Invariants of Quantum Programs: Characterisations and Generation

Mingsheng Ying
Centre for Quantum Computation and Intelligent Systems (QCIS), University of Technology Sydney, Australia
Tsinghua University, China
Institute of Software, Chinese Academy of Sciences, China
Mingsheng.Ying@uts.edu.au

Shenggang Ying
Centre for Quantum Computation and Intelligent Systems (QCIS), University of Technology Sydney, Australia
Shenggang.Ying@uts.edu.au

Xiaodi Wu
University of Oregon, USA
xiaodiwu@cs.uoregon.edu

Abstract

Program invariant is a fundamental notion widely used in program verification and analysis. The aim of this paper is twofold: (i) find an appropriate definition of invariants for quantum programs; and (ii) develop an effective technique of invariant generation for verification and analysis of quantum programs. Interestingly, the notion of invariant can be defined for quantum programs in two different ways – additive invariants and multiplicative invariants – corresponding to two interpretations of implication in a continuous valued logic: the Łukasiewicz implication and the Gödel implication. It is shown that both of them can be used to establish partial correctness of quantum programs. The problem of generating additive invariants of quantum programs is addressed by reducing it to an SDP (Semidefinite Programming) problem. This approach is applied with an SDP solver to generate invariants of two important quantum algorithms – quantum walk and quantum Metropolis sampling. Our examples show that the generated invariants can be used to verify correctness of these algorithms and are helpful in optimising quantum Metropolis sampling. To our knowledge, this paper is the first attempt to define the notion of invariant and to develop a method of invariant generation for quantum programs.

Categories and Subject Descriptors F.3.2 [*Logics and Meanings of Programs*]: Semantics of Programming Languages - Program Analysis; D.2.4 [*Software Engineering*]: Software/Program Verification

General Terms Algorithms, Theory, Verification.

Keywords Quantum programming, Partial correctness, Program invariants, Inductive assertions, Invariant generation.

1. Introduction

Quantum Programming: Research on quantum programming has already been conducted for two decades, as surveyed in [18, 42, 50]. Several high-level quantum programming languages were de- fined as early as in the later 1990's and early 2000's; for exam- ple, the first quantum programming language QCL was designed by Ö mer [37], a quantum programming language qGCL in the style of Dijkstra's guarded-command language was proposed by Sanders and Zuliani [40], and the first quantum language QPL of the func- tional programming paradigm was defined by Selinger [43]. Moti- vated by the rapid progress in quantum computing hardware in the last few years, several more practical and scalable quantum pro- gramming languages have recently been defined and their compil- ers have been implemented, including Quipper [23], Scaffold [2] and Microsoft's LIQUi|⟩ [48]. Various semantics of quantum pro- gramming languages have also been intensively studied; for exam- ple, a denotational semantics for higher order quantum computa- tion (i.e. quantum lambda calculus with recursion) was discovered by Hasuo and Hoshino [29] and Pagani et al. [38]

Verification of Quantum Programs: Also, various techniques, including program logics [3–5, 14, 30] and model-checking [15, 19, 52], have been extended for verification of quantum programs and quantum cryptographic protocols. For example, the notion of weakest precondition for quantum programs was introduced by D'Hondt and Panangaden in [13]. Furthermore, a logic of the Floyd-Hoare style was developed in [49] for reasoning about both partial and total correctness of quantum programs, and its (relative) completeness was established. A theorem prover was implemented in [36] for quantum Floyd-Hoare logic based on Isabelle/HOL. An algebraic theory of quantum computation was built by Staton in [46] that provides a framework for equational reasoning about quantum programs.

Invariants and Inductive Assertions: As is well-known, the notions of invariant and inductive assertion are essential for verification of programs as well as analysis of algorithms. An invariant of a program at a location is an assertion that is always true when the location is reached. It can be used to establish partial correctness of programs. On the other hand, an assertion is inductive at a location of a program if it is true for the first time the location is reached, and is preserved by every cycle back to the location. A standard method for proving an assertion O to be an invariant is to find an assertion O' that is stronger than O and is inductive [17]. Such a method of proving correctness of programs has also been developed by McIver and Morgan [35] in probabilistic programming.

The ***first contribution*** of this paper is to define the notions of invariant and inductive assertion for quantum programs. A first

thought might be that they can be defined by a straightforward generalisation from classical programs. Actually, this is not the case, and we show that invariants and inductive assertions for quantum programs can be introduced in two different ways, corresponding to two different interpretations of implication in a continuous valued logic [39]:

- Additive invariants, defined by the Łukasiewicz implication: $a \rightarrow_L b = \min(1, 1 - a + b)$ for $a, b \in [0, 1]$; (1)

- Multiplicative invariants, defined by the Gödel implication:

$$a \rightarrow_G b = \min\left(1, \frac{b}{a}\right) \quad \text{for } a, b \in [0, 1]. \tag{2}$$

As in classical programming, we prove in Sections 4 and 5 that both additive and multiplicative invariants can be employed to establish partial correctness of quantum programs, and additively/multiplicatively inductive assertions are additive/multiplicative invariants. These results are obvious for classical programs, but their proofs in the quantum case are much more involved. It seems that the idea of defining invariants using different implications also applies to probabilistic programs, but this is out of the scope of the present paper.

Invariant Generation: Discovering invariants is crucial for verification of programs, but it is a highly nontrivial task [32]. In the literature, there are mainly two approaches to invariant generation for classical programs: *abstract interpretation* and *constraint solving*. The abstract interpretation technique generates an invariant through an approximate symbolic execution of the program until an assertion is reached that remains unchanged by further execution [10, 11]. As its name suggests, the constraint-based technique of Colón et al. [8, 41] reduces invariant generation to a constraint solving problem by encoding the defining conditions of inductive assertions as constraints. Several automatic tools for invariant generation have been developed; for example, the Stanford Invariant Generator StInG [45] implements both the abstract interpretation and constraint-based techniques; and InvGen [27] can more efficiently generate linear arithmetic invariants using the constraint-based technique. Recently, the constraint-based technique was generalised by Katoen et al. [31] for generating invariants of probabilistic programs.

The ***second contribution*** of this paper is to extend the constraint-based approach of Colón et al. [8, 41] to the case of quantum programs. We will only consider how to generate additive invariants, but leave the generation problem of multiplicative invariants for further research. It is shown that additive invariant generation for quantum programs can be reduced to an SDP (Semi-Definite Programming) problem. This approach is applied with an SDP solver to generate invariants of two important quantum algorithms – quantum walk on a circle [1] and quantum Metropolis sampling [47]. We show that the generated invariants can be used to verify correctness of these algorithms and, in particular, are helpful in optimising quantum Metropolis sampling.

Organisation of the Paper: For convenience of the reader, the syntax and operational and denotational semantics of quantum programs written in a quantum extension of the **while**-language are recalled, and the notion of partial and total correctness for quantum programs are reviewed in Section 2. In Section 3, we introduce the notion of super-operator valued transition system (SVTS) and show how it can be used to model the control flow of quantum programs. Then additive and multiplicative invariants and the corresponding inductive assertion maps are defined in Sections 4 and 5, respectively. Generation of additive invariants of quantum programs is considered in Section 6, where two examples are presented showing how additive invariants of quantum walk on an n-circle and quantum Metropolis sampling are generated using the technique developed in Section 6. However, the generation problem of multiplicative invariants is left for future research.

1.1 Preliminaries and Notations

In this subsection, we briefly review several of the basic notions in quantum theory that are frequently used in this paper; for more details, the author can consult the previous quantum programming literature [13, 18, 43, 49, 50].

The state space of a quantum system is a Hilbert space H, i.e. a complex vector space with an inner product that is complete in the sense that every Cauchy sequence has a limit. For finite n, an n-dimensional Hilbert space is essentially the space $\mathbb{C}^n$ of complex vectors. We use Dirac's notation $|\varphi\rangle, |\psi\rangle, \ldots$ to denote vectors. The inner product of $|\varphi\rangle$ and $|\psi\rangle$ is denoted $\langle\varphi|\psi\rangle$. A pure quantum state is represented by a unit vector, i.e. a vector $|\psi\rangle$ with length $\||\psi\rangle\| = \sqrt{\langle\psi|\psi\rangle} = 1$; for example, a qubit (quantum bit) can be in the basis states $|0\rangle$, $|1\rangle$ of 2-dimensional Hilbert space, and it can also be in their superposition $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. A mixed state is represented by an ensemble $\mathsf{E} = \{(p_1, |\psi_1\rangle), \ldots, (p_k, |\psi_k\rangle)\}$ meaning that the system is in state $|\psi_i\rangle$ with probability p_i, where $0 \le p_i$ and $\sum_i p_i = 1$. Intuitively, it can be seen as a quantum generalisation of a probability distribution over states. A crucial mathematical tool in quantum mechanics is (linear) operators on a Hilbert space. The trace of an operator A is the complex number $tr(A) = \sum_i \langle i|A|i\rangle$, where $\{|i\rangle\}$ is an orthonormal basis of the space, and $\langle i|A|i\rangle$ stands for the inner product of $|i\rangle$ and $A|i\rangle$. The external product $A = |\varphi\rangle\langle\psi|$ of two vectors $|\varphi\rangle, |\psi\rangle$ is an operator defined as follows: $A|\eta\rangle = \langle\psi|\eta\rangle|\varphi\rangle$ for each vector $|\eta\rangle$. In the n-dimensional space $\mathbb{C}^n$, an operator is represented by an $n \times n$ complex matrix A and $tr(A) = \sum_i A_{ii}$ (the sum of the entries on the main diagonal); if $|\varphi\rangle, |\psi\rangle \in \mathbb{C}^n$, then its external product is the multiplication $|\varphi\rangle\langle\psi|$ of column vector $|\varphi\rangle$ and the row vector $\langle\psi|$ (the adjoint, i.e. conjugate and transpose of $|\psi\rangle$). An operator A is positive if $\langle\psi|A|\psi\rangle \ge 0$ for every vector $|\psi\rangle$. A positive operator ρ on H is called a *partial density operator* if $tr(\rho) \le 1$; in particular, a density operator ρ is a partial density operator with $tr(\rho) = 1$. We write $\mathsf{D}(\mathsf{H})$ for the set of partial density operators in H. Mathematically, a mixed state represented by ensemble E can also be described by the density operator $\rho_{\mathsf{E}} = \sum_i p_i |\psi_i\rangle\langle\psi_i|$; in particular, a pure state $|\psi\rangle$ can be identified with the density operator $\rho = |\psi\rangle\langle\psi|$. If we consider a partial ensemble E with $\sum_i p_i \le 1$, then ρ_{E} is a partial density operator and its trace is the total probability that the system is in this mixed state: $tr(\rho_{\mathsf{E}}) = \sum_i p_i$. A key difference between mixed quantum states and probability distributions over classical states is that two different ensembles may generate the same density operator, as shown by the simple example: $\rho_1 = 0.4|0\rangle\langle 0| + 0.6|1\rangle\langle 1|$ and $\rho_2 = 0.4|+\rangle\langle +| + 0.6|-\rangle\langle -|$ are different, but $\rho_3 = 0.5|0\rangle\langle 0| + 0.5|1\rangle\langle 1|$ and $\rho_4 = 0.5|+\rangle\langle +| + 0.5|-\rangle\langle -|$ are the same.

An operator U is unitary if $U^\dagger U = UU^\dagger = I$, where $U^\dagger$ is the adjoint of U, and I is the identity operator. It describes the evolution of pure states: $|\psi\rangle \mapsto U|\psi\rangle$. For example, the the Hadamard matrix

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3}$$

is an unitary operator in the 2-dimensional Hilbert space, and it maps qubit states $|0\rangle, |1\rangle$ to $|+\rangle, |-\rangle$, respectively. By a super-operator we mean a mapping E from $\mathsf{D}(\mathsf{H})$ into itself, which is completely positive and satisfies the condition: $\mathrm{tr}(\mathsf{E}(\rho)) \le \mathrm{tr}(\rho)$ for all $\rho \in \mathsf{D}(\mathsf{H})$. It models the evolution of mixed states: $\rho \mapsto$

$\mathcal{E}(\rho)$. In a sense, a super-operator can be seen as a quantum counterpart of a transformation of probability distributions over classical states. The Löwner order $\sqsubseteq$ between two operators A, B is defined as follows: $A \sqsubseteq B$ if and only if $B - A$ is a positive operator. Each super-operator $\mathcal{E}$ has a Kraus representation in terms of operators: $\mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger$ for all density operators, where the set $\{E_i\}$ of operators satisfies the sub-normalisation condition: $\sum_i E_i^\dagger E_i \sqsubseteq I$ (the identity operator); in this case we often write $\mathcal{E} = \sum_i E_i \circ E_i^\dagger$. $\mathcal{E}$ is defined by a single operator E, i.e. $\mathcal{E} = E \circ E^\dagger$ (or more precisely, $\mathcal{E}(\rho) = E\rho E^\dagger$ for all density operators ρ), then we simply write $\mathcal{E} = E$; for instance, a unitary operator U can be seen as the super-operator $\mathcal{E}_U = U \circ U^\dagger$. Two super-operators $\mathcal{E}$ and $\mathcal{F}$ are equivalent, written $\mathcal{E} \equiv \mathcal{F}$, if $\mathrm{tr}(\mathcal{E}(\rho)) = \mathrm{tr}(\mathcal{F}(\rho))$ for all $\rho \in \mathcal{D}(\mathcal{H})$. The Schrödinger-Heisenberg dual $\mathcal{E}^*$ of super-operator $\mathcal{E} = \sum_i E_i \circ E_i^\dagger$ is defined as follows: $\mathcal{E}^*(A) = \sum_i E_i^\dagger A E_i$ for all operators A.

