'[\bar{q}]}[e] \quad \text{where } M' = \langle\{M_{i0}, M_{i1}\}_i, l'\rangle \\
 & & \qquad \text{with } M_{i0} = M_i|0\rangle\langle 0|,\ M_{i1} = M_i|0\rangle\langle 1|, l'(i0) = l'(i1) = l(i) \\
h(r_1 \bowtie r_2) & := & h(r_1) \bowtie h(r_2) \\
h(P_1 \oplus P_2) & := & h(P_1) \oplus h(P_2)
\end{array}$$

To ensure the freshness requirement on x in $h(\mathbb{E}[e])$, we assume an enumeration of all the variables in **Cvar**. Each time a fresh variable is needed, we take the next one which has not appeared in all the programs under consideration.

In rule [QUnit'] we use the notation $g^U(P)$ for a syntactic substitution. It changes all $\mathbb{E}_{\bar{x}\sim M[\bar{q}]}[e]$ in P into $\mathbb{E}_{\bar{x}\sim M'[\bar{q}]}[e]$, where $M = \langle\{M_i\}_{i\in I}, l\rangle$, $M' = \langle\{M_iU\}_{i\in I}, l\rangle$, and distributes over most other syntactic constructors of assertions. A formal definition is given below.

$$\begin{array}{rcl}
g^U(\mathbb{E}[e]) & := & \mathbb{E}_{x\sim M[\bar{q}]}[e] \quad \text{where } M = \langle\{M_0\}, Id\rangle \text{ with } M_0 = U \text{ and } x \text{ is fresh} \\
g^U(\mathbb{E}_{\bar{x}\sim M[\bar{q}]}[e]) & := & \mathbb{E}_{\bar{x}\sim M'[\bar{q}]}[e] \quad \text{where } M' = \langle\{M_iU\}_{i\in I}, l\rangle \\
g^U(r_1 \bowtie r_2) & := & g^U(r_1) \bowtie g^U(r_2) \\
g^U(P_1 \oplus P_2) & := & g^U(P_1) \oplus g^U(P_2)
\end{array}$$

In rule [QMeas'] we use the notation $f^M_{x,\bar{q}}(P)$ for a syntactic substitution. It changes all $\mathbb{E}[e]$ in P into $\mathbb{E}_{x\sim M[\bar{q}]}[e]$. For the distribution expression $\mathbb{E}_{\bar{y}\sim N[\bar{q}]}[e]$, it adds an outer layer of measurement to N. A formal definition is given below.

$$\begin{array}{rcl}
f^M_{x,\bar{q}}(\mathbb{E}[e]) & := & \mathbb{E}_{x\sim M[\bar{q}]}[e] \\
f^M_{x,\bar{q}}(\mathbb{E}_{\bar{y}\sim N[\bar{q}']}[e]) & := & \begin{cases} \mathbb{E}_{x\bar{y}\sim M'[\bar{q}\cup\bar{q}']}[e] \text{ with } M' = \langle\{N_jM_i\}_{ij}, k'\rangle \text{ and } k'(ij) = (k(i), l(j)) & \text{if } x \notin \bar{y} \\ \mathbb{E}_{\bar{y}\sim M'[\bar{q}\cup\bar{q}']}[e] \text{ with } M' = \langle\{N_jM_i\}_{ij}, k'\rangle \text{ and } k'(ij) = l(j) & \text{if } x \in \bar{y} \end{cases} \\
f^M_{x,\bar{q}}(r_1 \bowtie r_2) & := & f^M_{x,\bar{q}}(r_1) \bowtie f^M_{x,\bar{q}}(r_2) \\
f^M_{x,\bar{q}}(P_1 \oplus P_2) & := & f^M_{x,\bar{q}}(P_1) \oplus f^M_{x,\bar{q}}(P_2)
\end{array}$$

Let us consider a simple example to illustrate the use of rule [QMeas']; other rules in Figure 7 are similar.

Example 5.3. Suppose ψ is a state assertion and M is a measurement. We claim that the following Hoare triple is derivable in the concrete proof system.

$$\{\mathbb{E}_{y\sim M[q]}[\mathbf{1}_\psi] = \mathbb{E}_{y\sim M[q]}[\mathbf{1}_{\mathbf{true}}]\}\ \ y := M[q]\ \ \{\Box\psi\} \tag{5}$$

We first observe that a direct application of rule [QMeas'] gives

$$\{f^M_{y,q}(\mathbb{E}[\mathbf{1}_\psi] = \mathbb{E}[\mathbf{1}_{\mathbf{true}}])\}\ \ y := M[q]\ \ \{\mathbb{E}[\mathbf{1}_\psi] = \mathbb{E}[\mathbf{1}_{\mathbf{true}}]\}\ . \tag{6}$$

The definition of function $f^M_{y,q}(\cdot)$ tells us that

$$\begin{array}{rl} & f^M_{y,q}(\mathbb{E}[\mathbf{1}_\psi] = \mathbb{E}[\mathbf{1}_{\textsf{true}}]) \\ \equiv & f^M_{y,q}(\mathbb{E}[\mathbf{1}_\psi]) = f^M_{y,q}(\mathbb{E}[\mathbf{1}_{\textsf{true}}]) \\ \equiv & \mathbb{E}_{y\sim M[q]}[\mathbf{1}_\psi] = \mathbb{E}_{y\sim M[q]}[\mathbf{1}_{\textsf{true}}] \end{array} \tag{7}$$

Recall that the first item of Lemma 5.2 says

$$\Box\psi \;\Leftrightarrow\; \mathbb{E}[\mathbf{1}_\psi] = \mathbb{E}[\mathbf{1}_{\textsf{true}}]\,. \tag{8}$$

We can now apply rule [Conseq], together with (7), (6) and (8) to obtain (5). □

We are going to show that the three functions $h(\cdot)$, $g^U(\cdot)$ and $f^M_{x,\bar{q}}(\cdot)$ behave well as they help to transform postconditions into preconditions for three kinds of commands: initialisation, applications of unitary operations, and measurements of quantum systems.

Lemma 5.4. The following two clauses hold.

(i) $\llbracket h(r)\rrbracket_\mu = \llbracket r\rrbracket_{\llbracket q:=|0\rangle\rrbracket_\mu}$.

(ii) $\llbracket h(P)\rrbracket_\mu \Rightarrow \llbracket P\rrbracket_{\llbracket q:=|0\rangle\rrbracket_\mu}$.

Proof: We prove the two statements by structural induction.

(i) There are two cases for the structure of r.