The way to extract information about a quantum system is called a quantum measurement. In quantum computation, measurement is usually used to read out a computational result. Mathematically, a measurement is modelled as a set of operators $M = \{M_m\}$ with $\sum_m M_m^\dagger M_m = I$. If we perform a measurement M on a system in state ρ, then an outcome m is observed with probability $p_m = \mathrm{tr}(M_m \rho M_m^\dagger)$, and after that, the system will be in state $M_m \rho M_m^\dagger / p_m$. Here, a major difference between classical and quantum systems occurs. Measuring a classical system does not change its state, whereas the state of a quantum system is changed after measuring it. For example, the measurement on a qubit in the computational basis is $M = \{M_0, M_1\}$, where $M_0 = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, $M_1 = |1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$. If we perform M on a qubit in (mixed) state $\rho = \frac{2}{3}|0\rangle\langle 0| + \frac{1}{3}|+\rangle\langle +| = \frac{1}{6}\begin{pmatrix} 5 & 1 \\ 1 & 1 \end{pmatrix}$ then the probability that we get outcome 0 is $p(0) = tr(M_0 \rho M_0) = \frac{5}{6}$ and then the qubit is in state $|0\rangle$. Similarly, outcome 1 is obtained with probability $p(1) = \frac{1}{6}$ and after that the qubit is in $|1\rangle$.

2. Quantum Programs

For convenience of the reader, in this section, we recall the syntax and operational and denotational semantics of quantum programs as well as the notions of partial and total correctness for quantum programs; for more details we refer to [49] (see also Chapters 3 and 4 of [50]).

2.1 Syntax

We consider a simple quantum programming language, the quantum extension of **while**-language.

Definition 2.1 (Syntax [49, 50])**.** *We assume a set Var of quantum variables. Quantum programs are defined by the following grammar:*

$$P ::= \mathbf{skip} \mid P_1; P_2 \quad (4)$$

$$\mid q := |0\rangle \quad (5)$$

$$\mid \overline{q} := U[\overline{q}] \quad (6)$$

$$\mid \mathbf{if}\ (\square_m M[\overline{q}] = m \to P_m)\ \mathbf{fi} \quad (7)$$

$$\mid \mathbf{while}\ M[\overline{q}] = 1\ \mathbf{do}\ P\ \mathbf{od} \quad (8)$$

In the above definition, $q \in Var$ and $\overline{q} \subseteq Var$. We write $\mathcal{H}_q$ for the state Hilbert space of quantum variable q; for example, if $type(q) = \mathbf{Bool}$ (respectively, $\mathbf{Int}$), then $\mathcal{H}_q$ is the 2-dimensional (respectively, infinite-dimensional) Hilbert space with $\{|0\rangle, |1\rangle\}$ (respectively, $\{|n\rangle : n \in \mathbb{Z}\}$) as an orthonormal basis. The program constructs in (4) are similar to their counterparts in a classical or probabilistic programming language. The initialisation (5) sets quantum variable q to a basis state $|0\rangle$. The statement (6) means that unitary transformation U is performed on quantum register $\overline{q}$, leaving the states of the variables not in $\overline{q}$ unchanged. The program construct (7) is a quantum generalisation of classical case statement. In executing it, measurement $M = \{M_m\}$ is performed on $\overline{q}$, and then a subprogram P_m is selected to be executed next according to the outcome of measurement. An essential difference between (7) and a classical case statement is that the state of program variables is changed after performing the measurement in the former, whereas it is not changed after checking the guards in the latter. The statement (8) is a quantum generalisation of **while**-loop. The measurement in (8) has only two possible outcomes 0, 1. If the outcome 0 is observed, then the program terminates, and if the outcome 1 occurs, the program executes the subprogram P and continues the loop. The only difference between quantum loop (8) and a classical loop is that checking the loop guard in the latter does not change the state of program variables, but it changes the state in the former.

Let us first consider a quantum variant of a simple probabilistic program given in [31] so that the reader can better understand the difference between a probabilistic program and a quantum program.

Example 2.1 (Three Quantum Dials)**.** *Suppose that a slot machine has three dials d_1, d_2, d_3 and two suits $\heartsuit$ and $\diamondsuit$, and spins the dials independently so that they come to rest on each of the suits with equal probability. It can be modelled as a probabilistic program:*

$$flip \equiv (d_1 := \heartsuit \oplus_{\frac{1}{2}} d_1 := \diamondsuit); (d_2 := \heartsuit \oplus_{\frac{1}{2}} d_2 := \diamondsuit); (d_3 := \heartsuit \oplus_{\frac{1}{2}} d_3 := \diamondsuit)$$

where $P_1 \oplus_p P_2$ stands for a probabilistic choice which chooses to execute P with probability p and to execute Q with probability $1 - p$. A quantum variant of flip can be defined as follows:

$$qflip \equiv H[d_1];\ H[d_2];\ H[d_3]$$

where H is the Hadamard operator defined by equation (3) in the 2-dimensional Hilbert space $\mathcal{H}_2$ with $\{|\heartsuit\rangle, |\diamondsuit\rangle\}$ as an orthonormal basis. It is worth noting that the program qflip also spins the dials, but does it in a quantum way modelled by the Hadamard "coin-tossing" operator H.

Quantum walks [1] have been successfully applied in a class of important quantum algorithms, including quantum simulation [20]. Next, we consider a quantum walk on an n-circle with an absorbing boundary at position 1. It gives us an interesting example of quantum program with **while**-loop.

Example 2.2 (Quantum Walk)**.** *Let $\mathcal{H}_c$ be the coin space, the 2-dimensional Hilbert space with orthonormal basis states $|L\rangle$ and $|R\rangle$, indicating directions Left and Right, respectively. Let $\mathcal{H}_p$ be the n-dimensional Hilbert space with orthonormal basis states $|0\rangle, |1\rangle, ..., |n-1\rangle$, where vector $|i\rangle$ denotes position i for each $0 \le i < n$. The state space of the walk is $\mathcal{H} = \mathcal{H}_c \otimes \mathcal{H}_p$. The initial state is $|L\rangle|0\rangle$. Each step of the walk consists of:*

1. *Measure the position of the system to see whether it is 1. If the outcome is "yes", then the walk terminates, otherwise, it continues. The measurement is $M = \{M_{yes}, M_{no}\}$, where*

$$M_{yes} = |1\rangle\langle 1|,\ M_{no} = I_p - M_{yes} = \sum_{i=1} |i\rangle\langle i|$$

and I_p is the identity operator in the position space $\mathcal{H}_p$;

2. *The Hadamard "coin-tossing" operator H is applied in the coin (or direction) space $\mathcal{H}_c$;*

3. *The shift operator S defined by* $S|L,i\rangle = |L, i\ominus 1\rangle$, $S|R,i\rangle = |R, i\oplus 1\rangle$ *for* $i = 0,1,\ldots,n-1$ *is performed on the space* H. *Intuitively, the system walks one step left or right according to the direction state. Here,* $\oplus$ *and* $\ominus$ *stand for addition and subtraction modulo n, respectively. The operator S can be equivalently written as*

$$S = \sum_{i=0}^{n-1} |L\rangle\langle L| \otimes |i \ominus 1\rangle\langle i| + \sum_{i=0}^{n-1} |R\rangle\langle R| \otimes |i \oplus 1\rangle\langle i|.$$

Using the language described in Definition 2.1, this walk can be written as the quantum program:

$$QW \equiv\ c := |L\rangle; p := |0\rangle; \textbf{while}\ M[p] = no\ \textbf{do}\ c := H[c]; c, p := S[c, p]\ \textbf{od}$$

An essential difference between the quantum walk and a classical random walk is that the coin (or direction) variable c can be in a superposition of $|L\rangle$ *and* $|R\rangle$ *like* $|+\rangle = \frac{1}{\sqrt{2}}(|L\rangle + |R\rangle)$, *and thus the walker is moving left and right "simultaneously"; for example,*

$$\frac{1}{\sqrt{2}}(|L\rangle + |R\rangle)|i\rangle \to \frac{1}{\sqrt{2}}(|L\rangle|i \ominus 1\rangle + |R\rangle|i \oplus 1\rangle).$$

This means that if the walker is currently at position i, then after one step she/he will be at both position $i \ominus 1$ *and* $i \oplus 1$.

2.2 Semantics

We now define the operational semantics of quantum programs. For each quantum program P, we write $var(P)$ for the set of quantum variables occurring in P and $\mathsf{H}_P = \bigotimes_{q\in var(P)} \mathsf{H}_q$ for the state Hilbert space of P, where H_q is the state space of q. A quantum configuration is a pair $\langle P,\rho\rangle$, where P is a program or the termination symbol $\downarrow$, and $\rho \in \mathsf{D}(\mathsf{H}_P)$ denotes the state of quantum variables. A transition between configurations $\langle P,\rho\rangle \to \langle P', \rho'\rangle$ means that after executing quantum program P one step in state ρ, the state of quantum variables becomes ρ', and P' is the remainder of P still to be executed; in particular, if $P' = \downarrow$, then P terminates in state ρ'.

Definition 2.2 (Operational Semantics [49, 50])**.** *The operational semantics of quantum programs is the transition relation* $\to$ *between configurations defined by the transition rules in Figure 1:*

(SK) $\langle \textbf{skip}, \rho\rangle \to \langle\downarrow, \rho\rangle$

(IN) $\langle q := |0\rangle, \rho\rangle \to \langle\downarrow, \rho_0^q\rangle$ where

$$\rho_0^q = \begin{cases} |0\rangle_q\langle 0|\rho|0\rangle_q\langle 0| + |0\rangle_q\langle 1|\rho|1\rangle_q\langle 0| & \text{if } type(q) = \textbf{Bool}, \\ \sum_{n=-\infty}^{\infty} |0\rangle_q\langle n|\rho|n\rangle_q\langle 0| & \text{if } type(q) = \textbf{Int}. \end{cases}$$

(UT) $\langle q := U[q], \rho\rangle \to \langle\downarrow, U\rho U^\dagger\rangle$

(SC) $\langle P_1, \rho\rangle \to \langle P_1', \rho'\rangle \Rightarrow \langle P_1; P_2, \rho\rangle \to \langle P_1'; P_2, \rho'\rangle$ where $\downarrow; P_2 = P_2$.

(IF) $\langle \textbf{if}\ (\square m \cdot M[q] = m \to P_m)\ \textbf{fi}, \rho\rangle \to \langle P_m, M_m\rho M_m^\dagger\rangle$ for each possible outcome m of measurement $M = \{M_m\}$.

(L0) $\langle \textbf{while}\ M[q] = 1\ \textbf{do}\ P\ \textbf{od}, \rho\rangle \to \langle\downarrow, M_0\rho M_0^\dagger\rangle$

(L1) $\langle \textbf{while}\ M[q] = 1\ \textbf{do}\ P\ \textbf{od}, \rho\rangle \to \langle P; \textbf{while}\ M[q] = 1\ \textbf{do}\ P\ \textbf{od}, M_1\rho M_1^\dagger\rangle$

Figure 1. Transition Rules for Quantum Programs

Remark 2.1. *Probabilities seems to be ignored in the above definition, but actually not; for example, the rule (IF) is equivalent to a probabilistic transition:* $\langle \textbf{if}\ (\square m \cdot M[q] = m \to S_m)\ \textbf{fi}, \rho\rangle \xrightarrow{p_m} \langle S_m, \rho_m\rangle$ *with probability* $p_m = tr(M_m\rho M_m^\dagger)$ *and post-measurement state* $\rho_m = M_m\rho M_m^\dagger/p_m$. *Following [43], we encode both probability* p_m *and density operator* ρ_m *into partial density operator* $M_m\rho M_m^\dagger = p_m\rho_m$, *then the rule can be presented as a non-probabilistic transition. The same idea applies to the rules (L0) and (L1). Such a non-probabilistic transition significantly simplifies the presentation of our results.*

The denotational semantics of quantum programs can be easily defined based on their operational semantics.

Definition 2.3 (Denotational Semantics [49, 50])**.** *For any program* P, *its semantic function is the mapping* $[\![P]\!] : \mathsf{D}(\mathsf{H}_P) \to \mathsf{D}(\mathsf{H}_P)$ *defined by*

$$[\![P]\!](\rho) = \sum \{\!\{ \rho' : \langle P,\rho\rangle \to^* \langle\downarrow, \rho'\rangle \}\!\} \tag{9}$$

for every $\rho \in \mathsf{D}(\mathsf{H}_P)$, *where* $\to^*$ *is the reflexive and transitive closure of* $\to$, *and* $\{\!\{\cdot\}\!\}$ *denotes a multi-set.*

Remark 2.2. *The structural representation of semantic function* $[\![P]\!]$ *was given in [49], Proposition 5.1 (see also [50], Propositions 3.3.1 and 3.3.2). Actually, it can be used as a definition of denotational semantics without reference to operational semantics. Then equation (9) can be recast as a theorem showing that operational and denotational semantics coincide.*

Example 2.3 (Semantics of Three Quantum Dials)**.** *Consider the probabilistic program flip and its quantum variant qflip. A state of flip is a configuration of the slot machine, i.e. a mapping from dials to suits. The semantics of flip is a function that maps each initial state to a uniform distribution of states in which every configurations has probability* $\frac{1}{8}$. *The state Hilbert space of qflip is then* $\mathsf{H}_2^{\otimes 3}$. *For instance, if we write* $|+\rangle = \frac{1}{\sqrt{2}}(|\heartsuit\rangle + |\diamondsuit\rangle)$ *and* $|-\rangle = \frac{1}{\sqrt{2}}(|\heartsuit\rangle - |\diamondsuit\rangle)$ *for the equal superpositions of* $|\heartsuit\rangle$ *and* $|\diamondsuit\rangle$, *then* $[\![qflip]\!](|+,-,+\rangle) = |\heartsuit,\diamondsuit,\heartsuit\rangle$; *if we write* $|W\rangle = \frac{1}{\sqrt{3}}(|\heartsuit,\heartsuit,\diamondsuit\rangle + |\heartsuit,\diamondsuit,\heartsuit\rangle + |\diamondsuit,\heartsuit,\heartsuit\rangle)$ *for the Werner state, a typical entangled state of three qubits, then*

$$[\![qflip]\!](|W\rangle) = \frac{1}{6}(3|\heartsuit,\heartsuit,\heartsuit\rangle + |\heartsuit,\heartsuit,\diamondsuit\rangle + |\heartsuit,\diamondsuit,\heartsuit\rangle + |\diamondsuit,\heartsuit,\heartsuit\rangle + |\heartsuit,\diamondsuit,\diamondsuit\rangle + |\diamondsuit,\heartsuit,\diamondsuit\rangle + |\diamondsuit,\diamondsuit,\heartsuit\rangle + 3|\diamondsuit,\diamondsuit,\diamondsuit\rangle).$$

Here, for simplicity, a pure state $|\psi\rangle$ *is identified with the corresponding density operator* $\rho = |\psi\rangle\langle\psi|$.