- $r \equiv \mathbb{E}[e]$. We note that $\llbracket e\rrbracket_\sigma = \llbracket e\rrbracket_{\sigma[n/x]}$ for any number n and fresh variable x in the sense that x does not appear in e. Then we reason as follows.

$$\begin{array}{rcll} \llbracket h(r)\rrbracket_\mu & = & \llbracket \mathbb{E}_{x\sim M}[e]\rrbracket_\mu & \text{where } M = \langle\{M_0, M_1\}, Id\rangle \text{ with } M_0 = |0\rangle\langle 0|,\ M_1 = |0\rangle\langle 1|, \\ & & & \hfill x \text{ is fresh} \\ & = & \multicolumn{2}{l}{\sum_\sigma(|0\rangle\langle 0|\mu(\sigma)|0\rangle\langle 0| \cdot \llbracket e\rrbracket_{\sigma[0/x]} + |0\rangle\langle 1|\mu(\sigma)|1\rangle\langle 0| \cdot \llbracket e\rrbracket_{\sigma[1/x]})} \\ & = & \multicolumn{2}{l}{\sum_\sigma(|0\rangle\langle 0|\mu(\sigma)|0\rangle\langle 0| \cdot \llbracket e\rrbracket_{\sigma} + |0\rangle\langle 1|\mu(\sigma)|1\rangle\langle 0| \cdot \llbracket e\rrbracket_{\sigma}) \qquad x \text{ is fresh}} \\ & = & \multicolumn{2}{l}{\sum_\sigma(|0\rangle\langle 0|\mu(\sigma)|0\rangle\langle 0| + |0\rangle\langle 1|\mu(\sigma)|1\rangle\langle 0|) \cdot \llbracket e\rrbracket_{\sigma}} \\ & = & \multicolumn{2}{l}{\sum_\sigma \llbracket q := |0\rangle\rrbracket_\mu(\sigma) \cdot \llbracket e\rrbracket_{\sigma}} \\ & = & \multicolumn{2}{l}{\llbracket \mathbb{E}[e]\rrbracket_{\llbracket q:=|0\rangle\rrbracket_\mu}} \\ & = & \multicolumn{2}{l}{\llbracket r\rrbracket_{\llbracket q:=|0\rangle\rrbracket_\mu}} \end{array}$$

- $r \equiv \mathbb{E}_{\bar{x}\sim M[\bar{q}]}[e]$ for some $M = \langle\{M_i\}_i, l\rangle$. Then

$$\begin{array}{rcll} \llbracket h(r)\rrbracket_\mu & = & \llbracket \mathbb{E}_{\bar{x}\sim M'[\bar{q}]}[e]\rrbracket_\mu & \text{where } M' = \langle\{M_{i0}, M_{i1}\}_i, l'\rangle \\ & & & \quad\text{with } M_{i0} = M_i|0\rangle\langle 0|,\ M_{i1} = M_i|0\rangle\langle 1|, l'(i0) = l'(i1) = l(i) \\ & = & \multicolumn{2}{l}{\sum_\sigma \sum_i (M_i|0\rangle\langle 0|\mu(\sigma)|0\rangle\langle 0|M_i^\dagger + M_i|0\rangle\langle 1|\mu(\sigma)|1\rangle\langle 0|M_i^\dagger) \cdot \llbracket e\rrbracket_{\sigma[l(i)/\bar{x}]}} \\ & = & \multicolumn{2}{l}{\sum_\sigma \sum_i (M_i(|0\rangle\langle 0|\mu(\sigma)|0\rangle\langle 0| + |0\rangle\langle 1|\mu(\sigma)|1\rangle\langle 0|)M_i^\dagger \cdot \llbracket e\rrbracket_{\sigma[l(i)/\bar{x}]}} \\ & = & \multicolumn{2}{l}{\sum_\sigma \sum_i M_i\llbracket q := |0\rangle\rrbracket_\mu(\sigma)M_i^\dagger \cdot \llbracket e\rrbracket_{\sigma[l(i)/\bar{x}]}} \\ & = & \multicolumn{2}{l}{\llbracket \mathbb{E}_{\bar{x}\sim M[\bar{q}]}[e]\rrbracket_{\llbracket q:=|0\rangle\rrbracket_\mu}} \\ & = & \multicolumn{2}{l}{\llbracket r\rrbracket_{\llbracket q:=|0\rangle\rrbracket_\mu}} \end{array}$$

(ii) There are two cases for the structure of P.

- $P \equiv r_1 \bowtie r_2$. In this case, we need to use statement (i).

$$\begin{array}{rcl}
\llbracket h(P)\rrbracket_\mu & = & \llbracket h(r_1) \bowtie h(r_2)\rrbracket_\mu \\
& = & \llbracket h(r_1)\rrbracket_\mu \bowtie \llbracket h(r_2)\rrbracket_\mu \\
& = & \llbracket r_1\rrbracket_{\llbracket q:=|0\rangle\rrbracket_\mu} \bowtie \llbracket r_2\rrbracket_{\llbracket q:=|0\rangle\rrbracket_\mu}) \\
& = & \llbracket r_1 \bowtie r_2\rrbracket_{\llbracket q:=|0\rangle\rrbracket_\mu} \\
& = & \llbracket P\rrbracket_{\llbracket q:=|0\rangle\rrbracket_\mu}
\end{array}$$

- $P \equiv P_1 \oplus P_2$. This case is proved by induction.

$$\begin{array}{rcl}
\llbracket h(P)\rrbracket_\mu & = & \llbracket h(P_1) \oplus h(P_2)\rrbracket_\mu \\
& = & \exists \mu_1, \mu_2.\ \mu = \mu_1 + \mu_2 \wedge \llbracket h(P_1)\rrbracket_{\mu_1} \wedge \llbracket h(P_2)\rrbracket_{\mu_2} \\
& \Rightarrow & \exists \mu_1, \mu_2.\ \llbracket q := |0\rangle\rrbracket_\mu = \llbracket q := |0\rangle\rrbracket_{\mu_1} + \llbracket q := |0\rangle\rrbracket_{\mu_2} \\
& & \wedge \llbracket P_1\rrbracket_{\llbracket q:=|0\rangle\rrbracket_{\mu_1}} \wedge \llbracket P_2\rrbracket_{\llbracket q:=|0\rangle\rrbracket_{\mu_2}} \qquad \text{by Lemma 4.5} \\
& = & \llbracket P_1 \oplus P_2\rrbracket_{\llbracket q:=|0\rangle\rrbracket_\mu} \\
& = & \llbracket P\rrbracket_{\llbracket q:=|0\rangle\rrbracket_\mu}
\end{array}$$

□

Lemma 5.5. The following two clauses hold.

(i) $\llbracket g^U(r)\rrbracket_\mu = \llbracket r\rrbracket_{\llbracket U[\bar{q}]\rrbracket_\mu}$.

(ii) $\llbracket g^U(P)\rrbracket_\mu \Rightarrow \llbracket P\rrbracket_{\llbracket U[\bar{q}]\rrbracket_\mu}$.

Proof: The proof is similar to that of Lemma 5.4 except for the treatment of two cases for statement (i).

- $r \equiv \mathbb{E}[e]$. We infer that

$$\begin{array}{rcl}
\llbracket g^U(r)\rrbracket_\mu & = & \llbracket \mathbb{E}_{x\sim M[\bar{q}]}[e]\rrbracket_\mu \qquad \text{where } M = \langle\{M_0\}, Id\rangle \text{ with } M_0 = U \text{ and } x \text{ is fresh} \\
& = & \sum_\sigma U\mu(\sigma)U^\dagger \cdot \llbracket e\rrbracket_{\sigma[0/x]} \\
& = & \sum_\sigma U\mu(\sigma)U^\dagger \cdot \llbracket e\rrbracket_\sigma \\
& = & \sum_\sigma \llbracket U[\bar{q}]\rrbracket_\mu(\sigma) \cdot \llbracket e\rrbracket_\sigma \\
& = & \llbracket \mathbb{E}[e]\rrbracket_{\llbracket U[\bar{q}]\rrbracket_\mu} \\
& = & \llbracket r\rrbracket_{\llbracket U[\bar{q}]\rrbracket_\mu}
\end{array}$$

- $r \equiv \mathbb{E}_{\bar{x}\sim M[\bar{q}]}[e]$. Suppose $M = \langle\{M_i\}_{i\in I}, l\rangle$. We reason as follows.

$$\begin{array}{rcl}
\llbracket g^U(r)\rrbracket_\mu & = & \llbracket \mathbb{E}_{\bar{x}\sim M'[\bar{q}]}[e]\rrbracket_\mu \quad \text{where } M' = \langle\{M_iU\}_{i\in I}, l\rangle \\
& = & \sum_\sigma \sum_i M_iU\mu(\sigma)U^\dagger M_i^\dagger \cdot \llbracket e\rrbracket_{\sigma[l(i)/\bar{x}]} \\
& = & \sum_\sigma \sum_i M_i\llbracket U[\bar{q}]\rrbracket_\mu(\sigma)M_i^\dagger \cdot \llbracket e\rrbracket_{\sigma[l(i)/\bar{x}]} \\
& = & \llbracket \mathbb{E}_{\bar{x}\sim M[\bar{q}]}[e]\rrbracket_{\llbracket U[\bar{q}]\rrbracket_\mu} \\
& = & \llbracket r\rrbracket_{\llbracket U[\bar{q}]\rrbracket_\mu}
\end{array}$$

□

Lemma 5.6. Let a be an arithmetic expression, σ be any state and $\sigma' := \sigma[\llbracket a\rrbracket_\sigma/x]$. The following four clauses hold, where x is not a bound variable in e, ψ, r and P.

(i) $\llbracket e[a/x]\rrbracket_\sigma = \llbracket e\rrbracket_{\sigma'}$

(ii) $[\![\psi[a/x]]\!]_\sigma = [\![\psi]\!]_{\sigma'}$

(iii) $[\![r[a/x]]\!]_\mu = [\![r]\!]_{[\![x:=a]\!]_\mu}$.

(iv) $[\![P[a/x]]\!]_\mu \Rightarrow [\![P]\!]_{[\![x:=a]\!]_\mu}$.

Proof: The proof is similar to that of Lemma 5.4. As an example, we only consider one case for statement (iii).

Suppose $r \equiv \mathbb{E}_{\bar{y}\sim M[\bar{q}]}[e]$ with $M = \langle \{M_i\}_i, l\rangle$. There are two possibilities.

- $x \in \bar{y}$. In this case, x is a bound variable in r, which contradicts our assumption.
- $x \notin \bar{y}$. Notice that
$$[\![x := a]\!]_\mu(\sigma') \;=\; \sum_\sigma \{\mu(\sigma) \mid \sigma[[\![a]\!]_\sigma/x] = \sigma'\} \tag{9}$$
holds for any μ and σ'. We reason as follows.
$$\begin{array}{rcll}
[\![r[a/x]]\!]_\mu & = & [\![\mathbb{E}_{\bar{y}\sim M[\bar{q}]}[e[a/x]]]\!]_\mu & \\
& = & \sum_\sigma \sum_i M_i\mu(\sigma)M_i^\dagger [\![e[a/x]]\!]_{\sigma[l(i)/\bar{y}]} & \\
& = & \sum_\sigma \sum_i M_i\mu(\sigma)M_i^\dagger [\![e]\!]_{\sigma[l(i)/\bar{y}][[\![a]\!]_{\sigma[l(i)/\bar{y}]}/x]} & \text{by statement (i)} \\
& = & \sum_i M_i \sum_\sigma \mu(\sigma)M_i^\dagger [\![e]\!]_{\sigma[[\![a]\!]_\sigma/x][l(i)/\bar{y}]} & \\
& = & \sum_i M_i \sum_{\sigma'} \sum_\sigma \{\mu(\sigma) \mid \sigma[[\![a]\!]_\sigma/x] = \sigma'\} M_i^\dagger [\![e]\!]_{\sigma'[l(i)/\bar{y}]} & \\
& = & \sum_i M_i \sum_{\sigma'} [\![x := a]\!]_\mu(\sigma')M_i^\dagger [\![e]\!]_{\sigma'[l(i)/\bar{y}]} & \text{by (9)} \\
& = & \sum_{\sigma'} \sum_i M_i [\![x := a]\!]_\mu(\sigma')M_i^\dagger [\![e]\!]_{\sigma'[l(i)/\bar{y}]} & \\
& = & [\![\mathbb{E}_{\bar{y}\sim M[\bar{q}]}[e]]\!]_{[\![x:=a]\!]_\mu} & \\
& = & [\![r]\!]_{[\![x:=a]\!]_\mu} &
\end{array}$$

□

Lemma 5.7. (i) $[\![f^M_{x,\bar{q}}(r)]\!]_\mu = [\![r]\!]_{[\![x:=M[\bar{q}]]\!]_\mu}$.

(ii) $[\![f^M_{x,\bar{q}}(P)]\!]_\mu \Rightarrow [\![P]\!]_{[\![x:=M[\bar{q}]]\!]_\mu}$.

Proof: We consider two cases for statement (i); the other cases are easier. Assume that $M = \langle \{M_i\}_i, k\rangle$.

- $r \equiv \mathbb{E}[e]$. We reason as follows.
$$\begin{array}{rcll}
[\![f^M_{x,\bar{q}}(r)]\!]_\mu & = & [\![\mathbb{E}_{x\sim M[\bar{q}]}[e]]\!]_\mu & \\
& = & \sum_\sigma \sum_i M_i\mu(\sigma)M_i^\dagger \cdot [\![e]\!]_{\sigma[l(i)/x]} & \\
& = & \sum_{\sigma'} \sum_\sigma \sum_i \{M_i\mu(\sigma)M_i^\dagger \mid \sigma[l(i)/x] = \sigma'\} \cdot [\![e]\!]_{\sigma'} & \\
& = & \sum_{\sigma'} \sum_\sigma \mu_\sigma(\sigma') \cdot [\![e]\!]_{\sigma'} & \text{where } \mu_\sigma(\sigma') = \sum_i \{M_i\mu(\sigma)M_i^\dagger \mid \sigma[l(i)/x] = \sigma'\} \\
& = & [\![\mathbb{E}[e]]\!]_{\sum_\sigma \mu_\sigma} & \\
& = & [\![\mathbb{E}[e]]\!]_{[\![x:=M[\bar{q}]]\!]_\mu} & \\
& = & [\![r]\!]_{[\![x:=M[\bar{q}]]\!]_\mu} &
\end{array}$$
The second last equality holds because $[\![x := M[\bar{q}]]\!]_{(\sigma,\mu(\sigma))} = \mu_\sigma$.

$$
\begin{array}{rcl}
pc(\textbf{skip}, P) & := & P \\
pc(x := a, P) & := & P[a/x] \\
pc(c_0; c_1, P) & := & pc(c_0, pc(c_1, P)) \\
pc(\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, P) & := & (pc(c_0, P) \wedge \Box b) \oplus (pc(c_1, P) \wedge \Box \neg b) \\
pc(\textbf{abort}, P) & := & \begin{cases} \top & \text{if } P = \Box\textbf{false} \\ \text{undefined} & \text{otherwise} \end{cases} \\
pc(q := |0\rangle, P) & := & h(P) \\
pc(U[\bar{q}], P) & := & g^U(P) \\
pc(x := M[\bar{q}], P) & := & f^M_{x,\bar{q}}(P)
\end{array}
$$

Figure 8: Precondition calculus

- $r \equiv \mathbb{E}_{\bar{y}\sim N[\bar{q}']}[e]$. There are two possibilities. Let us first assume that $x \notin \bar{y}$ and $N = \langle \{N_j\}_j, l\rangle$.