2.3 Partial Correctness and Total Correctness

Recall from [13] that a *quantum predicate* is an observable, i.e. a Hermitian operator A with $0 \sqsubseteq A \sqsubseteq I$, where 0 and I are the zero operator and the identity operator, respectively. A quantum predicate is also called an *effect* in the quantum foundations and quantum logic literature. Then correctness of quantum programs can be defined in a standard way:

Definition 2.4 (Correctness Formula, Hoare Triple [49, 50])**.** *A correctness formula (or a Hoare triple) is a statement of the form* $\{A\}P\{B\}$ *where* P *is a quantum program, and both* A, B *are quantum predicates in* H_P, *called the precondition and postcondition, respectively.*

Definition 2.5 (Partial Correctness, Total Correctness [49, 50])**.** *1. The correctness formula* $\{A\}P\{B\}$ *is true in the sense of total correctness, written* $\models_{tot} \{A\}P\{B\}$, *if for all* $\rho \in \mathsf{D}(\mathsf{H}_P)$ *we have:*

$$\mathrm{tr}(A\rho) \le \mathrm{tr}(B[\![P]\!](\rho)). \tag{10}$$

2. The correctness formula $\{A\}P\{B\}$ *is true in the sense of partial correctness, written* $\models_{par} \{P\}S\{Q\}$, *if for all* $\rho \in$

$\mathsf{D}(\mathsf{H}_P)$ *we have:*

$$\mathrm{tr}(B\llbracket P \rrbracket(\rho)) + \mathrm{tr}(\llbracket P \rrbracket(\rho)) = \mathrm{tr}(\rho). \qquad (11)$$

According to the interpretation of observables in quantum mechanics, $\mathrm{tr}(A\rho)$ in equations (10) and (11) can be understood as the expected truth value that input ρ satisfies precondition A, and $\mathrm{tr}(B\llbracket P \rrbracket(\rho))$ the expected truth value that output $\llbracket P \rrbracket(\rho)$ satisfies postcondition B. Moreover, $\mathrm{tr}(\rho) - \mathrm{tr}(\llbracket P \rrbracket(\rho))$ is the probability that program P diverges from input ρ.

Example 2.4 (Correctness of Three Quantum Dials)**.** *We write* $|GHZ\rangle = \frac{1}{\sqrt{2}}(|\heartsuit,\heartsuit,\heartsuit\rangle + |\diamondsuit,\diamondsuit,\diamondsuit\rangle)$ *for the GHZ (Greenberger-Horne-Zeilinger) state, another typical entangled state of three qubits,* $|\Phi\rangle = \frac{1}{2}(|\heartsuit,\heartsuit,\heartsuit\rangle + |\heartsuit,\diamondsuit,\diamondsuit\rangle + |\diamondsuit,\heartsuit,\diamondsuit\rangle + |\diamondsuit,\diamondsuit,\heartsuit\rangle)$*, and* $|\Psi\rangle = |\heartsuit,\heartsuit,\heartsuit\rangle$*. Let* $A = |\Phi\rangle\langle\Phi|$*,* $B = |\Psi\rangle\langle\Psi|$*, and* $C = |GHZ\rangle\langle GHZ|$*. Obviously, A, B and C are all quantum predicates. It is easy to check that*

$$\models_{tot} \{A\}\, qflip\, \{C\}, \qquad \models_{tot} \{\tfrac{1}{4}B\}\, qflip\, \{C\}.$$

This means that if the input is state $|\Phi\rangle$*, then program qflip will certainly output the GHZ state; and if the input is state* $|\Psi\rangle$*, it will output a state* $|\Gamma\rangle$ *that is similar to the GHZ state in the sense:*

$$\Pr(|\Gamma\rangle \text{ and the GHZ state cannot be discriminated}) \geq \frac{1}{4}.$$

Note that partial and total correctness are the same for qflip because it does not contain any loop. The quantum predicates A, B, C are very simple and defined by a particular input/output state. Of course, Definition 2.5 can be used for any quantum predicates, but here we are not going to present more general examples due to the limited space.

3. Super-Operator-Valued Transition Systems

In this section, we introduce the notion of super-operator-valued transition system, which can be seen as a quantum extension of an ordinary transition system. It provides us with a convenient way for modelling the control flow of quantum programs.

3.1 Basic Definitions

Definition 3.1 (Super-operator-Valued Transition Systems)**.** *A super-operator-valued transition system (SVTS for short) is a* 5*-tuple* $\mathsf{S} = \langle \mathsf{H}, L, l_0, \mathsf{T}, \Theta \rangle$*, where:*

1. H *is a Hilbert space, called the state space;*
2. L *is a finite set of locations;*
3. $l_0 \in L$ *is the initial location;*
4. T *is a set of transitions. Each transition* $\tau \in \mathsf{T}$ *is a triple* $\tau = \langle l, l', \mathsf{E} \rangle$*, often written as* $\tau = l \xrightarrow{\mathsf{E}} l'$ *where* $l, l' \in L$ *are the pre- and post-locations of* τ*, respectively, and* E *is a super-operator in* H*. It is required that*

$$\sum \{|\mathsf{E} : l \xrightarrow{\mathsf{E}} l' \in \mathsf{T}|\} \cong \mathsf{I} \qquad (12)$$

for each $l \in L$*, where* I *is the identity super-operator in* H*, i.e.* $\mathsf{I}(\rho) = \rho$ *for all* $\rho \in \mathsf{D}(\mathsf{H})$*;*

5. Θ *is a quantum predicate in* H *denoting the initial condition.*

Remark 3.1. *To avoid the technical problem that an SVTS may contain some terminal location l which does not satisfy equation (12), we can simply add a circle* $l \xrightarrow{\mathbf{1}} l$*.*

The symbol $\{|\cdot|\}$ in equation (12) stands for a multi-set. We also assume that the transition relation $\rightarrow$ is countably branching; of equation (12) is well-defined. For any path $\pi = l_1 \xrightarrow{\mathsf{E}_1} l_2 \xrightarrow{\mathsf{E}_2} \cdots \xrightarrow{\mathsf{E}_{n-1}} l_n$ in the transition graph, we write $l_1 \overset{\pi}{\Rightarrow} l_n$ and use E_π to denote the composition of the super-operators along the path, i.e. $\mathsf{E}_\pi = \mathsf{E}_{n-1} \circ \ldots \circ \mathsf{E}_2 \circ \mathsf{E}_1$.

If for every transition $l \xrightarrow{\mathsf{E}} l'$ in S, super-operator E is simply defined by an operator E, i.e. $\mathsf{E}(\rho) = E\rho E^\dagger$ for all density operators ρ, then S is called an operator-valued transition system. We will write $l \xrightarrow{E} l'$ for $l \xrightarrow{\mathsf{E}} l'$ when $\mathsf{E} = E \circ E^\dagger$. In particular, we write $l \xrightarrow{I} l'$ instead of $l \xrightarrow{\mathbf{I}} l'$, where I and $\mathbf{I}$ are the identity operator and identity super-operator, respectively, in H.

Remark 3.2. *An SVTS is essentially a quantum Markov chain defined in [15, 25] together with an initial quantum predicate* Θ*. Each SVTS* S *can be seen as a transition graph with locations as its vertices and transitions as its edges. An operator-valued graph is called a quiver in representation theory [12].*

3.2 Control Flow Graphs of Quantum Programs

Now the control flow graph of a quantum program can be represented by an SVTS. For every quantum program P, we define an SVTS S_P in the state Hilbert space H_P of P by induction on the length of P. This transition system has two designated locations l^P_{in}, l^P_{out}, with the former being the initial location and the latter the exit location.

- $P \equiv \mathbf{skip}$. Then S_P has only two locations l^P_{in}, l^P_{out} and a single transition $l^P_{in} \xrightarrow{I} l^P_{out}$;
- $P \equiv q := |0\rangle$. Let $\{|n\rangle\}$ be an orthonormal basis of H_q. Then S_P has locations l^P_{in}, l^P_{out} together with l^P_n for each basis state $|n\rangle$. The transitions are $l^P_{in} \xrightarrow{E_n} l^P_n$ and $l^P_n \rightarrow l^P_{out}$ for every basis state $|n\rangle$, where $E_n = |0\rangle\langle n|$.
- $P \equiv P_1; P_2$. Suppose that $\mathsf{S}_{P_1}, \mathsf{S}_{P_2}$ are the control flow graphs of subprograms P_1, P_2, respectively. Then S_P is constructed as follows: we identify $l^P_{in} = l^{P_1}_{in}$ and $l^P_{out} = l^{P_2}_{out}$; further set $l^{P_1}_{out} = l^{P_2}_{in}$ and concatenate P_1 and P_2. We;
- $P \equiv \mathbf{if}\ (\square m\, M[\bar{q}] = m \rightarrow P_m)\ \mathbf{fi}$. Suppose that S_{P_m} is the control flow graph of subprogram P_m for every m. Then S_P is constructed as follows: we put all S_{P_m}'s together, and add a new location l^P_{in} and a transition $l^P_{in} \rightarrow l^{P_m}_{in}$ for every m. Furthermore, we identify $l^P_{out} = l^{P_m}_{out}$ for all m;
- $P \equiv \mathbf{while}\ M[\bar{q}] = 1\ \mathbf{do}\ Q\ \mathbf{od}$. We construct S_P from the control flow graph S_Q of subprogram Q as follows: we add two new locations l^P_{in}, l^P_{out} and two transitions $l^P_{in} \xrightarrow{M_0} l^P_{out}$, $l^P_{in} \xrightarrow{M_1} l^Q_{in}$. We identify $l^Q_{out} = l^P_{in}$.

Note that S_P is an operator-valued transition system, i.e. every transition in S_P is of the form $l \xrightarrow{E} l'$ with E being an operator in H_P. This is possible because we choose to depict each initialisation statement $q := |0\rangle$ in P by a family of transitions with operators $E_n = |0\rangle\langle n|$ for basis states $|n\rangle$. On the other hand, we can also use

a single transition with super-operator $\mathsf{E}_0(\rho) = \sum_n |0\rangle\langle n|\rho|n\rangle\langle 0|$ to model the initialisation. Then S_P becomes an SVTS, but the number of locations is significantly reduced.

Example 3.1 (Control Flow Graph of Quantum Walk)**.** *The control flow graph of the quantum walk QW defined in Example 2.2 is given as an SVTS* $\mathsf{S}_{QW} = (\mathsf{H}, L, l_0, \mathsf{T}, \Theta)$, *where:*

- $\mathsf{H} = \mathsf{H}_c \otimes \mathsf{H}_p$;
- $L = \{l_0 = l_{in}, l_1, l_2, l_{out}\}$;
- $\mathsf{T} = \{l_0 \overset{M_{yes}}{\rightarrow} l_{out}, l_0 \overset{M_{no}}{\rightarrow} l_1, l_1 \overset{H}{\rightarrow} l_2, l_2 \overset{S}{\rightarrow} l_0, l_{out} \overset{\mathbf{I}}{\rightarrow} l_{out}\}$;
- $\Theta = |L\rangle\langle L| \otimes |0\rangle\langle 0|$.

that is, for every $l \in L$, the set $\{|\mathsf{E} : l \overset{\mathsf{E}}{\rightarrow} l' \text{ for some } l'|\}$ is finite or countably infinite. Therefore, the summation in the left-hand side

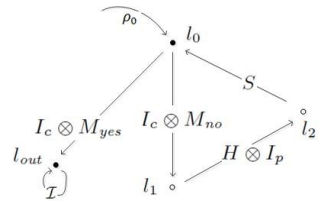

Figure 2. The SVTS of quantum walks on a cycle. Locations with dark dot are chosen as the cut-points later in Example 7.1.

The SVTS S_{QW} is visualised in Figure 2.

4. Additive Invariants and Additively Inductive Assertion Maps

The notions of invariant and inductive assertion map have played a crucial role in program analysis and verification since introduced in the seminal paper [17]. Recall that for a classical program, an invariant at a location l in its control flow graph is an assertion O fulfilling the following condition:

- **C-Invariance**: if an input at the initial location l_0 satisfies the initial condition Θ, then for all paths π, provided π is from l_0 to l, O is always true whenever l is reached through π.

Let S be a classical transition system. A cut-set of S is a subset $C \subseteq L$ of locations such that every cyclic path in S passes through some location in C. Every location $l \in C$ is called a cut-point. A basic path π between two cut-points l and l' is a path that does not pass through any cut-point other than the endpoints. An assertion map for a classical program assigns an assertion $\eta(l)$ to each cut-point $l \in C$ in its control flow graph. It is said to be inductive if it fulfils the following two conditions:

- **C-Initiation**: if an input satisfies the initial condition Θ, then for all basic paths π, provided π is from the initial location l_0 to some cut-point l, $\eta(l)$ is true when π reaches l;
- **C-Consecution**: if $\eta(l)$ is satisfied, then for all basic paths π, provided π is from l to another cut-point l', $\eta(l')$ is satisfied too, when π reaches l'.

In this and next sections, we will generalise the notions of invariant and inductive assertion map into the quantum case. To do this, of course we will deal with an SVTS rather than a classical transition system. This SVTS models the semantics of a quantum program, which is determined by the principles of quantum mechanics. However, there is another key issue to be addressed: in the quantum realm, how to (re-)interpret the implication as well as the quantifier "for all" appearing in the conditions **C-Invariance**, **C-Initiation** and **C-Consection**? As already mentioned in the introduction, it turns out that we can use both the Łukasiewicz system and Gödel system of continuous valued logic for this purpose. This section focuses on quantum invariants and inductive assertion maps defined in Łukasiewicz logic, and those based on Gödel logic will be discussed in the next section.