$$
\begin{array}{rcl}
[\![f^M_{x,\bar{q}}(r)]\!]_\mu & = & [\![\mathbb{E}_{x\bar{y}\sim M'[\bar{q}\cup\bar{q}']}[e]]\!]_\mu \text{ with } M' = \langle \{N_j M_i\}_{ij}, k'\rangle \text{ and } k'(ij) = (k(i), l(j)) \\
& = & \sum_\sigma \sum_{ij} N_j M_i \mu(\sigma) M_i^\dagger N_j^\dagger \cdot [\![e]\!]_{\sigma[k(i),l(j)/x\bar{y}]} \\
& = & \sum_\sigma \sum_j N_j (\sum_i M_i \mu(\sigma) M_i^\dagger) N_j^\dagger \cdot [\![e]\!]_{\sigma[k(i)/x][l(j)/\bar{y}]} \\
& = & \sum_{\sigma'} \sum_\sigma \sum_j N_j (\sum_i \{M_i \mu(\sigma) M_i^\dagger \mid \sigma[k(i)/x] = \sigma'\}) N_j^\dagger \cdot [\![e]\!]_{\sigma'[l(j)/\bar{y}]} \\
& = & \sum_{\sigma'} \sum_\sigma \sum_j N_j \mu_\sigma(\sigma') N_j^\dagger \cdot [\![e]\!]_{\sigma'[l(j)/\bar{y}]} \\
& & \quad \text{where } \mu_\sigma(\sigma') = \sum_i \{M_i \mu(\sigma) M_i^\dagger \mid \sigma[k(i)/x] = \sigma'\} \\
& = & \sum_{\sigma'} \sum_j N_j \sum_\sigma \mu_\sigma(\sigma') N_j^\dagger \cdot [\![e]\!]_{\sigma'[l(j)/\bar{y}]} \\
& = & [\![\mathbb{E}_{\bar{y}\sim N[\bar{q}']}[e]]\!]_{\sum_\sigma \mu_\sigma} \\
& = & [\![\mathbb{E}_{\bar{y}\sim N[\bar{q}']}[e]]\!]_{[\![x:=M[\bar{q}]]\!]_\mu} \\
& = & [\![r]\!]_{[\![x:=M[\bar{q}]]\!]_\mu}
\end{array}
$$

If $x \in \bar{y}$, the proof is similar by noting that $\sigma[k(i)/x][l(j)/\bar{y}] = \sigma[l(j)/\bar{y}]$.

□

The next theorem states that the concrete proof system is sound.

Theorem 5.8. Every judgement provable in $\mathcal{S}_c$ is valid.

Proof: We only need to prove that the four new rules [Assgn'], [QInit'], [QUnit'] and [QMeas'] are sound, which follows from Lemmas 5.4 - 5.7; the soundness of all other rules are already shown in Theorem 4.7. □

We can define a precondition calculus to help with syntactic reasoning. Given an assertion P as a postcondition and a loop-free command c, we construct an assertion as a precondition for c, written as $pc(c, P)$. The computation rules for preconditions are given in Figure 8.

Theorem 5.9. Let c be a non-looping command. The following rule is derivable.

$$
\frac{}{\{pc(c,P)\}\ c\ \{P\}}\ \text{[PC]}
$$

Proof: We proceed by induction on the structure of c.

- $c \equiv$ **skip**. Then $pc(c, P) = P$ and we have $\vdash \{P\}\ c\ \{P\}$ by rule [Skip].

- $c \equiv$ **abort**. Then $pc(c, P)$ is only defined for $P = \Box$**false**. In this case, we can infer that $\vdash \{\top\}$ **abort** $\{\Box$**false**$\}$ by rule [Abort].

- $c \equiv x := a$. Then $pc(c, P) = P[a/x]$ and we have $\vdash \{P[a/x]\}\ c\ \{P\}$ by rule [Assgn'].

- $c \equiv c_0; c_1$. Then $pc(c, P) = pc(c_0, pc(c_1, P))$. By induction, we have $\vdash \{pc(c_1, P)\}\ c_1\ \{P\}$ and $\vdash \{pc(c_0, pc(c_1, P))\}\ c_0\ \{pc(c_1, P)\}$. By using rule [Seq], we obtain $\vdash \{pc(c_0, pc(c_1, P))\}\ c\ \{P\}$.

- $c \equiv$ **if** b **then** c_0 **else** c_1. Then $pc(c, P) = (pc(c_0, P) \wedge \Box b) \oplus (pc(c_1, P) \wedge \Box\neg b)$. By induction, we have that $\vdash \{pc(c_0, P)\}\ c_0\ \{P\}$ and $\vdash \{pc(c_1, P)\}\ c_1\ \{P\}$. It is obvious that $pc(c_0, P) \wedge \Box b \Rightarrow pc(c_0, P)$. We can use rule [Conseq] to infer that $\vdash \{pc(c_0, P) \wedge \Box b\}\ c_0\ \{P\}$. Similarly, we have $\vdash \{pc(c_1, P) \wedge \Box\neg b\}\ c_1\ \{P\}$. By applying rule [Cond], we have that $\vdash \{pc(c, P \oplus P)\}\ c\ \{P \oplus P\}$. Since $P \oplus P \Leftrightarrow P$, we use rule [Conseq] again to infer the required result that $\vdash \{pc(c, P)\}\ c\ \{P\}$.

- $c \equiv q := |0\rangle$. A direct consequence of rule [QInit'].

- $c \equiv U[\bar{q}]$. By using rule [QUnit'].

- $c \equiv M[\bar{q}]$. By using rule [QMeas'].

□

6. Example: superdense coding

In this section, we illustrate the use of the proof system $\mathcal{S}_c$ via the example of superdense coding.

Superdense coding was proposed by Bennett and Wiesner in 1992 [4]. It is a quantum communication protocol allowing two classical bits to be encoded in one qubit during a transmission, so it needs only one quantum channel. Such advantage is based on the use of a maximally entangled state, EPR state. An EPR state can be transformed into all the four kinds of EPR states through 1-qubit operations, and these EPR states are mutually orthogonal.

Protocol.. We suppose the sender and the receiver of the communication are Alice and Bob, then the protocol goes as follows:

1. Alice and Bob prepare an EPR state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ together. Then they share the qubits, Alice holding q_0 and Bob holding q_1.
2. Depending on the message Alice wants to send, she applies a gate to her qubit. If Alice wants to send 00, she does nothing. If Alice wants to send 01, she applies the X gate. To send 10, she applies the Z gate. To send 11, she applies both X and Z.
3. Then Alice sends the qubit q_0 to Bob.
4. Bob applies a CNOT operation on q_0, q_1 and a Hadamard operation on q_0 to remove the entanglement.
5. Bob measures q_0 and q_1 to get the message.

After the execution of the protocol above, Bob gets the value that Alice wants to send. The protocol exactly transmits two classical bits of information by sending one qubit from Alice to Bob. A quantum circuit implementing the protocol is illustrated in Figure 9.

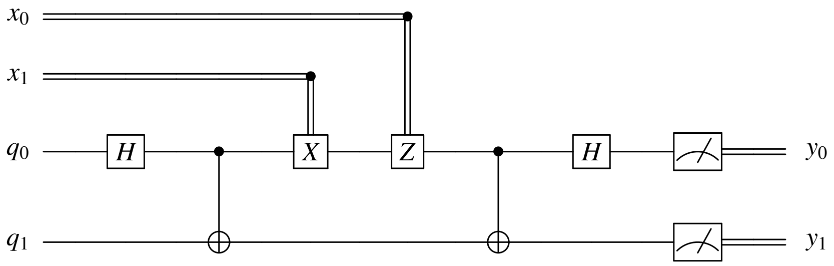

Figure 9: Superdense coding

```
$SC \equiv$
    $q_0 := |0\rangle$;
    $q_1 := |0\rangle$;
    $H[q_0]$;
    $CNOT[q_0 q_1]$;
    **if** $x_1 = 1$ **then** $X[q_0]$;
    **if** $x_0 = 1$ **then** $Z[q_0]$;
    $CNOT[q_0 q_1]$;
    $H[q_0]$;
    $y_0 := M[q_0]$;
    $y_1 := M[q_1]$
      where $M = \langle \{M_0, M_1\}, Id \rangle, M_0 = [|0\rangle], M_1 = [|1\rangle]$
```

Figure 10: The quantum program of implementing superdense coding

The protocol can also be described by the quantum program SC given in Figure 10, where for any pure state $|\varphi\rangle$, we write $[|\varphi\rangle]$ for its density operator $|\varphi\rangle\langle\varphi|$.

According to the operational rules in Figure 2, we can derive the following sequence of transitions, where the initial values of the four classical variables in the first configuration can be arbitrary and we use $*$ to stand for unimportant commands or the values of variables.

$$
\begin{aligned}
& (SC, x_0x_1y_0y_1, [|00\rangle]) \\
\rightarrow\quad & (*, *, [\tfrac{|0\rangle+|1\rangle}{\sqrt{2}}|0\rangle]) \\
\rightarrow\quad & (*, *, [\tfrac{|00\rangle+|11\rangle}{\sqrt{2}}]) \\
\rightarrow\quad & (*, *, [X_0^{x_1}\tfrac{|00\rangle+|11\rangle}{\sqrt{2}}]) \\
\rightarrow\quad & (*, *, [Z_0^{x_0}X_0^{x_1}\tfrac{|00\rangle+|11\rangle}{\sqrt{2}}]) \\
\equiv\quad & \begin{cases}
(*, 00y_0y_1, [\frac{|00\rangle+|11\rangle}{\sqrt{2}}]) & \text{if } x_0 = x_1 = 0 \\
(*, 01y_0y_1, [\frac{|10\rangle+|01\rangle}{\sqrt{2}}]) & \text{if } x_0 = 0, x_1 = 1 \\
(*, 10y_0y_1, [\frac{|00\rangle-|11\rangle}{\sqrt{2}}]) & \text{if } x_0 = 1, x_1 = 0 \\
(*, 11y_0y_1, [\frac{|10\rangle-|01\rangle}{\sqrt{2}}]) & \text{if } x_0 = x_1 = 1
\end{cases} \\
\rightarrow\quad & \begin{cases}
(*, 00y_0y_1, [\frac{|00\rangle+|10\rangle}{\sqrt{2}}]) \\
(*, 01y_0y_1, [\frac{|11\rangle+|01\rangle}{\sqrt{2}}]) \\
(*, 10y_0y_1, [\frac{|00\rangle-|10\rangle}{\sqrt{2}}]) \\
(*, 11y_0y_1, [\frac{|11\rangle-|01\rangle}{\sqrt{2}}])
\end{cases} \\
\rightarrow\quad & \begin{cases}
(*, 00y_0y_1, [|00\rangle]) \\
(*, 01y_0y_1, [|01\rangle]) \\
(*, 10y_0y_1, [|10\rangle]) \\
(*, 11y_0y_1, [|11\rangle])
\end{cases} \\
\rightarrow\quad & \begin{cases}
(*, 000y_1, [|00\rangle]) \\
(*, 010y_1, [|01\rangle]) \\
(*, 101y_1, [|10\rangle]) \\
(*, 111y_1, [|11\rangle])
\end{cases} \\
\rightarrow\quad & \begin{cases}
(\textbf{nil}, 0000, [|00\rangle]) \\
(\textbf{nil}, 0101, [|01\rangle]) \\
(\textbf{nil}, 1010, [|10\rangle]) \\
(\textbf{nil}, 1111, [|11\rangle])
\end{cases}
\end{aligned}
$$

We observe that in each case of the four last configurations, we always have the value of x_0x_1 coincide with y_0y_1 as expected. Indeed, we would like to show that the judgement

$$\{\textbf{true}\}\ SC\ \{\Box(x_0 = y_0 \wedge x_1 = y_1)\} \tag{10}$$

is provable in our concrete proof system. This can be accomplished by a sequence of derivations; for every line of command in Figure 10 we need to prove a Hoare triple. We start from line 10 and proceed backwards. The first step is essentially the same as (5) in Example 5.3. The other five steps can be derived by using the rules [QMeas'], [QUnit'], [Cond] and [Split], as shown in Figure 11.

```
  $\{\mathbf{true}\}$
$SC \equiv$
  $q_0 := |0\rangle;$
  $q_1 := |0\rangle;$
  $H[q_0];$
  $CNOT[q_0q_1];$
  $\{(\mathbb{E}_{y_0y_1\sim M_6[q_0q_1]}[\mathbf{1}_{\psi\wedge x_0=1\wedge x_1=1}] = \mathbb{E}_{y_0y_1\sim M_6[q_0q_1]}[\mathbf{1}_{\mathbf{true}}])$
  $\oplus(\mathbb{E}_{y_0y_1\sim M_4[q_0q_1]}[\mathbf{1}_{\psi\wedge x_0=1\wedge x_1=0}] = \mathbb{E}_{y_0y_1\sim M_4[q_0q_1]}[\mathbf{1}_{\mathbf{true}}])$
  $\oplus(\mathbb{E}_{y_0y_1\sim M_5[q_0q_1]}[\mathbf{1}_{\psi\wedge x_0=0\wedge x_1=1}] = \mathbb{E}_{y_0y_1\sim M_5[q_0q_1]}[\mathbf{1}_{\mathbf{true}}])$
  $\oplus(\mathbb{E}_{y_0y_1\sim M_3[q_0q_1]}[\mathbf{1}_{\psi\wedge x_0=0\wedge x_1=0}] = \mathbb{E}_{y_0y_1\sim M_3[q_0q_1]}[\mathbf{1}_{\mathbf{true}}])\},$
  where $M_6 \equiv \langle\{E_{00}H_{q_0}CNOT_{q_0q_1}Z_{q_0}X_{q_0},\ E_{01}H_{q_0}CNOT_{q_0q_1}Z_{q_0}X_{q_0},$
                 $E_{10}H_{q_0}CNOT_{q_0q_1}Z_{q_0}X_{q_0},\ E_{11}H_{q_0}CNOT_{q_0q_1}Z_{q_0}X_{q_0}\}, Id\rangle,$
  $M_5 \equiv \langle\{E_{00}H_{q_0}CNOT_{q_0q_1}X_{q_0},\ E_{01}H_{q_0}CNOT_{q_0q_1}X_{q_0},$
                 $E_{10}H_{q_0}CNOT_{q_0q_1}X_{q_0},\ E_{11}H_{q_0}CNOT_{q_0q_1}X_{q_0}\}, Id\rangle$
  if $x_1 = 1$ then $X[q_0];$
  $\{(\mathbb{E}_{y_0y_1\sim M_4[q_0q_1]}[\mathbf{1}_{\psi\wedge x_0=1}] = \mathbb{E}_{y_0y_1\sim M_4[q_0q_1]}[\mathbf{1}_{\mathbf{true}}])$
  $\oplus(\mathbb{E}_{y_0y_1\sim M_3[q_0q_1]}[\mathbf{1}_{\psi\wedge x_0=0}] = \mathbb{E}_{y_0y_1\sim M_3[q_0q_1]}[\mathbf{1}_{\mathbf{true}}])\},$
  where $M_4 \equiv \langle\{E_{00}H_{q_0}CNOT_{q_0q_1}Z_{q_0},\ E_{01}H_{q_0}CNOT_{q_0q_1}Z_{q_0},$
                 $E_{10}H_{q_0}CNOT_{q_0q_1}Z_{q_0},\ E_{11}H_{q_0}CNOT_{q_0q_1}Z_{q_0}\}, Id\rangle$
  if $x_0 = 1$ then $Z[q_0];$
  $\{\mathbb{E}_{y_0y_1\sim M_3[q_0q_1]}[\mathbf{1}_{\psi}] = \mathbb{E}_{y_0y_1\sim M_3[q_0q_1]}[\mathbf{1}_{\mathbf{true}}]\},$
  where $M_3 \equiv \langle\{E_{00}H_{q_0}CNOT_{q_0q_1}, E_{01}H_{q_0}CNOT_{q_0q_1}, E_{10}H_{q_0}CNOT_{q_0q_1}, E_{11}H_{q_0}CNOT_{q_0q_1}\}, Id\rangle$
  $CNOT[q_0q_1];$
  $\{\mathbb{E}_{y_0y_1\sim M_2[q_0q_1]}[\mathbf{1}_{\psi}] = \mathbb{E}_{y_0y_1\sim M_2[q_0q_1]}[\mathbf{1}_{\mathbf{true}}]\},$
  where $M_2 \equiv \langle\{E_{00}H_{q_0}, E_{01}H_{q_0}, E_{10}H_{q_0}, E_{11}H_{q_0}\}, Id\rangle$
  $H[q_0];$
  $\{\mathbb{E}_{y_0y_1\sim M_1[q_0q_1]}[\mathbf{1}_{\psi}] = \mathbb{E}_{y_0y_1\sim M_1[q_0q_1]}[\mathbf{1}_{\mathbf{true}}]\}$
  where $M_1 \equiv \langle\{E_{00}, E_{01}, E_{10}, E_{11}\}, Id\rangle,$
  $E_{00} \equiv |00\rangle_{q_0q_1}\langle 00|,\ E_{01} \equiv |01\rangle_{q_0q_1}\langle 01|,\ E_{10} \equiv |10\rangle_{q_0q_1}\langle 10|,\ E_{11} \equiv |11\rangle_{q_0q_1}\langle 11|$
  $y_0 := M[q_0];$
  $\{\mathbb{E}_{y_1\sim M[q_1]}[\mathbf{1}_{\psi}] = \mathbb{E}_{y_1\sim M[q_1]}[\mathbf{1}_{\mathbf{true}}]\}$
  $y_1 := M[q_1]$
    where $M = \langle\{E_0, E_1\}, Id\rangle, E_0 = |0\rangle_{q_1}\langle 0|, E_1 = |1\rangle_{q_1}\langle 1|$
  $\{\Box\psi\}$, where $\psi \equiv x_0 = y_0 \wedge x_1 = y_1$
```