4.1 Additive Invariants

We first define quantum invariants based on Łukasiewicz logic. To this end, let us introduce an auxiliary notion. A set Π of paths is said to be prime if for each $\pi = l_1 \xrightarrow{\mathsf{E}_1} \cdots \xrightarrow{\mathsf{E}_{n-1}} l_n \in \Pi$, its proper initial segments $l_1 \xrightarrow{\mathsf{E}_1} \cdots \xrightarrow{\mathsf{E}_{k-1}} l_k \notin \Pi$ for all $k < n$.

Definition 4.1 (Additive Invariants)**.** *Let $\mathsf{S} = \langle \mathsf{H}, L, l_0, \mathsf{T}, \Theta \rangle$ be an SVTS and $l \in L$. An additive invariant at location $l \in L$ is a quantum predicate O in state Hilbert space H satisfying the condition:*

- ***A-Invariance**: for any density operator ρ, and for any prime set Π of paths from l_0 to l, we have:*

$$\mathrm{tr}(\Theta\rho) \le 1 - \mathrm{tr}(\mathsf{E}_\Pi(\rho)) + \mathrm{tr}(O\mathsf{E}_\Pi(\rho)) \tag{13}$$

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

The above definition deserves some explanations.

1. Note that if $\Pi_1 \subseteq \Pi_2$ and inequality (13) is true for Π_2 then it is also true for Π_1 (see equation (18) below for a more general argument). Thus, we do not need to check inequality (13) for all prime sets of paths from l_0 to l but only the maximal ones.
2. For each path $\pi \in \Pi$, $\mathsf{E}_\pi(\rho)$ is a partial density operator, but it can be normalised to a density operator $\rho_\pi = \frac{\mathsf{E}_\pi(\rho)}{p_\pi}$, where $p_\pi = \mathrm{tr}(\mathsf{E}_\pi(\rho))$ can be understood as the probability that path π reaches state ρ_π. Furthermore, we have:
$$\mathrm{tr}(O\mathsf{E}_\Pi(\rho)) = \sum_{\pi\in\Pi} \mathrm{tr}(O\mathsf{E}_\pi(\rho)) = \sum_{\pi\in\Pi} p_\pi \cdot \mathrm{tr}(O\rho_\pi).$$
Since $\mathrm{tr}(O\rho_\pi)$ is the (probabilistic) truth value that state ρ_π satisfies quantum predicate O, $\mathrm{tr}(O\mathsf{E}_\Pi(\rho))$ is the expected (or average) truth value that for all paths $\pi \in \Pi$, O is satisfied when π reaches location l. Therefore, here, quantifier "for all" is interpreted as "the expected value according to probability (sub-)distribution $\{p_\pi\}_{\pi\in\Pi}$". This understanding can be seen as a special case of Keisler's integral quantifier in probability logic [33].
3. The quantity $\mathrm{tr}(\mathsf{E}_\Pi(\rho)) = \sum_{\pi\in\Pi} p_\pi$ is the total probability that location l is reached through paths in Π. Here, we see that the condition that Π is prime is necessary; otherwise, a certain probability is calculated repeatedly.
4. Using the Łukasiewicz implication $\to_L$ defined in equation (1), inequality (13) can be rewritten as:
$$\mathrm{tr}(\Theta\rho) \le \mathrm{tr}(\mathsf{E}_\Pi(\rho)) \to_L \mathrm{tr}(O\mathsf{E}_\Pi(\rho))$$
because $0 \sqsubseteq \Theta \sqsubseteq I$ and thus $\mathrm{tr}(\Theta\rho) \le 1$. Combined with items 2 and 3, it shows that inequality (13) is indeed the reinterpretation of **C-Invariance** in the Łukasiewicz system of continuous valued logic.

Of course, as in classical programming, the purpose of introducing invariants for quantum programs is to help us in their verification and analysis. The following theorem shows that additive invariants can actually be used to prove partial correctness of quantum programs.

Theorem 4.1 (Partial Correctness)**.** *Let P be a quantum program and S_P the SVTS defined by P with initial condition Θ. If O is an additive invariant at l^P_{out} in S_P, then $\models_{par} \{\Theta\} P \{O\}$.*

Proof. We write

$$\Pi = \{ \text{paths } \pi : l^P_{in} \Rightarrow l^P_{out} \text{ and } \pi \text{ visits } l^P_{out} \text{ only once} \}.$$

It is easy to see that set Π is prime because any path only visits l^P_{out} once. On the other hand, for any density operator ρ, we claim:

$$[P](\rho) = \sum \{|\mathsf{E}_\pi(\rho) : l^P_{in} \Rightarrow l^P_{out}|\} = \mathsf{E}_\Pi(\rho) \tag{14}$$

We prove equation (14) by induction on the structure of P. We only consider the case of $P \equiv \textbf{while}\ M[\overline{q}] = 1\ \textbf{do}\ Q\ \textbf{od}$ as an example, and other cases are easier (and thus omitted here). Let $\mathsf{E}_i(\rho) = M_i \rho M_i^\dagger$ for all $\rho \in \mathsf{D}(\mathsf{H}_P)$ $(i = 0, 1)$. Then it follows

that

$$[\![P]\!](\rho) = \sum_{n=0}^{\infty} \mathsf{E}_0 \circ ([\![Q]\!] \circ \mathsf{E}_1)^n](\rho)$$
$$= \sum_{n=0}^{\infty} \{\mathsf{E}_\pi \circ \mathsf{E}_1)^n](\rho) : l_{in}^P \xrightarrow{M_0} l_{in}^Q \Rightarrow_{\pi_1} l_{out}^Q \xrightarrow{M_0} l_{out}^P\}$$
$$= \{\mathsf{E}_\pi(\rho) : l_{in}^P \Rightarrow_{\pi} l_{out}^P\}$$

Here, the first equality comes from Proposition 5.1(6) in [49], the second equation from the induction hypothesis on Q as well as linearity of E_0, E_1 and E_π, and the last from the construction of S^P for the quantum **while**-loop P.

Now we are ready to prove the conclusion $\models_{par} \{\Theta\}P\{O\}$. Since O is an additive invariant at l_{out}^P, it follows from equation (14) that for any density operator ρ,

$$\begin{aligned} \mathrm{tr}(\Theta\rho) &\le 1 - \mathrm{tr}(\mathsf{E}_\Pi(\rho)) + \mathrm{tr}(O\mathsf{E}_\Pi(\rho)) \\ &= \mathrm{tr}(O\mathsf{E}_\Pi(\rho)) + (\mathrm{tr}(\rho) - \mathrm{tr}(\mathsf{E}_\Pi(\rho))) \\ &= \mathrm{tr}(O\mathsf{E}_\Pi(\rho)) + (\mathrm{tr}(\rho) - \mathrm{tr}([\![P]\!](\rho))). \end{aligned} \quad (15)$$

We further notice that equation (11) holds for any partial density operator ρ because O, Θ, $[\![P]\!]$ and $\mathrm{tr}(\cdot)$ are all linear. □

4.2 Additively Inductive Assertion Maps

It is not easy to show by its definition that an assertion O is an invariant at a location l for a classical program because we need to check condition **C-Invariant** for every path π from the initial location l_0 to l. It is even harder to do the same for a quantum program directly using Definition 4.1 since we have to verify inequality (13) for every set of paths (rather than a single path) from l_0 to l. In classical programming, inductive assertions give us an effective way for finding and proving program invariants. In this subsection, we define a corresponding notion for quantum programs as a tool for establishing additive invariants.

Definition 4.2 (Assertion Maps). *Given an SVTS $\mathsf{S} = \langle \mathsf{H}, L, l_0, \mathsf{T}, \Theta\rangle$ with a cut-set C (that is, every cyclic path in S passes through some location in C). An assertion map is a mapping η from each cut-point $l \in C$ to a quantum predicate $\eta(l)$ in H.*

For each cut-point $l \in C$, we write Ω_l for the set of all basic paths from l to some cut-point. Moreover, the last location in a path π is denoted by l_π.

Definition 4.3 (Additively Inductive Assertion Maps). *Let η be an assertion map from SVTS $\mathsf{S} = \langle \mathsf{H}, L, l_0, \mathsf{T}, \Theta\rangle$ with a cut-set C. Then η is said to be inductive if it satisfies the following conditions:*

- ***A-Initiation**: for any density operator ρ, we have:*
$$\mathrm{tr}(\Theta\rho) \le 1 - \mathrm{tr}(\mathsf{E}_{\Omega_{l_0}}(\rho)) + \sum_{\pi\in\Omega_{l_0}} \mathrm{tr}(\eta(l_\pi)\mathsf{E}_\pi(\rho)); \quad (16)$$
- ***A-Consecution**: for any density operator ρ, and for each cut-point $l \in C$, we have:*
$$\mathrm{tr}(\eta(l)\rho) \le 1 - \mathrm{tr}(\mathsf{E}_{\Omega_l}(\rho)) + \sum_{\pi\in\Omega_l} \mathrm{tr}(\eta(l_\pi)\mathsf{E}_\pi(\rho)). \quad (17)$$

With an argument similar to that after Definition 4.1, we can see that conditions (**A-Initiation**) and (**A-Consecution**) are essentially the reinterpretations of (**C-Initiation**) and (**C-Consecution**), respectively, in the quantum setting using Łukasiewicz logic.

As its first application, let us see an interesting connection between the notion of additively inductive assertion map and the proof rule for quantum loop in the quantum Floyd-Hoare logic presented in [49].

Example 4.1 (Proof Rule for Quantum Loops). *Consider the general quantum loop (8), of which the quantum walk considered in Examples 2.2 and 3.1 is an example. The proof rule for loop (8) in the quantum Floyd-Hoare logic [49] is given as follows:*

$$\{M_0^\dagger A M_0 + M_1^\dagger B M_1\}\ \mathbf{while}\ M[q] = 1\ \mathbf{do}\ P\ \mathbf{od}\ \{A\}$$
$$\{B\}P\{M_0^\dagger A M_0 + M_1^\dagger B M_1\}$$

Here, we show how this rule can be derived from an additively inductive assertion map. The control flow graph of loop (8) is given as SVTS $\mathsf{S} = (\mathsf{H}, L, l_0, \mathsf{T}, \Theta)$ where H is the state Hilbert space of the loop, $L = \{l_0 = l_{in}, l_1, l_{out}\}$, $\mathsf{T} = \{l_0 \xrightarrow{M_0} l_{out}, l_0 \xrightarrow{M_1} l_1, l_1 \xrightarrow{P} l_0\}$, $[\![P]\!]$ is the denotational semantics of P, and $\Theta = M_0^\dagger A M_0 + M_1^\dagger B M_1$. We choose cut-set $C = \{l_{out}, l_1\}$, and assertion map η is defined by $\eta(l_{out}) = A$ and $\eta(l_1) = B$. It is routine to prove that η is additively inductive whenever
$$\models_{par} \{B\}P\{M_0^\dagger A M_0 + M_1^\dagger B M_1\}.$$

The next theorem shows that the notion of additively inductive assertion can be used to establish additive invariants of quantum programs. Combined with Theorem 4.1, it provides us with a method for verification of quantum programs.

Theorem 4.2 (Additive Invariance). *Let S be an SVTS with cut-set C. If η is an additively inductive assertion map, then for every cut-point $l \in C$, $\eta(l)$ is an additive invariant at l.*

Proof. First of all, we observe that if $\Pi_1 \subseteq \Pi_2$, then

$$\begin{aligned} &1 - \mathrm{tr}(\mathsf{E}_{\Pi_2}(\rho)) + \sum_{\pi\in\Pi_2} \mathrm{tr}(\eta(l_\pi)\mathsf{E}_\pi(\rho)) \\ &= [1 - \mathrm{tr}(\mathsf{E}_{\Pi_1}(\rho)) - \mathrm{tr}(\mathsf{E}_{\Pi_2\setminus\Pi_1}(\rho))] \\ &\quad + \left[\sum_{\pi\in\Pi_1} \mathrm{tr}(\eta(l_\pi)\mathsf{E}_\pi(\rho)) + \sum_{\pi\in\Pi_2\setminus\Pi_1} \mathrm{tr}(\eta(l_\pi)\mathsf{E}_\pi(\rho))\right] \\ &= \left[1 - \mathrm{tr}(\mathsf{E}_{\Pi_1}(\rho)) + \sum_{\pi\in\Pi_1} \mathrm{tr}(\eta(l_\pi)\mathsf{E}_\pi(\rho))\right] \\ &\quad + \sum_{\pi\in\Pi_2\setminus\Pi_1} \mathrm{tr}((\eta(l_\pi) - I)\mathsf{E}_\pi(\rho)) \\ &\le 1 - \mathrm{tr}(\mathsf{E}_{\Pi_1}(\rho)) + \sum_{\pi\in\Pi_1} \mathrm{tr}(\eta(l_\pi)\mathsf{E}_\pi(\rho)) \end{aligned} \quad (18)$$

because $\eta(l_\pi)$ is a quantum predicate and thus $\eta(l_\pi) \sqsubseteq I$.

Now we are going to prove the following claim, which is stronger than equation (13): for any prime set Π of paths from l_0 to some cut-points (not necessarily a single cut-point),

$$\mathrm{tr}(\Theta\rho) \le 1 - \mathrm{tr}(\mathsf{E}_\Pi(\rho)) + \sum_{\pi\in\Pi} \mathrm{tr}(\eta(l_\pi)\mathsf{E}_\pi(\rho)) \quad (19)$$

where ρ is an arbitrary density operator. Let us consider the following two cases:

Case 1: Π is finite. For any path π, its height $h(\pi)$ is defined to be the number of times that π passes through a cut-point. We further define the height of Π as $h(\Pi) = \max_{\pi\in\Pi} h(\pi)$. Then we can prove equation (19) by induction on $h(\Pi)$. For the case of $h(\Pi) = 1$, we have $\Pi \subseteq \Omega_{l_0}$, it follows immediately from equations (16) and (18) that

$$\begin{aligned} \mathrm{tr}(\Theta\rho) &\le 1 - \mathrm{tr}(\mathsf{E}_{\Omega_{l_0}}(\rho)) + \sum_{\pi\in\Omega_{l_0}} \mathrm{tr}(\eta(l_\pi)\mathsf{E}_\pi(\rho)) \\ &\le 1 - \mathrm{tr}(\mathsf{E}_\Pi(\rho)) + \sum_{\pi\in\Pi} \mathrm{tr}(\eta(l_\pi)\mathsf{E}_\pi(\rho)). \end{aligned}$$

In general, assume that $h(\Pi) = n \geq 2$. We can partition:

$$\Pi = \Delta \cup \bigcup_j \Gamma_j$$

such that

1. $h(\Delta) \leq n - 1$;
2. for each j, there exists a path π_j such that $h(\pi_j) = n - 1$ and every path π in Γ_j can be written as $\pi = \pi_j \xrightarrow{E_1} l_1 \ldots \xrightarrow{E_k} l_k \xrightarrow{E_{k+1}} l_{k+1}$ with $l_1, \ldots, l_k \notin C$ and $l_{k+1} \in C$. We write $tail(\pi) = l_{\pi_j} \to l_1 \ldots \to l_k \xrightarrow{E_{k+1}} l_{k+1}$.