Figure 11: The quantum program with pre- and postconditions

Continue the reasoning until line 1, we obtain the following precondition for SC.

$$\begin{array}{l}
\{(\mathbb{E}_{y_0y_1\sim M_{10}[q_0q_1]}[\mathbf{1}_{\psi\wedge x_0=1\wedge x_1=1}] = \mathbb{E}_{y_0y_1\sim M_{10}[q_0q_1]}[\mathbf{1}_{\textbf{true}}]) \\
\oplus(\mathbb{E}_{y_0y_1\sim M_9[q_0q_1]}[\mathbf{1}_{\psi\wedge x_0=1\wedge x_1=0}] = \mathbb{E}_{y_0y_1\sim M_9[q_0q_1]}[\mathbf{1}_{\textbf{true}}]) \\
\oplus(\mathbb{E}_{y_0y_1\sim M_8[q_0q_1]}[\mathbf{1}_{\psi\wedge x_0=0\wedge x_1=1}] = \mathbb{E}_{y_0y_1\sim M_8[q_0q_1]}[\mathbf{1}_{\textbf{true}}]) \\
\oplus(\mathbb{E}_{y_0y_1\sim M_7[q_0q_1]}[\mathbf{1}_{\psi\wedge x_0=0\wedge x_1=0}] = \mathbb{E}_{y_0y_1\sim M_7[q_0q_1]}[\mathbf{1}_{\textbf{true}}])\}, \\
\text{where } M_{10} \equiv \langle\{E_{1100}, E_{1101}, E_{1110}, E_{1111}\}, f\rangle \\
\qquad\quad\; M_9 \equiv \langle\{E_{1000}, E_{1001}, E_{1010}, E_{1011}\}, f\rangle \\
\qquad\quad\; M_8 \equiv \langle\{E_{0100}, E_{0101}, E_{0110}, E_{0111}\}, f\rangle \\
\qquad\quad\; M_7 \equiv \langle\{E_{0000}, E_{0001}, E_{0010}, E_{0011}\}, f\rangle
\end{array} \tag{†}$$

with

$$f(xyx'y') = xy$$

$$E_{xyx'y'} = |xy\rangle_{q_0q_1}\langle x'y'|$$

for any $x, y, x', y' \in \{0, 1\}$. Note that if we literally apply the rules in Figure 7, we will obtain 16 measurement operators for M_{10}. However, 12 of them are easily simplified to be the zero matrix, thus only four non-zero measurement operators $|11\rangle\langle x'y'|$, where $x', y' \in \{0, 1\}$, are left. Similarly for M_9, M_8 and M_7.

Write P for the assertion in (†). We have seen that

$$\{P\}\; SC\; \{\Box\psi\}. \tag{11}$$

We observe that **true** $\Leftrightarrow P$. This can be seen as follows. Let

$$\begin{array}{rcl}
b_{11} & \equiv & x_0 = 1 \wedge x_1 = 1 \\
b_{10} & \equiv & x_0 = 1 \wedge x_1 = 0 \\
b_{01} & \equiv & x_0 = 0 \wedge x_1 = 1 \\
b_{00} & \equiv & x_0 = 0 \wedge x_1 = 0 \\
P_{11} & \equiv & (\mathbb{E}_{y_0y_1\sim M_{10}[q_0q_1]}[\mathbf{1}_{\psi\wedge b_{11}}] = \mathbb{E}_{y_0y_1\sim M_{10}[q_0q_1]}[\mathbf{1}_{\textbf{true}}]) \\
P_{10} & \equiv & (\mathbb{E}_{y_0y_1\sim M_9[q_0q_1]}[\mathbf{1}_{\psi\wedge b_{10}}] = \mathbb{E}_{y_0y_1\sim M_9[q_0q_1]}[\mathbf{1}_{\textbf{true}}]) \\
P_{01} & \equiv & (\mathbb{E}_{y_0y_1\sim M_8[q_0q_1]}[\mathbf{1}_{\psi\wedge b_{01}}] = \mathbb{E}_{y_0y_1\sim M_8[q_0q_1]}[\mathbf{1}_{\textbf{true}}]) \\
P_{00} & \equiv & (\mathbb{E}_{y_0y_1\sim M_7[q_0q_1]}[\mathbf{1}_{\psi\wedge b_{00}}] = \mathbb{E}_{y_0y_1\sim M_7[q_0q_1]}[\mathbf{1}_{\textbf{true}}])
\end{array}$$