We notice that by linearity, equation (17) can be slightly generalised as follows: for any partial density operator ρ,

$$\mathrm{tr}(\eta(l)\rho) \leq \mathrm{tr}(\rho) - \mathrm{tr}(E_{\Omega_l}(\rho)) + \sum_{\pi \in \Omega_l} \mathrm{tr}(\eta(l_\pi) E_\pi(\rho)). \quad (20)$$

Thus, using equation (20) for $l = l_{\pi_j}$ and $\rho = E_{\pi_j}(\rho)$ yields:

$$\mathrm{tr}\left(E_{\Gamma_j}(\rho)\right) - \sum_{\pi \in \Gamma_j} \mathrm{tr}(\eta(l_\pi) E_\pi(\rho))$$

$$= \mathrm{tr}\left(E_{\Gamma_j}\left(E_{\pi_j}(\rho)\right)\right) - \sum_{\pi \in \Gamma_j} \mathrm{tr}\left(\eta(l_\pi) E_{tail(\pi)}\left(E_{\pi_j}(\rho)\right)\right)$$

$$\leq \sum_{\pi' \in \Omega_{l_{\pi_j}}} \mathrm{tr}\left(E_{\pi'}\left(E_{\pi_j}(\rho)\right)\right) - \sum_{\pi' \in \Omega_{l_{\pi_j}}} \mathrm{tr}\left(\eta(l_{\pi'}) E_{\pi'}\left(E_{\pi_j}(\rho)\right)\right)$$

$$\leq \mathrm{tr}\left(E_{\pi_j}(\rho)\right) - \mathrm{tr}\left(\eta(l_{\pi_j}) E_{\pi_j}(\rho)\right). \quad (21)$$

Here, the first inequality comes from equation (18). Note that each $\pi_j \notin \Delta$ because Π is prime. Furthermore, we see that $\Pi' = \Delta \cup \{\text{all } \pi_j\}$ is prime, and $h(\Pi') = n - 1$. Then applying the induction hypothesis to Π', we obtain:

$$\mathrm{tr}(\Theta\rho) \leq 1 - \mathrm{tr}(E_{\Pi'}(\rho)) + \sum_{\pi \in \Pi'} \mathrm{tr}(\eta(l_\pi) E_\pi(\rho))$$

$$= 1 - \left[\mathrm{tr}(E_\Delta(\rho)) + \sum_j \mathrm{tr}\left(E_{\pi_j}(\rho)\right)\right] + \left[\sum_{\pi \in \Delta} \mathrm{tr}(\eta(l_\pi) E_\pi(\rho)) + \sum_j \mathrm{tr}\left(\eta(l_{\pi_j}) E_{\pi_j}(\rho)\right)\right]$$

$$= \left[1 - \mathrm{tr}(E_\Delta(\rho)) + \sum_{\pi \in \Delta} \mathrm{tr}(\eta(l_\pi) E_\pi(\rho))\right] - \sum_j \left[\mathrm{tr}\left(E_{\pi_j}(\rho)\right) - \mathrm{tr}\left(\eta(l_{\pi_j}) E_{\pi_j}(\rho)\right)\right]$$

$$\leq \left[1 - \mathrm{tr}(E_\Delta(\rho)) + \sum_{\pi \in \Delta} \mathrm{tr}(\eta(l_\pi) E_\pi(\rho))\right] - \sum_j \left[\mathrm{tr}\left(E_{\Gamma_j}(\rho)\right) - \sum_{\pi \in \Gamma_j} \mathrm{tr}(\eta(l_\pi) E_\pi(\rho))\right]$$

$$= 1 - \mathrm{tr}(E_\Delta(\rho)) + \sum_j \mathrm{tr}\left(E_{\Gamma_j}(\rho)\right) + \sum_{\pi \in \Delta} \mathrm{tr}(\eta(l_\pi) E_\pi(\rho)) + \sum_j \sum_{\pi \in \Gamma_j} \mathrm{tr}(\eta(l_\pi) E_\pi(\rho))$$

$$\leq 1 - \mathrm{tr}(E_\Pi(\rho)) + \sum_{\pi \in \Pi} \mathrm{tr}(\eta(l_\pi) E_\pi(\rho)).$$

Here, the second inequality follows from equation (21). Thus, we complete the proof for finite Π.

Case 2: Π is infinite. It is clear that Π is countably infinite. So, there exists an infinite sequence $\Pi_1 \subseteq \ldots \subseteq \Pi_m \subseteq \Pi_{m+1} \subseteq \ldots$ such that Π_m is finite for all m, and $\Pi = \bigcup_m \Pi_m$. Thus, with the conclusion for Case 1, we have:

$$\mathrm{tr}(\Theta\rho) \leq x_m \triangleq 1 - \mathrm{tr}(E_{\Pi_m}(\rho)) + \sum_{\pi \in \Pi_m} \mathrm{tr}(\eta(l_\pi) E_\pi(\rho)).$$

On the other hand, it follows from equation (18) that $\{x_m\}$ is a decreasing sequence. Therefore,

$$\mathrm{tr}(\Theta\rho) \leq \lim_{m \to \infty} x_m = 1 - \mathrm{tr}(E_\Pi(\rho)) + \sum_{\pi \in \Pi} \mathrm{tr}(\eta(l_\pi) E_\pi(\rho)).$$

□

5. Multiplicative Invariants and Multiplicatively Inductive Assertion Maps

In the last section, we saw that partial correctness of quantum programs can be proved using additive invariants and additively inductive assertions. In this section, we show that there is another kind of invariants and inductive assertions that can be used for the same purpose; namely multiplicative invariant and multiplicatively inductive assertion. Additive invariants and additively inductive assertions are the quantum extensions of the corresponding notions for classical programs with the implication interpreted in the Łukasiewicz system of continuous valued logic, whereas as we will see, multiplicative invariants and multiplicatively inductive assertions are defined with the Gödel implication in continuous valued logic. The discussions of this section is largely in parallel with the last section.

5.1 Multiplicative Invariants

Definition 5.1 (Multiplicative Invariants)**.** *Let $S = \langle H, L, l_0, T, \Theta \rangle$ be a SVTS and $l \in L$. A multiplicative invariant at location $l \in L$ is a quantum predicate O in state Hilbert space H satisfying the condition:*

- ***M-Invariance**: for any density operator ρ, and for any path π from l_0 to l, we have:*

$$\mathrm{tr}(\Theta\rho) \leq \frac{\mathrm{tr}(OE_\pi(\rho))}{\mathrm{tr}(E_\pi(\rho))}. \quad (22)$$

Using the Gödel implication $\to_G$ defined in equation (2), we can rewrite inequality (22) as follows:

$$\mathrm{tr}(\Theta\rho) \leq \mathrm{tr}(E_\pi(\rho)) \to_G \mathrm{tr}(OE_\pi(\rho)).$$

Consequently, condition (**M-Invariance**) is a quantum extension of (**C-Invariance**) with the implication interpreted in the Gödel system of continuous valued logic. Except the different interpretations of implication, there is another interesting distinction between Definitions 4.1 and 5.1: in equation (13), the paths from l_0 to l were dealt with collectively as a set Π, whereas in equation (22) they were considered individually. This distinction essentially comes from two different interpretations of the universal quantifier "for all": in (22) it was interpreted in a standard way, but in (13), as explained in the paragraph after Definition 4.1, it was interpreted as the expectation with respected to a certain probability distribution.

The following theorem shows that multiplicative invariants can also be used to establish partial correctness of quantum programs.

Theorem 5.1 (Partial Correctness)**.** *Let P be a quantum program and S_P the SVTS defined by P with initial condition Θ. If O is a multiplicative invariant at l_{out} in S_P, then $\models_{par} \{\Theta\} P \{O\}$.*

Proof. Similar to the proof of Theorem 4.1. □

5.2 Multiplicatively Inductive Assertion Maps

Definition 5.2 (Multiplicatively Inductive Assertion Maps)**.** *Let η be an assertion map for SVTS $\mathsf{S} = \langle \mathsf{H}, L, l_0, \mathsf{T}, \Theta\rangle$ with a cut-set C. Then η is said to be multiplicatively inductive if it satisfies the following conditions:*

- ***M-Initiation**: for any density operator ρ, for each cut-point $l \in C$, and for any basic path from l_0 to l, we have:*

$$\mathrm{tr}(\Theta\rho) \le \frac{\mathrm{tr}(\eta(l)\mathsf{E}_\pi(\rho))}{\mathrm{tr}(\mathsf{E}_\pi(\rho))};$$

- ***M-Consection**: for any density operator ρ, for each cut-points $l, l' \in C$, and for any basic path π from l to l', we have:*

$$\mathrm{tr}(\eta(l)\rho) \le \frac{\mathrm{tr}(\eta(l')\mathsf{E}_\pi(\rho))}{\mathrm{tr}(\mathsf{E}_\pi(\rho))}.$$

The conditions (**M-Initiation**) and (**M-Consecution**) can be understood as the quantum extensions of (**C-Initiation**) and (**C-Consecution**), respectively, with the Gödel implication.

We can show that multiplicatively inductive assertions are multiplicative invariants, and thus with Theorem 5.1 they can be used to prove partial correctness of quantum programs.

Theorem 5.2 (Multiplicative Invariance)**.** *Let S be an SVTS with cut-set C. If η is a multiplicatively inductive assertion map, then for every cut-point $l \in C$, $\eta(l)$ is a multiplicative invariant at l.*

Proof. Similar to the proof of Theorem 4.2. □

Conceptually, it is interesting to observe that the fundamental notion of invariant for classical programs can split into two different forms for quantum programs due to the different underlying logics. But it seems that multiplicative invariants are not as useful as additive invariants. On the other hand, whenever a multiplicative invariant is strong enough to establish partial correctness of a quantum program, then we prefer to use it rather than an additive invariant because checking whether a quantum predicate is a multiplicative invariant is easier since we only need to consider every single path, but in order to show an additive invariance, we have to deal with every prime set of paths.

6. Generation of Additively Inductive Invariants

Two kinds of invariants and inductive assertions for quantum programs were introduced in the previous two sections, and it was shown that partial correctness of quantum programs can be proved by finding appropriate invariants. In this section, we further consider how to (automatically) generate additively inductive assertions, which are then additive invariants, according to Theorem 4.2. The generation problem of multiplicative invariants is left for future studies.

The problem of generating additively inductive assertions can be precisely stated as the following:

Problem 6.1 (Generation of Additively Inductive Invariants)**.** *Given an SVTS $\mathsf{S} = \langle \mathsf{H}, L, l_0, \mathsf{T}, \Theta\rangle$ and a cut-set $C \subseteq L$. For each cut-point $l \in C$, find a quantum predicate $\eta(l)$ such that $\eta : l \mapsto \eta(l)$ is an additively inductive map.*

In order to solve the above problem, we are going to generalise the constraint-based technique in [8, 41] to the quantum case. The basic idea is to reduce the above invariant generation problem into an SDP (Semi-Definite Programming) problem by encoding the initiation and consecution conditions in Definition 4.3 for all cut-points $l \in C$ as constraints. Assume that $C = \{l_0, l_1, ..., l_m\}$. We write $O_i = \eta(l_i)$ for $i = 0, 1, ..., m$. Moreover, we write

$$\mathsf{E}^*_{ij} = \sum_{\text{basic path } \pi} \{|\mathsf{E}_\pi : l_i \Rightarrow l_j|\}.$$

for $i, j = 0, 1, ..., m$; in particular, if there is no basic path from l_i to l_j, then E^*_{ij} is the zero super-operator. Then we have:

Theorem 6.1. *Problem 6.1 is equivalent to find complex matrices $O_0, O_1, ..., O_m$ satisfying the following constraints:*

$$0 \sqsubseteq \sum_j \mathsf{E}^*_{0j}(O_j) - \Theta, \tag{23}$$

$$0 \sqsubseteq \sum_j \mathsf{E}^*_{ij}(O_j) + (\mathsf{E}^*_{ii} - \mathsf{I})(O_i) \ (i = 0, 1, ..., m), \tag{24}$$

$$0 \sqsubseteq O_i \sqsubseteq I \ (i = 0, 1, ..., m), \tag{25}$$

Proof. We prove this theorem by three steps of reduction.

Step 1: We first notice that an assertion map η for an SVTS with cut-set C is additively inductive if and only if it is a solution to the following system of constraints:

$$0 \sqsubseteq I + \sum_{\pi \in \Omega_{l_0}} \mathsf{E}^*_\pi(\eta(l_\pi) - I) - \Theta, \tag{26}$$

$$0 \sqsubseteq I + \sum_{\pi \in \Omega_l} \mathsf{E}^*_\pi(\eta(l_\pi) - I) - \eta(l) \text{ for every } l \in C, \tag{27}$$

$$0 \sqsubseteq \eta(l) \sqsubseteq I \text{ for every } l \in C. \tag{28}$$

To show this, we observe that $\mathrm{tr}(B\mathsf{E}(\rho)) = \mathrm{tr}(\mathsf{E}^*(B)\rho)$ for all B and ρ. Indeed, if E has Kraus representation $\mathsf{E}(\rho) = \sum_i E_i \rho E_i^\dagger$ then we have:

$$\mathrm{tr}(B\mathsf{E}(\rho)) = \mathrm{tr}\left(B\sum_i E_i\rho E_i^\dagger\right) = \sum_i \mathrm{tr}\left(BE_i\rho E_i^\dagger\right) = \sum_i \mathrm{tr}\left(E_i^\dagger B E_i \rho\right) = \mathrm{tr}\left(\sum_i E_i^\dagger B E_i \rho\right) = \mathrm{tr}(\mathsf{E}^*(B)\rho).$$

Therefore, it holds that

$$1 - \mathrm{tr}(\mathsf{E}_{\Omega_l}(\rho)) + \sum_{\pi \in \Omega_l} \mathrm{tr}(\eta(l_\pi)\mathsf{E}_\pi(\rho)) - \mathrm{tr}(\eta(l)\rho)$$
$$= \mathrm{tr}(\rho) - \sum_{\pi \in \Omega_l} \mathrm{tr}(\mathsf{E}_\pi(\rho)) + \sum_{\pi \in \Omega_l} \mathrm{tr}(\eta(l_\pi)\mathsf{E}_\pi(\rho)) - \mathrm{tr}(\eta(l)\rho)$$
$$= \mathrm{tr}(\rho) + \sum_{\pi \in \Omega_l} \mathrm{tr}[(\eta(l_\pi) - I)\mathsf{E}_\pi(\rho)] - \mathrm{tr}(\eta(l)\rho)$$
$$= \mathrm{tr}(\rho) + \sum_{\pi \in \Omega_l} \mathrm{tr}[\mathsf{E}^*_\pi((\eta(l_\pi) - I))\rho] - \mathrm{tr}(\eta(l)\rho)$$
$$= \mathrm{tr}\left[I + \sum_{\pi \in \Omega_l} \mathsf{E}^*_\pi(\eta(l_\pi) - I) - \eta(l)\right]\rho.$$

Consequently, inequality (17) is true for all density operators ρ if and only if (27) is valid. Similarly, we can prove that (26) is equivalent to that inequality (16) is true for all ρ. Finally, we notice that constraint (28) comes from the fact that $\eta(l)$ is a quantum predicate.

Step 2: Obviously, inequality (25) is equivalent to (28). If $l = l_i$, then by the definitions of O_j and E^*_{ij} we obtain:

$$\sum_{\pi \in \Omega_l} \mathsf{E}^*_\pi(\eta(l_\pi)) = \sum_j \mathsf{E}^*_{ij}(O_j),$$

$$\sum_{\pi \in \Omega_l} \mathsf{E}_\pi(I)) = \sum_j \mathsf{E}_{ij}(I).$$

Thus, we have

$$0 \sqsubseteq \sum_j \mathsf{E}^*_{0j}(O_j) + A, \quad (29)$$

$$0 \sqsubseteq \sum_j \mathsf{E}^*_{ij}(O_j) + (\mathsf{E}^*_{ii} - \mathsf{I})(O_i) + A_i \ (i = 0, 1, ..., m), \quad (30)$$

$$0 \sqsubseteq \sum_{j \neq i} O_i \sqsubseteq I \ (i = 0, 1, ..., m), \quad (31)$$

where:

$$\begin{cases} A = I - \sum_j \mathsf{E}^*_{0j}(I) - \Theta, \\ A_i = I - \sum_j \mathsf{E}^*_{ij}(I) \ (i = 0, 1, ..., m). \end{cases} \quad (32)$$

Step 3: Now it suffices to show that $A = -\Theta$ and $A_i = 0$ for all i. Let Ω_i be the set of all basic paths starting from l_i. Since there is no basic path which is a prefix of another basic path, every basic path can only occur at most once in computing $\mathrm{tr}(\mathsf{E}_{\Omega_i}(\rho))$. Thus, we have $\mathrm{tr}(\mathsf{E}_{\Omega_i}(\rho)) \leq 1$ for all ρ. Actually, we assert that $\mathrm{tr}(\mathsf{E}_{\Omega_i}(\rho)) = 1$ for all density operators ρ. Otherwise, there exist a density operator σ and a (finite) path π starting form l_i such that

1. π is not a prefix of any basic path in Ω_i;
2. any basic path in Ω_i is not a prefix of π; and
3. $\mathrm{tr}(\mathsf{E}_\pi(\rho)) > 0$.

By fact 2, we see that there is no cut-point in π except the initial location. Suppose $\bar{\pi}$ is an infinite path with π as a prefix. By the definition of cut-set, $\bar{\pi}$ must pass cut-points infinitely many times; otherwise there must be a loop in $\bar{\pi}$ which does not pass any cut-point. This implies that π is a prefix of some basic path in Ω_i, a contradiction.

Furthermore, for all density operator ρ, it follows from $\mathrm{tr}(\mathsf{E}_{\Omega_i}(\rho)) = 1$ that

$$\sum_j \mathrm{tr}(\mathsf{E}_{ij}(\rho)) = \mathrm{tr}(\mathsf{E}_{\Omega_i}(\rho)) = 1.$$

Therefore, $\mathsf{E}_i = \sum_j \mathsf{E}_{ij}$ is trace-preserving. This implies that $\mathsf{E}^*_i = \sum_j \mathsf{E}^*_{ij}$ is unital, i.e. $\sum_j \mathsf{E}^*_{ij}(I) = I$ for every i. Consequently, we have $A = -\Theta$ and $A_i = 0$ for every i. □

It is interesting to compare the constraint problem for generating invariants of quantum programs with that for classical programs. The constraint problems for generating linear and non-linear invariants of classical programs were derived and solved in [8, 41] using different techniques – Farkas's Lemma and Gröbner bases, respectively. However, stipulated by the basic postulates of quantum mechanics, all operations as well as observables involved in a quantum program must be a linear operator. The solutions to the constraint problem for classical programs are coefficients in a template of invariants, which are real numbers. But invariant generation of quantum programs is reduced to a constraint problem over linear operators in a Hilbert space, which are complex matrices when the state Hilbert space is finite-dimensional.

To conclude this section, let us briefly discuss the computational complexity of the constraint problem in Theorem 6.1. Because the problem is naturally an instance of SDP problem, it admits polynomial time algorithms in the size of Hilbert space $d = \dim \mathcal{H}$ and the number of cut-points to solve (note that d is exponential to the number of quantum variables; for example if the programs contain n qubits, then $d = 2^n$). On the other hand, if a classical algorithm for generating quantum invariants *solely* employs density operators as input/output, we claim there is a lower bound polynomial in d, since it costs $\Omega(d^2)$ to input/output a density operator. It is, however, not necessarily a lower bound for the most general approach for generating quantum invariants. For example, it is possible that (1) there exist some succinct representations of quantum invariants which are of size $O(\mathrm{poly}(\log d))$, or (2) there exist classical algorithms which verify the partial correctness of a quantum program by quantum invariants without outputting any explicit quantum invariant. We find exploring these possibilities an interesting open question, and leave it for further research. Moreover, this SDP problem is well-structured such that it not only admits general solutions of polynomial-time in d, but also admits efficient parallel algorithms [28]. It is worthwhile mentioning that one cannot hope to have efficient parallel algorithms for general instances of SDP problems, which contains linear programs as special cases, unless NC = P [34, 44]. The fact that invariants of quantum programs can be generated efficiently in parallel is a promising feature, which makes it amenable to leverage distributed or parallel machines for large-scale quantum programs. In summary, we have:

Corollary 6.1. *The constraint problem for generating invariants of quantum programs in Theorem 6.1 can be solved in polynomial time in the size of Hilbert space and the number of cut-points. Moreover, it also admits an efficient* parallel *algorithm.*

7. Applications

In this section, we present several applications of our results obtained in the previous sections.

7.1 Two Technical Lemmas

First, we need to tailor the general results presented in the previous sections in order to suit the problems in our applications.

The quantum variables in many quantum algorithms are initialised in a special pure state $|\psi\rangle$; that is, $\rho_0 = |\psi\rangle\langle\psi|$. In this case, the initial condition can be chosen as $\Theta = |\psi\rangle\langle\psi|$ (the projection onto the one-dimensional subspace spanned by $|\psi\rangle$), and the following lemma is very useful in helping us to understand physical meaning of the generated invariants.

Lemma 7.1. *Assume that the initial state ρ_0 is a pure state, i.e. $\mathrm{tr}(\rho_0^2) = 1$, and the initial condition $\Theta = \rho_0$. If the program reaches cut-point l_i and the current state is ρ, then $\mathrm{tr}(O_i \rho) = \mathrm{tr}(\rho) = 1$.*

Proof. For the initial density operator ρ_0, any density operator ρ and $i \in \{1, ..., m\}$, from inequalities (16) and (17) we obtain:

$$\mathrm{tr}(\Theta\rho_0) \leq 1 - \mathrm{tr}\left(\sum_j \mathsf{E}_{0j}(\rho_0)\right) + \mathrm{tr}\left(\sum_j O_j \mathsf{E}_{0j}(\rho_0)\right),$$

$$\mathrm{tr}(O_i\rho) \leq 1 - \mathrm{tr}\left(\sum_j \mathsf{E}_{ij}(\rho)\right) + \mathrm{tr}\left(\sum_j O_j \mathsf{E}_{ij}(\rho)\right).$$

As shown in the proof of Theorem 6.1, $\mathrm{tr}\left(\sum_j \mathsf{E}_{ij}(\rho)\right) = 1$ for any density operator ρ. Moreover, since $\mathrm{tr}(\rho_0) = \mathrm{tr}(\rho) = 1$ and $0 \sqsubseteq O_i \sqsubseteq I$, we can derive:

$$1 \leq \mathrm{tr}\left(\sum_j O_j \mathsf{E}_{0j}(\rho_0)\right) \leq \mathrm{tr}\left(\sum_j \mathsf{E}_{0j}(\rho_0)\right) = 1,$$

$$\mathrm{tr}(O_i\rho) \leq \mathrm{tr}\left(\sum_j O_j \mathsf{E}_{ij}(\rho)\right).$$

Then, by induction, it is easy to prove that, at any time, if the system is at cut point l_i, then the current quantum state ρ satisfies $\mathrm{tr}(O_i\rho) = \mathrm{tr}(\rho) = 1$. □

We will employ MATLAB and the CVX package [22] to solve certain SDP problems so that we can find some useful invariants of quantum programs and thus verify correctness of these programs. To this end, we use a single matrix $O = \mathrm{diag}\{O_0, O_1, \cdots, O_m\}$

to represent the observables (Hermitian operators) $O_0, O_1, \cdots, O_m$ at the cut-points. Obviously, O is an $N \times N$ matrix with $N = m \times \dim \mathsf{H}$. Note that each O_i can be rewritten in terms of O; for instance, O_0 can be written as $zOz^\dagger$ where $z = [I, 0, 0, \cdots]$ is a $\dim \mathsf{H} \times N$ matrix. Then inequalities (23) to (25) can be accordingly transferred to inequalities about O.

To solve our problems using the SDP solver, we have to set a optimisation target. Let $O' = \mathrm{diag}\{O'_0, O'_1, \cdots, O'_m\}$. We write $O \sqsubseteq O'$ if $O_i \sqsubseteq O'_i$ for every $i = 0, 1, ..., m$. Note that $O_i \sqsubseteq O'_i$ means that O_i is a quantum predicate stronger than O'_i. So, what we like to do is to find the smallest O_i $(i = 0, 1, ...m)$ with respect to the Löwner order $\sqsubseteq$ that satisfying (23) to (25). The following lemma shows that it only requires us to choose minimising $\mathrm{tr}(O)$ as our optimisation target.

Lemma 7.2. *Suppose that* $O_{\min} = diag\{O_{\min,0}, O_{\min,1}, \cdots, O_{\min,m}\}$ *is the solution of constraints* (23) *to* (25) *with* $[\min \mathrm{tr}(O)]$ *as the objective function. Then* $O_{\min} \sqsubseteq O$ *for any solution* $O = diag\{O_0, O_1, \cdots, O_m\}$ *of constraints* (23) *to* (25).

Proof. For each i, we write H_i for the subspace spanned by all possible states ρ at cut-point l_i. Moreover, let P_{H_i} be the projection onto H_i. We first prove that $O_{\min,i} = P_{H_i}$. By Lemma 7.1, we see that whenever the current location is cut-point l_i and the current state is ρ, then $\mathrm{tr}(\rho O_{\min,i}) = \mathrm{tr}(\rho)$. This means that $\mathrm{supp}(\rho) \subseteq \mathrm{supp}(O_{\min,i})$. Therefore, we have $H_i \subseteq \mathrm{supp}(O_{\min,i})$, or equivalently

$$P_{H_i} \sqsubseteq O_{\min,i}. \tag{33}$$

On the other hand, let $O_P = \mathrm{diag}\{P_{H_1}, \cdots, P_{H_l}\}$. Then by the definition of P_{H_i}, we have $\mathrm{tr}(\rho P_{H_i}) = \mathrm{tr}(\rho O_{\min,i})$ for any possible state ρ at cut point l_i. Therefore, O_P is also a solution of constraints (23) to (25). Obviously, $\mathrm{tr}(O_P) \leq \mathrm{tr}(O_{\min})$, and thus $O_{\min} = O_P$. In a way similar to the proof of equation (33), we can show that $P_{H_i} \sqsubseteq O_i$. Then we complete the proof by combining it with $O_{\min,i} = P_{H_i}$. □

7.2 Quantum Walks on a Cycle

Now we are ready to generate the invariants of quantum walk on a circle and to prove its partial correctness.