We have $P = P_{11} \oplus P_{10} \oplus P_{10} \oplus P_{00}$. For any POVD μ, it is easy to see that

$$\mu = \mu_{|b_{11}} + \mu_{|b_{10}} + \mu_{|b_{01}} + \mu_{|b_{00}}$$

We have that $\mu_{|b_{11}} \models P_{11}$ because

$$\begin{array}{rl}
& [\![\mathbb{E}_{y_0y_1\sim M_{10}[q_0q_1]}[\mathbf{1}_{\psi\wedge b_{11}}]]\!]_{\mu_{|b_{11}}} \\
= & \sum_\sigma \sum_i M_i\; \mu_{|b_{11}}(\sigma)\; M_i^\dagger \cdot \mathbf{1}_{[\![\psi\wedge b_{11}]\!]_{\sigma[f(i)/y_0y_1]}} \\
& \text{where } i \in \{1100, 1101, 1110, 1100\} \\
= & \sum_\sigma \sum_i M_i\; \mu_{|b_{11}}(\sigma)\; M_i^\dagger \cdot \mathbf{1}_{[\![\psi\wedge b_{11}]\!]_{\sigma[11/y_0y_1]}} \\
= & \sum_\sigma \sum_i M_i\; \mu_{|b_{11}}(\sigma)\; M_i^\dagger \cdot 1 \\
= & \sum_\sigma \sum_i M_i\; \mu_{|b_{11}}(\sigma)\; M_i^\dagger \cdot \mathbf{1}_{[\![\textbf{true}]\!]_{\sigma[11/y_0y_1]}} \\
= & [\![\mathbb{E}_{y_0y_1\sim M_{10}[q_0q_1]}[\mathbf{1}_{\textbf{true}}]]\!]_{\mu_{|b_{11}}}
\end{array}$$

which implies $[\![P_{11}]\!]_{\mu_{|b_{11}}} =$ **true**. Similarly, we can check that $\mu_{|b_{10}} \models P_{10}$, etc. Therefore, we obtain that $\mu \models P$. As μ is arbitrarily chosen, we have verified that **true** $\Leftrightarrow P$. By (11) and rule [Conseq], we finally see that the triple $\{$**true**$\}\; SC\; \{\Box\psi\}$ is provable.

As we can see in (10), the postcondition of that Hoare triple is an assertion about classical variables, even though quantum computation takes place during the execution of the program SC. For such scenario, it is very natural to prove the correctness of programs via an satisfaction-based proof system, rather than an expectation-based one.

7. Conclusion and future work

We have introduced a simple quantum imperative language that has both classical and quantum constructs by extending the language **IMP** studied in depth by Winskel. We have investigated its formal semantics by providing a small-step operational semantics, a denotational semantics and two Hoare-style proof systems: an abstract one and a concrete one. In order to define the semantics, we have used the notion of POVDs to represent the states of programs. Therefore, a program can be considered as a transformer of POVDs. Following the work of Barthe et al, we have designed two satisfaction-based proof systems, as opposed to the usual expectation-based systems. The abstract proof system turns out to be sound and relatively complete, while the concrete one is sound.

As to the future work, at least three immediate improvements are interesting and worth being considered.

- The proof rule [While] is not satisfactory because it involves two sequences of assertions (P_n) and (P'_n). In either the purely classical [11] or purely quantum setting [25], the rule can be elegantly formulated. However, in the presence of both classical and quantum variables, it remains a challenge to find a more concise formulation of the rule.
- The reasoning in Section 6 about the example of superdense coding was done manually. In the future, we would like to embed our program logic in a proof assistant so as to facilitate the reasoning.
- In the concrete proof system, we keep the syntax of assertions simple as long as it can handle some interesting examples. In other words, this is a proof-of-concept system. It is possible to add more constructors for assertions to obtain a more expressive but still sound proof system. We believe that typical quantum algorithms whose correctness can be handled by expectation-based QHLs, e.g., Shor's algorithm and Grover's algorithm, can also be proved in our system. In some applications the correctness consists of several sub-properties. It is more natural and convenient to specify and verify this property in a conjunction form with a satisfaction-based QHL, while in an expectation-based QHL one has to separately encode each sub-property as a Hermitian operator and verify it independently. Having informally said that, if our system is to be implemented in a proof assistant, then we will need to precisely define the syntax of our assertion language and delimit the expressiveness of the system.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. Deng would like to acknowledge the support of the National Natural Science Foundation of China (Nos. 62072176 and 61832015), the Inria-CAS joint project Quasar and and Shanghai Trusted Industry Internet Software Collaborative Innovation Center. Feng was partially supported by the National Key Research and Development Program of China under Grant No. 2018YFA0306704, and the Australian Research Council under Grant No. DP180100691.

References

[1] K. Apt, F. S. De Boer, and E.-R. Olderog. *Verification of sequential and concurrent programs*. Springer Science & Business Media, 2010.

[2] K. R. Apt and E.-R. Olderog. Fifty years of Hoare's logic. *Formal Aspects of Computing*, 31(6):751–807, 2019.

[3] G. Barthe, T. Espitau, M. Gaboardi, B. Grégoire, J. Hsu, and P. Strub. An assertion-based program logic for probabilistic programs. In *Proceedings of the 27th European Symposium on Programming*, volume 10801 of *Lecture Notes in Computer Science*, pages 117–144. Springer, 2018.

[4] C. Bennett and S. Wiesner. Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. *Physical Review Letters*, 69(20):2881–2884, 1992.

[5] R. Chadha, L. Cruz-Filipe, P. Mateus, and A. Sernadas. Reasoning about probabilistic sequential programs. *Theoretical Computer Science*, 379(1-2):142–165, 2007.

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

[7] 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.

[8] Y. Feng, S. Li, and M. Ying. Verification of distributed quantum programs. *arXiv preprint arXiv:2104.14796*, 2021.

[9] Y. Feng and M. Ying. Quantum Hoare logic with classical variables. *arXiv preprint arXiv:2008.06812*, 2020.

[10] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 212–219. ACM, 1996.

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

[12] Y. Kakutani. A Logic for Formal Verification of Quantum Programs. *Lecture Notes in Computer Science*, pages 79–93, 2009.

[13] D. Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 22:328–350, 1981.

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

[15] Y. Li and D. Unruh. Quantum relational Hoare logic with expectations. *arXiv preprint arXiv:1903.08357*, 2019.

[16] A. McIver, C. Morgan, and C. C. Morgan. *Abstraction, refinement and proof for probabilistic systems*. Springer Science & Business Media, 2005.

[17] C. Morgan, A. McIver, and K. Seidel. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 18(3):325–353, 1996.

[18] M. Nielsen and I. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2000.

[19] F. Olmedo, B. L. Kaminski, J.-P. Katoen, and C. Matheja. Reasoning about recursive probabilistic programs. In *2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10. IEEE, 2016.

[20] L. H. Ramshaw. Formalizing the analysis of algorithms. Technical report, Stanford University, 1979.

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

[22] D. Unruh. Quantum Hoare logic with ghost variables. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2019.

[23] J. von Neumann. *States, Effects and Operations: Fundamental Notions of Quantum Theory*. Princeton University Press, 1955.

[24] G. Winskel. *The Formal Semantics of Programming Languages - An Introduction*. The MIT Press, 1993.

[25] M. Ying. Floyd–Hoare logic for quantum programs. *ACM Transactions on Programming Languages and Systems*, 33(6):1–49, 2012.

[26] M. Ying. *Foundations of Quantum Programming*. Morgan Kaufmann, 2016.