Example 7.1 (Generating Additive Invariants for Quantum Walk)**.** *Consider the quantum walk in Examples 2.2 and 3.1. We choose cut-set* $C = \{l_0, l_3\}$ *with* $l_3 = l_{out}$. *Then the problem is to find operators* O_0 *and* O_3 *satisfying the following constraints:*

$$0 \sqsubseteq \mathsf{E}^*_{00}(O_0) + \mathsf{E}^*_{03}(O_3) - \Theta, \tag{34}$$

$$0 \sqsubseteq (\mathsf{E}^*_{00} - \mathsf{I})(O_0) + \mathsf{E}^*_{03}(O_3), \tag{35}$$

$$0 \sqsubseteq (\mathsf{E}^*_{33} - \mathsf{I})(O_3), \tag{36}$$

$$0 \sqsubseteq O_0, O_3 \sqsubseteq I \tag{37}$$

where $\mathsf{E}_{00} = E_{00} \circ E^\dagger_{00}$, $\mathsf{E}_{03} = E_{03} \circ E^\dagger_{03}$, $\mathsf{E}_{33} = \mathsf{I}$, $E_{00} = S(H \otimes I_p)(I_c \otimes M_{no})$, $E_{03} = I_c \otimes M_{yes}$, *and* I_c, I_p *are the identity operators in the coin space* H_c *and position space* H_p, *respectively.*

The solution of the SDP problem with constraints (34) to (37) is as follows:

$$O_0 = |R\rangle\langle R| \otimes |0\rangle\langle 0| + |R\rangle\langle R| \otimes |1\rangle\langle 1| + |\phi\rangle\langle\phi| + \sum_{i=2}^{\infty} |L\rangle\langle L| \otimes |i\rangle\langle i| + \sum_{i=4}^{\infty} |R\rangle\langle R| \otimes |i\rangle\langle i|,$$

$$O_3 = I_c \otimes |1\rangle\langle 1|,$$

where:

$$|\phi\rangle = \frac{1}{\sqrt{2}}(|L\rangle\langle L| \otimes |1\rangle\langle 1| + |R\rangle\langle R| \otimes |3\rangle\langle 3|).$$

n	Number of iterations	Total CPU time (sec)
4	18	0.47
5	19	0.67
10	13	0.98
15	14	5.23
20	17	21.03

Table 1. Running time for generating invariants of quantum random walks. Here, n is the number of nodes on the cycle, i.e. $\dim \mathsf{H}_p = n$ (and thus $\dim \mathsf{H} = 2n$).

With Lemma 7.1, we see that the final state ρ_{out} satisfy $\mathrm{tr}(O_3\rho_{out}) = 1$. This means that if the quantum walk always terminates, the final position is $|1\rangle$. Thus, we proved the partial correctness of this program.

The computer platform that we used to generate invariants O_0 and O_3 is a desktop computer with Intel(R) Core(TM) i5-4570 CPU @ 3.20GHz and 8 GB RAM. The total CPU time and the number of iteration of the CVX program for this example is given in Table 1.

7.3 Quantum Metropolis Sampling

To further show its power, in this subsection, we use our method for generating additive invariants to verify correctness of an important quantum algorithm – quantum Metropolis sampling.

The Metropolis sampling method is a cornerstone of statistical modelling and computation in physics and chemistry. It was successfully extended in [47] for simulation of quantum systems. Quantum Metropolis sampling solves one of the key problems in quantum physics: find a ground state or thermal state for a given quantum system. Roughly speaking, at each round, the algorithm first generates some new state. If the energy of this new state is low, then an update is accepted with probability 1; otherwise it is accepted with a small probability exponential to the difference of energies between the old and new states. To complete this task, quantum Metropolis sampling requires three ancilla quantum systems. According to the description in [47], the size of these ancilla quantum systems depends on the original system. More precisely, the total number of qubits required in this algorithm is at least $3n + 1$, where n is the number of qubits in the original system. Thus, the dimensions of state Hilbert space is at least 2^{3n+1}. Here, we only consider two simple cases of quantum Metropolis sampling algorithm. The purpose of these cases is to find a ground state in a one-qubit quantum system. By the previous discussion, there are only 4 qubits involved in these cases.

Let us first consider a very simple case where a random choice is removed from the original quantum Metropolis sampling.

Example 7.2 (Quantum Metropolis sampling without random choice)**.** *The structure of a quantum Metropolis sampling algorithm for a one-qubit target system is given in Figure 3, where:*

1. $\rho_0 = |0000\rangle\langle 0000|$. *Here, the first qubit is the target qubit, and the other three qubits are the ancilla qubits.*
2. c_0, c_1, c_2, c_3 *are the cut-points with* $c_1 = l_{out}$.
3. E_0 *represents* 8 *sub-routes, which initialise the three ancilla qubits, i.e.*
$$\mathsf{E}_0\Big(\sum_{abcd} \alpha_{abcd} |abcd\rangle\langle abcd|\Big) = \sum_{abcd} \alpha_{abcd} |a000\rangle\langle a000|.$$
4. $U_{E,1}$ *represents the phase estimation on the first ancilla qubit to estimate the energy of original system before updating. In [47], the algorithm is developed for a general quantum system, and thus it is necessary to specify the ground and excited states and*

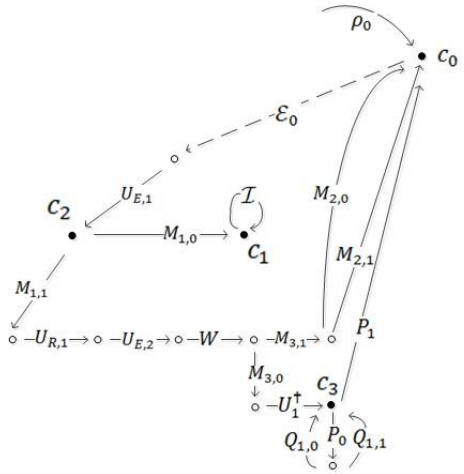

Figure 3. Quantum Metropolis sampling algorithm for a single-qubit target system without random choice of unitary operators.

their corresponding energies. Here, for simplicity we assume that the ground state is $|\psi_0\rangle$ with energy label 0, and the excited state is $|\psi_1\rangle$ with energy label 1. Then $U_{E,1}$ can be written as

$$U_{E,1} = |\psi_0\rangle\langle\psi_0| \otimes I_2 \otimes I_2 \otimes I_2 + |\psi_1\rangle\langle\psi_1| \otimes X \otimes I_2 \otimes I_2,$$

where $X = |0\rangle\langle 1| + |1\rangle\langle 0|$ is the NOT gate, $I_2 = |0\rangle\langle 0| + |1\rangle\langle 1|$ is the identity operator for a single-qubit system. We will examine different $|\psi_0\rangle$ to verify the algorithm.

5. *$\{M_{1,0}, M_{1,1}\}$ represents a projective measurement on the first ancilla qubit to measure the energy of the current system, where*

$$M_{1,0} = I_2 \otimes |0\rangle\langle 0| \otimes I_2 \otimes I_2,$$

$$M_{1,1} = I_2 \otimes |1\rangle\langle 1| \otimes I_2 \otimes I_2.$$

6. *$U_{R,1} = H \otimes I_2 \otimes I_2 \otimes I_2$ updates the target state. In [47], $U_{R,1}$ is chosen from a set of unitary operators at random. Here, this randomisation is removed. But in Example 7.3 we will consider a random choice from $U_{R,2}$ and $U_{R,3}$ at this step.*
7. *$U_{E,2}$ represents the phase estimation on the second ancilla qubit to estimate the energy of original system after updating, i.e.*

$$U_{E,2} = |\psi_0\rangle\langle\psi_0| \otimes I_2 \otimes I_2 \otimes I_2 + |\psi_1\rangle\langle\psi_1| \otimes I_2 \otimes X \otimes I_2.$$

8. *W decides whether the energy of the new state is low. Originally in [47], W depends on the absolute temperature of the original system. For simplicity, we will only examine the following two choices of W in our experiment:*

$$W_1 = I_2 \otimes |0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes I_2 + I_2 \otimes |1\rangle\langle 1| \otimes |1\rangle\langle 1| \otimes I_2 + I_2 \otimes |0\rangle\langle 0| \otimes |1\rangle\langle 1| \otimes X + I_2 \otimes |1\rangle\langle 1| \otimes |0\rangle\langle 0| \otimes X,$$

$$W_2 = I_2 \otimes |0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes I_2 + I_2 \otimes |1\rangle\langle 1| \otimes |1\rangle\langle 1| \otimes I_2 + I_2 \otimes |0\rangle\langle 0| \otimes |1\rangle\langle 1| \otimes H + I_2 \otimes |1\rangle\langle 1| \otimes |0\rangle\langle 0| \otimes X.$$

9. *$\{M_{2,0}, M_{2,1}\}$ represents a projective measurement on the second ancilla qubit to measure the energy of the current system, where*

$$M_{2,0} = I_2 \otimes I_2 \otimes |0\rangle\langle 0| \otimes I_2,$$

$$M_{2,1} = I_2 \otimes I_2 \otimes |1\rangle\langle 1| \otimes I_2.$$

10. *$\{M_{3,0}, M_{3,1}\}$ represents a projective measurement on the second ancilla qubit to find whether the updating is accepted, where*

$\|\psi_0\rangle$	W_1		W_2	
	N_i	T (sec)	N_i	T (sec)
$\|+\rangle$	12	1.95	12	2.10
$\|-\rangle$	12	1.89	12	3.81
$\|\phi_3\rangle$	12	1.87	12	2.12
$\|\phi_4\rangle$	12	1.87	12	2.07
$\|\phi_5\rangle$	17	2.53	17	2.82
$\|0\rangle$	12	1.22	12	1.20
$\|1\rangle$	12	1.18	12	1.22

Table 2. Running time for generating invariants of Example 7.2. Here, N_i is the number of iterations, T is the total CPU time (sec), and $|\phi_3\rangle = 0.8|0\rangle + 0.6|1\rangle$, $|\phi_4\rangle = 0.6|0\rangle - 0.8|1\rangle$, $|\phi_5\rangle = 0.28|0\rangle + 0.96|1\rangle$.

11. *$U_1 = WU_{E,2}U_{R,1}$. If the updating is rejected, $U_1^\dagger$ will undo unitary operators W, $U_{E,2}$, $U_{R,1}$.*
12. $P_1 = |\psi_0\rangle\langle\psi_0| \otimes |0\rangle\langle 0| \otimes I_2 \otimes I_2 + |\psi_1\rangle\langle\psi_1| \otimes |1\rangle\langle 1| \otimes I_2 \otimes I_2$ *and* $P_0 = I - P_1$.
13. $Q_{1,0} = U_1^\dagger(I_2 \otimes I_2 \otimes I_2 \otimes |0\rangle\langle 0|)U_1$ *and* $Q_{1,1} = U_1(I_2 \otimes I_2 \otimes I_2 \otimes |1\rangle\langle 1|)U_1$.

The invariants O_0, O_1, O_2, O_3 generated using our method are as follows:

- For $|\psi_0\rangle \neq |0\rangle$ in Table 2, we have:
  $O_0 = |\psi_1\rangle\langle\psi_1| \otimes |100\rangle\langle 100| + |\psi_0\rangle\langle\psi_0| \otimes |101\rangle\langle 101|$,
  $O_2 = |\psi_0\rangle\langle\psi_0| \otimes |000\rangle\langle 000| + |\psi_1\rangle\langle\psi_1| \otimes |100\rangle\langle 100|$,
  $O_3 = |0100\rangle\langle 0100| + |1100\rangle\langle 1100|$,
  $O_{out} = O_1 = |\psi_0\rangle\langle\psi_0| \otimes |000\rangle\langle 000|$.

- For $|\psi_0\rangle = |0\rangle$, we have: $O_0 = 0$, $O_2 = |0000\rangle\langle 0000|$, $O_3 = 0$, and $O_{out} = O_1 = |0000\rangle\langle 0000|$.

It is interesting to note that these invariants are independent of the choice of W. With Lemma 7.1, we see from O_{out} that the sampling algorithm can find the ground state $|\psi_0\rangle$ with ancilla qubits $|000\rangle$. Since no matter how we set the target system through $U_{E,1}$ and $U_{E,2}$, the output O_{out} is always the ground state $|\psi_0\rangle$, we have verified the partial correctness of the algorithm. The running time of the CVX program for generating these invariants is shown in Table 2.

We now consider an extension of the quantum Metropolis sampling algorithm for a single-qubit target system in Example 7.2. At this time, a random choice between $U_{R,2}$ and $U_{R,3}$ is allowed.

Example 7.3 (Quantum Metropolis sampling with a random choice)**.** *The structure of quantum Metropolis sampling algorithm for a single-qubit target system with a random choice is given in Figure 4, where:*

- *ρ_0, $\mathcal{E}_0$, $U_{E,1}$, $\{M_{1,0}, M_{1,1}\}$, $\{M_{2,0}, M_{2,1}\}$, $\{M_{3,0}, M_{3,1}\}$, $U_{E,2}$, W, and $\{P_0, P_1\}$ are the same as in Example 7.2.*
- *c_0, c_1, c_2, c_3 and c_4 are the cut-points. Note that here we have one more cut-point c_4 than in Example 7.2.*
- $U_{R,2} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$, $U_{R,3} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$.

The factor $\frac{1}{\sqrt{2}}$ occurring before $U_{R,2}$ or $U_{R,3}$ in Figure 4 indi-

$$M_{3,0} = I_2 \otimes I_2 \otimes I_2 \otimes |0\rangle\langle 0|,$$
$$M_{3,1} = I_2 \otimes I_2 \otimes I_2 \otimes |1\rangle\langle 1|.$$

cates that they are chosen with equal probability $\frac{1}{2}$.

- $U_2 = WU_{E,2}U_{R,2}$, *and* $U_3 = WU_{E,2}U_{R,3}$.

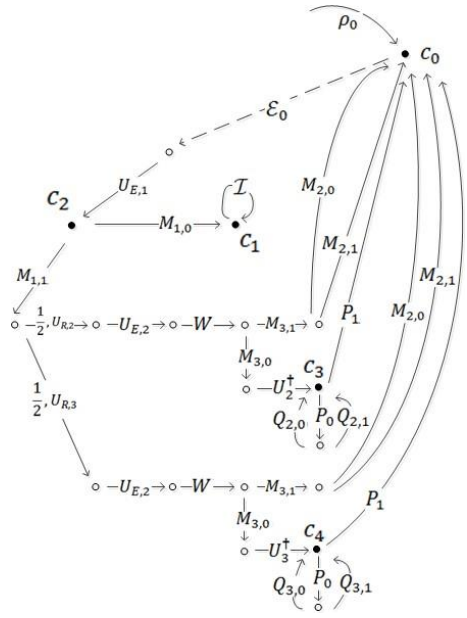

Figure 4. Quantum Metropolis sampling algorithm for a one-qubit target system with random choice of unitary operators.