[27] M. Ying, L. Zhou, and Y. Li. Reasoning about parallel quantum programs. *arXiv preprint arXiv:1810.11334*, 2018.

[28] L. Zhou, N. Yu, and M. Ying. An applied quantum Hoare logic. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 1149–1162, 2019.

Appendix A. Proof of Theorem 3.4

We proceed by induction on the structure of c. The most difficult case is when $c \equiv \textbf{while}\ b\ \textbf{do}\ c'$ for some command c'. Below we consider this case.

Let $\textbf{While}^n = (\textbf{if}\ b\ \textbf{then}\ c')^n; \textbf{if}\ b\ \textbf{then abort}$ and $\langle c, \sigma, \rho\rangle \rightarrow^n \langle \textbf{nil}, \sigma', \rho'\rangle$ be the sequence of maximal transitions from $\langle c, \sigma, \rho\rangle$ such that the unfolding rule

$$\langle \textbf{while}\ b\ \textbf{do}\ c', \sigma'', \rho''\rangle \rightarrow \langle \textbf{if}\ b\ \textbf{then}\ (c'; \textbf{while}\ b\ \textbf{do}\ c')\ \textbf{else skip}, \sigma'', \rho''\rangle\ ,$$

for any σ'' and ρ'', has been applied at most n times.

Claim: $$[\![\textbf{While}^n]\!]_{(\sigma,\rho)} = \sum_i \{(\sigma_i, \rho_i) \mid \langle c, \sigma, \rho\rangle \rightarrow^{n+1} \langle \textbf{nil}, \sigma_i, \rho_i\rangle\}\ .$$

We prove the above claim by induction on n.

- $n = 0$. On the left hand side, we have

$$[\![\textbf{While}^0]\!]_{(\sigma,\rho)} = [\![\textbf{if}\ b\ \textbf{then abort}]\!]_{(\sigma,\rho)} = \begin{cases} \varepsilon & \text{if } [\![b]\!]_\sigma = \textbf{true} \\ (\sigma,\rho) & \text{if } [\![b]\!]_\sigma = \textbf{false} \end{cases}\ .$$

On the right hand side, we observe that

$$\begin{array}{rcl} \langle \textbf{while}\ b\ \textbf{do}\ c', \sigma, \rho\rangle & \rightarrow & \langle \textbf{if}\ b\ \textbf{then}\ (c'; \textbf{while}\ b\ \textbf{do}\ c')\ \textbf{else skip}, \sigma, \rho\rangle \\ & \rightarrow^* & \begin{cases} \langle (c'; \textbf{while}\ b\ \textbf{do}\ c'), \sigma, \rho\rangle & \text{if } [\![b]\!]_\sigma = \textbf{true} \\ \langle \textbf{skip}, \sigma, \rho\rangle & \text{if } [\![b]\!]_\sigma = \textbf{false} \end{cases} \end{array} \quad \text{(A.1)}$$

The unfolding rule has been used in the first reduction step in (A.1). If $[\![b]\!]_\sigma = \textbf{false}$ then the claim clearly holds. If $[\![b]\!]_\sigma = \textbf{true}$ then the configuration $\langle (c'; \textbf{while}\ b\ \textbf{do}\ c'), \sigma, \rho\rangle$ cannot reduce to any $\langle \textbf{nil}, \sigma'', \rho''\rangle$ without using the unfolding rule again, which means that there is no maximal transition from $\langle c, \sigma, \rho\rangle$ that uses the unfolding rule at most once. It follows that the claim also holds in this case.

- Suppose $n = k + 1$ and the claim holds for some k. On the left hand side, we have

$$\begin{array}{rcl} [\![\textbf{While}^{k+1}]\!]_{(\sigma,\rho)} & = & [\![\textbf{if}\ b\ \textbf{then}\ c'; \textbf{While}^k]\!]_{(\sigma,\rho)} \\ & = & \begin{cases} [\![\textbf{While}^k]\!]_{[\![c']\!]_{(\sigma,\rho)}} & \text{if } [\![b]\!]_\sigma = \textbf{true} \\ (\sigma,\rho) & \text{if } [\![b]\!]_\sigma = \textbf{false} \end{cases} \end{array} \quad \text{(A.2)}$$

On the right hand side, we have the same transitions as in (A.1). If $[\![b]\!]_\sigma = \textbf{false}$ then the claim clearly holds. If $[\![b]\!]_\sigma = \textbf{true}$ then we infer as follows. Since c' is a subterm of c, we know from the hypothesis of the structural induction that

$$[\![c']\!]_{(\sigma,\rho)} = \sum_{j\in J} \{(\sigma_j, \rho_j) \mid \langle c', \sigma, \rho\rangle \rightarrow^* \langle \textbf{nil}, \sigma_j, \rho_j\rangle\} \quad \text{(A.3)}$$

for some set J. It follows that

$$[\![\textbf{While}^k]\!]_{[\![c']\!]_{(\sigma,\rho)}} = \sum_{j\in J} [\![\textbf{While}^k]\!]_{(\sigma_j,\rho_j)}\ . \quad \text{(A.4)}$$

By induction hypothesis on k,

$$[\![\textbf{While}^k]\!]_{(\sigma_j,\rho_j)} = \sum_{i\in I_j}\{(\sigma_i,\rho_i) \mid \langle c,\sigma_j,\rho_j\rangle \rightarrow^{k+1} \langle \textbf{nil},\sigma_i,\rho_i\rangle\} \tag{A.5}$$

for some index set I_j. As a result, when $[\![b]\!]_\sigma = \textbf{true}$, we have

$$\begin{array}{lll} \langle c,\sigma,\rho\rangle & \rightarrow & \langle \textbf{if } b \textbf{ then } (c';c) \textbf{ else skip},\sigma,\rho\rangle \\ & \rightarrow^* & \langle (c';c),\sigma,\rho\rangle \\ & \rightarrow^* & \langle c,\sigma_j,\rho_j\rangle \quad \text{by (A.3)} \\ & \rightarrow^{k+1} & \langle \textbf{nil},\sigma_i,\rho_i\rangle \quad \text{by (A.5)} \end{array} \tag{A.6}$$

for each $j\in J$ and $i\in I_j$. This means that

$$\langle c,\sigma,\rho\rangle \rightarrow^{k+2} \langle \textbf{nil},\sigma_i,\rho_i\rangle \tag{A.7}$$

for each $j\in J$ and $i\in I_j$. Thus, we rewrite (A.5) as follows.

$$[\![\textbf{While}^k]\!]_{(\sigma_j,\rho_j)} = \sum_{i\in I_j}\{(\sigma_i,\rho_i) \mid \langle c,\sigma,\rho\rangle \rightarrow^{k+2} \langle \textbf{nil},\sigma_i,\rho_i\rangle\} \tag{A.8}$$

Combining (A.2), (A.4) and (A.8), we obtain the desired result that

$$[\![\textbf{While}^{k+1}]\!]_{(\sigma,\rho)} = \sum_{j\in J}\sum_{i\in I_j}\{(\sigma_i,\rho_i) \mid \langle c,\sigma,\rho\rangle \rightarrow^{k+2} \langle \textbf{nil},\sigma_i,\rho_i\rangle\}$$

So far we have proved the claim. Then by taking the limit on both sides of the claim, we see that $[\![c]\!]_{(\sigma,\rho)} = \sum_i\{(\sigma_i,\rho_i) \mid \langle c,\sigma,\rho\rangle \rightarrow^* \langle \textbf{nil},\sigma_i,\rho_i\rangle\}$. □