$\|\psi_0\rangle$	W_1		W_2	
	N_i	T(sec)	N_i	T (sec)
$\|+\rangle$	13	10.87	13	11.18
$\|-\rangle$	13	10.87	13	11.17
$\|\phi_3\rangle$	13	3.84	13	4.35
$\|\phi_4\rangle$	13	3.93	13	4.38
$\|\phi_5\rangle$	12	3.67	12	4.09
$\|0\rangle$	14	2.56	14	2.57
$\|1\rangle$	12	2.31	12	2.39

Table 3. Running time for generating invariants of Example 7.3. Here, N_i is the number of iterations, T is the total CPU time (sec), and $|\phi_3\rangle = 0.8|0\rangle + 0.6|1\rangle$, $|\phi_4\rangle = 0.6|0\rangle - 0.8|1\rangle$, $|\phi_5\rangle = 0.28|0\rangle + 0.96|1\rangle$.

- $Q_{i,0} = U_i^\dagger(I_2 \otimes I_2 \otimes I_2 \otimes |0\rangle\langle 0|)U_i$ *and* $Q_{i,1} = U_i^\dagger(I_2 \otimes I_2 \otimes I_2 \otimes |1\rangle\langle 1|)U_i$.

The invariants O_0, O_1, O_2, O_3, O_4 can be generated using our method. The result is as follows: O_0, O_1, O_2, O_3 are the same as in Example 7.2 and $O_4 = O_3$. Based on these invariants, one can easily show that as stated in [47], random choice between unitary operators $U_{R,i}$ is irrelevant to correctness of the sampling algorithm. The running time of the CVX program for generating invariants of the algorithm in Example 7.3 is given in Table 3.

In [47], the authors claimed that the detailed form of random unitary operators $U_{R,i}$ is not important, provided that the probability of choosing U is equal to the probability of choosing $U^\dagger$. Actually, this statement is automatically verified by our method of invariant generation. It suffices to note that for $i = 0, 1, 2, 3$, O_i in Example 7.2 is the same as in Example 7.3, although these two examples employ different random unitary operators.

Discussion. In Examples 7.2 and 7.3, we have verified correctness of the quantum Metropolis algorithm through automatically generating invariants O_0, O_1, ..., O_4. Here, we further show how these generated invariants can help us to optimise the quantum Metropolis algorithm. Such an optimisation has not been noticed in the previous literature [47] yet.

Firstly, in Example 7.2, we obtained $O_0 = O_3 = 0$ for $|\psi_0\rangle = |0\rangle$. Then by Lemma 7.1, we see that the states at cut-points c_0 and c_3 always vanish (that is, the partial density operators become 0), except the initial state. This means that once the program leaves location c_0, it will never go to either c_0 or c_3; in other words, the only effective route is $c_0 \to c_2 \to c_3$, and the other parts of the program is redundant for the case of $|\psi_0\rangle = |0\rangle$. This information is useful in compilation of the program.

Secondly, the operator W given in [47] is quite complicated and dependent on the current absolute temperature. In the above examples, we showed that W can be replaced by one of the two much simpler operators W_1 and W_2 that can achieve the same computational result. This indicates that our techniques can be used to find operators to simplify the original ones in a quantum program.

8. Conclusion

We introduced the notions of invariant and inductive assertion for quantum programs in two different ways: additive and multiplicative. It was proved that both additive and multiplicative invariants can be used to verify partial correctness of quantum programs. We also showed that additive (resp. multiplicative) invariants can be derived from additively (resp. multiplicatively) inductive assertions. Furthermore, it was demonstrated that the defining conditions for additively inductive assertions can be transformed into certain constraints of an SDP (Semidefinite Programming) problem. Therefore, additive invariants can be generated by using the SDP solvers.

For the further studies, an interesting problem is generation of multiplicative invariants of quantum programs, which was not addressed in Section 6. A problem closely related to invariant generation is synthesis of ranking functions [9], which has been deeply studied for probabilistic programs in the last few years using (super-)martingales (see e.g. [6, 7, 16]). The corresponding problem for quantum programs is also important, but cannot be solved by straightforwardly extending the approach in [6, 7, 16] because the necessary mathematical tool, namely a theory of quantum (super-)martingales is still to be developed. In this paper, we only consider quantum programs written in a quantum extension of the **while**-language. So, another interesting problem is to define invariants for recursive quantum programs and to extend the approach presented in [26] for inter-procedural analysis and verification of quantum programs.

References

[1] D. Aharonov, A. Ambainis, J. Kempe and U. Vazirani, Quantum walks on graphs, In: *Proc. of the 33rd ACM Symposium on Theory of Computing (STOC)*, 2001, pp. 50-59.

[2] A. J. Abhari, A. Faruque, M. Dousti, L. Svec, O. Catu, A. Chakrabati, C.-F. Chiang, S. Vanderwilt, J. Black, F. Chong, M. Martonosi, M. Suchara, K. Brown, M. Pedram and T. Brun, *Scaffold: Quantum Programming Language*, Technical Report TR-934-12, Dept. of Computer Science, Princeton University, 2012.

[3] A. Baltag and S. Smets, LQP: The dynamic logic of quantum information, *Mathematical Structures in Computer Science*, 16(2006)491-525.

[4] O. Brunet and P. Jorrand, Dynamic quantum logic for quantum programs, *International Journal of Quantum Information*, 2(2004)45-54.

[5] R. Chadha, P. Mateus and A. Sernadas, Reasoning about imperative quantum programs, *Electronic Notes in Theoretical Computer Science*, 158(2006)19-39.

[6] A. Chakarov and S. Sankaranarayanan, Probabilistic program analysis with martingales, In: *Proc. of the 25th International Conference*

Computer Aided Verification (CAV), 2013, Springer LNCS 8044, pp. 511-526.

[7] K. Chatterjee, H. F. Fu, P. Novotný and R. Hasheminezhad, Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs, In: *Proceedings of the 43rd ACM Symposium on Principles of Programming Languages (POPL)*, 2016, pp. 327-342

[8] M. A. Colón, S. Sankaranarayanan and H. B. Sipma, Linear invariant generation using non-linear constraint solving, In: *Proc. of the 15th International Conference on Computer Aided Verification (CAV)*, 2003, Springer LNCS, pp. 420-433.

[9] M. Colón and H. Sipma, Synthesis of linear ranking functions, In: *Proc. of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2001, Springer LNCS 2031, pp. 67-81.

[10] P. Cousot and R. Cousot, Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, In: *Proceedings of the 4th ACM Symposium on Principles of Programming Languages (POPL)*, 1977, pp. 238-252.

[11] P. Cousot and N. Halbwachs, Automatic discovery of linear restraints among variables of a program, In: *Proceedings of the 5th ACM Symposium on Principles of Programming Languages (POPL)*, 1978, pp. 84-96.

[12] H. Derksen and J. Weyman, Quiver representations, *Notices of the American Mathematical Society*, 52(2005)200-206.

[13] E. D'Hondt and P. Panangaden, Quantum weakest preconditions, *Mathematical Structures in Computer Science*, 16(2006)429-451.

[14] Y. Feng, R. Y. Duan, Z. F. Ji and M. S. Ying, Proof rules for the correctness of quantum programs, *Theoretical Computer Science*, 386(2007)151-166.

[15] Y. Feng, N. K. Yu and M. S. Ying, Model checking quantum Markov chains, *Journal of Computer and System Sciences*, 79(2013)1181-1198.

[16] L. M. F. Fioriti and H. Hermanns, Probabilistic termination: Soundness, completeness, and compositionality, In: *Proceedings of the 42nd ACM Symposium on Principles of Programming Languages (POPL)*, 2015, pp. 489-501.

[17] R. W. Floyd, Assigning meanings to programs, In: *Proceedings of the Symposium on Mathematical Aspects of Computer Science*, 1967, 19-33.

[18] S. Gay, Quantum programming languages: survey and bibliography, *Mathematical Structures in Computer Science*, 16(2006)581-600.

[19] S. Gay, R. Nagarajan, and N. Panaikolaou. QMC: A model checker for quantum systems, In: *Proceedings of the 20th International Conference on Computer Aided Verification (CAV)*, 2008, Springer LNCS 5123, pp. 543-547.

[20] I.?M. Georgescu, S. Ashhab and F. Nori, Quantum simulation, *Reviews of Modern Physics*, 86(2014)153-185.

[21] S. M. German and B. Wegbreit, A synthesiser of inductive assertions, *IEEE Transactions on Software Engineering*, 1(1975)68-75.

[22] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.0 beta. http://cvxr.com/cvx, September 2013.

[23] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger and B. Valiron, Quipper: A scalable quantum programming language, *Proceedings of the 34th ACM Conference on Programming Language Design and Implementation (PLDI)*, 2013, pp. 333-342.

[24] F. Gretz, J. -P. Katoen and A. McIver, Prinsys - On a quest for probabilistic loop invariants, in: *Proc. 10th International Conference on Quantitative Evaluation of Systems (QEST)*, 2013, Springer LNCS 8054, pp.193-208.

[25] S. Gudder, Quantum Markov chains, *Journal of Mathematical Physics*, 49(2008) art. no. 072105.

[26] S. Gulwani, S. Srivastava and R. Venkatesan, Program analysis as constraint solving, In: *Proceedings of 29th ACM Conference on Programming Language Design and Implementation (PLDI)*, 2008, pp. 281-292.

[27] A. Gupta and A. Rybalchenko, InvGen: an efficient invariant generator, *Proceedings of the 21st International Conference on Computer Aided Verification (CAV)*, 2009, pp. 634-640.

[28] G. Gutoski and X. Wu, Parallel approximation of min-max problems with applications to classical and quantum zero-sum games, *Computational Complexity*, 22(2013) 385-428.

[29] I. Hasuo and N. Hoshino, Semantics of higher-order quantum computation via Geometry of Interaction, In: *Proceedings of the 26th IEEE Symposium on Logic in Computer Science (LICS)*, 2011, 237-246.

[30] Y. Kakutani, A logic for formal verification of quantum programs, in: *Proceedings of the 13th Asian Computing Science Conference (ASIAN)*, 2009, Springer LNCS 5913, pp. 79-93.

[31] J. -P. Katoen, A. McIver, L. Meinicke and C. C. Morgan, Linear-invariant generation for probabilistic programs - Automated support for proof-based methods, in: *Proc. 17th International Symposium on Static Analysis (SAS)*, 2010, Springer LNCS 6337, pp. 390-406.

[32] S. Katz and Z. Manna, Logical analysis of programs, *Communications of the ACM*, 19(1976)188-206.

[33] H. J. Keisler, Probability quantifiers, In: J. Barwise and S. Feferman (eds.), *Model Theoretic Logics*, Springer, 1985, pp. 509-556.

[34] N. Megiddo, A note on approximate linear programming, *Information Processing Letters*, 42(1992) 42-53.

[35] A. McIver and C. C. Morgan, *Abstraction, Refinement and Proof of Probabilistic Systems*, Springer, Heidelberg, 2004.

[36] Y. J. Li, T. Liu, S. L. Wang, N. J. Zhan and M. S. Ying, A theorem prover for quantum Hoare logic and its applications, *http://arxiv.org/pdf/1601.03835.pdf*

[37] B. Ömer, *Structured Quantum Programming*, Ph.D thesis, Technical University of Vienna, 2003.

[38] M. Pagani, P. Selinger and B. Valiron, Applying quantitative semantics to higher-order quantum computing, In: *Proceedings of 41st ACM Symposium on Principles of Programming Languages (POPL)*, 2014, pp. 647-658.

[39] N. Rescher, *Many-Valued Logic*, McGraw-Hill, 1969.

[40] J. W. Sanders and P. Zuliani, Quantum programming, In: *Proceedings of 5th International Conference on Mathematics of Program Construction (MPC)*, 2000, Springer LNCS 1837, Springer pp. 88-99.

[41] S. Sankaranarayanan, H. B. Sipma and Z. Manna, Non-linear loop invariant generation using Gröbner bases, in: *Proceedings of the 31st ACM Symposium on Principles of Programming Languages (POPL)*, 2004, pp. 318-329.

[42] P. Selinger, A brief survey of quantum programming languages, In: *Proc. of 7th International Symposium on Functional and Logic Programming*, 2004, Springer LNCS 2998, pp. 1-6.

[43] P. Selinger, Towards a quantum programming language, *Mathematical Structures in Computer Science 14*, (2004) 527-586.

[44] M. J. Serna, Approximating linear programming is log-space complete for P, *Information Processing Letters*, 37(1991) 233-236.

[45] Stanford Invariant Generator, http://theory.stanford.edu/ srirams/Software/sting.html.

[46] S. Staton, Algebraic effects, linearity, and quantum programming languages, In: *Proceedings of 42nd ACM Symposium on Principles of Programming Languages (POPL)*, 2015, pp. 395-406.

[47] K. Temme, T. J. Osborne, K. G. Vollbrecht, D. Poulin, and F. Verstraete. Quantum Metropolis sampling. *Nature*, 471(2011) 87-90.

[48] D. Wecker and K. M. Svore, LIQUi|⟩: A software design architecture and domain-specific language for quantum computing, http://research.microsoft.com/pubs/209634/1402.4467.pdf.

[49] M. S. Ying, Floyd-hoare logic for quantum programs, *ACM Transactions on Programming Languages and Systems*, 33(2011) art no. 19, pp. 1-49.

[50] M. S. Ying, *Foundations of Quantum Programming*, Morgan-Kaufmann, 2016.

[51] M. S. Ying and Y. Feng, Quantum loop programs, *Acta Informatica*, 47(2010)221-250.

[52] M. S. Ying, Y. J. Li, N. K. Yu and Y. Feng, Model-checking linear-time properties of quantum systems, *ACM Transactions on Computational Logic*, 15(2014) art. no. 22.

[53] M. S. Ying, N. K. Yu, Y. Feng and R. Y. Duan, Verification of quantum programs, *Science of Computer Programming*, 78(2013)1679-1700.