Alternation in Quantum Programming: From Superposition of Data to Superposition of Programs

Mingsheng Ying, Nengkun Yu and Yuan Feng*

Abstract

We extract a novel quantum programming paradigm - superposition of programs - from the design idea of a popular class of quantum algorithms, namely quantum walk-based algorithms. The generality of this paradigm is guaranteed by the universality of quantum walks as a computational model. A new quantum programming language QGCL is then proposed to support the paradigm of superposition of programs. This language can be seen as a quantum extension of Dijkstra's GCL (Guarded Command Language). Surprisingly, alternation in GCL splits into two different notions in the quantum setting: classical alternation (of quantum programs) and quantum alternation, with the latter being introduced in QGCL for the first time. Quantum alternation is the key program construct for realizing the paradigm of superposition of programs.

The denotational semantics of QGCL are defined by introducing a new mathematical tool called the guarded composition of operator-valued functions. Then the weakest precondition semantics of QGCL can straightforwardly derived. Another very useful program construct in realizing the quantum programming paradigm of superposition of programs, called quantum choice, can be easily defined in terms of quantum alternation. The relation between quantum choices and probabilistic choices is clarified through defining the notion of local variables. We derive a family of algebraic laws for QGCL programs that can be used in program verification, transformations and compilation. The expressive power of QGCL is illustrated by several examples where various variants and generalizations of quantum walks are conveniently expressed using quantum alternation and quantum choice. We believe that quantum programming with quantum alternation and choice will play an important role in further exploiting the power of quantum computing.

Key Words: Quantum computation, Programming language, Semantics, alternation, Superposition of data, Superposition of prograsm

"I suggested the notation for a 'case expression'which selects between any number of alternatives according to the value of an integer expression. That was my second language

*Mingsheng Ying and Yuan Feng are with University of Technology, Sydney and Tsinghua University; Nengkun Yu is with University of Waterloo. This work was partly supported by the Australian Research Council (Grant No: DP110103473 and DP130102764). *Email: Mingsheng.Ying@uts.edu.au or yingmsh@tsinghua.edu.cn*

design proposal. I am still most proud of it, because it raises essentially no problems either for the implementor, the programmer, or the reader of a program."

– A. C. R. Hoare, The emperor's old clothes, *Communications of the ACM* 24(1981)75-83.

1 Introduction

Since Knill [24] introduced the Quantum Random Access Machine (QRAM) model for quantum computing and proposed a set of conventions for writing quantum pseudo-codes in 1996, several high-level quantum programming languages have been defined in the last 17 years; for example imperative languages QCL by Ömer [34] and qGCL by Sanders and Zuliani [35, 50], and functional languages QPL by Selinger [36] and QML by Altenkirch and Grattage [4]. Also, Tafliovich and Hehner [40] defined a quantum extension of Hehner's predicative programming language. Various semantics have been introduced for quantum programming languages; for example, D'Hondt and Panangaden [16] introduced the notion of quantum weakest precondition, and a quantum predicate transformer semantics was proposed in [45]. Several proof systems for verification of quantum programs have been developed; for example Baltag and Smets [8] presented a dynamic logic of quantum information flow, Brunet and Jorrand [12] introduced a way of using Birkhoff-von Neumann quantum logic to reason about quantum programs, and Hoare logic was generalized to prove both partial and total correctness of quantum programs by Chadha, Mateus and Sernadas [13], Kakutani [23] and the authors [18, 43]. The implementation of quantum programming languages has also attracted attention [39, 32, 47] as the rapid progress of quantum technology has made people widely believe that large-scalable and functional quantum computers will be built in not too far future. An excellent survey of research on quantum programming before 2006 can be found in [19], and for a more recent survey, see [44, 48]. It is particularly worth pointing out that three more practical quantum programming languages were announced in the last two years: two general-purpose languages Quipper by Green, LeFanu Lumsdaine, Ross, Selinger and Valiron [20], and Scaffold by Abhari, Faruque, Dousti, et. al. [1], and a domain-specific language QuaFL by Lapets, da Silva, Thome, Adler, Beal and Rötteler [27].

Now the development of the theory of quantum programming has reached such a stage that the quantum extensions of various basic program constructs (e.g. sequential composition) have been properly introduced in the languages mentioned above. Then an important problem for further studies would be to (re)examine more sophisticated program constructs and programming abstractions and models that have been successfully used in classical programming in the quantum setting: how to define the quantum counterparts of them? how can they be used in programming a quantum computer? is it possible to employ them to solve a problem more efficiently on a quantum computer than on a classical computer? A further problem is: what kind of new programming language features that have not been introduced or even irrelevant in classical programming are needed in order to exploit the full capability of a quantum computer?

Alternation, case statement or switch statement is a very convenient program construct to implement a case analysis in classical programming [17, 31]. Thus, most high-level imperative programming languages possess alternation constructs. In particular, the nondeterminism derived from alternation provides a basis for refinement-based program development; for example, Barman, Bodík, Chandra, Galenson, Kimelman, Rodarmor and Tung [7] recently introduced a methodology of programming with angelic nondeterminism. This paper identifies a novel quantum programming paradigm - superposition of programs - from the design idea of a class of popular quantum algorithms, namely quantum walk-based algorithms. We find that a quantum generalization of alternation is crucial to support quantum programming in this new paradigm. Surprisingly, the notion of alternation in classical programming languages splits into two different notions in the quantum setting: classical alternation (of quantum programs) and quantum alternation. Classical alternation of quantum programs has already been properly introduced in the previous works on quantum programming, but it is not the program construct that we require for the purpose of realizing superposition of programs. The major aim of this paper is to define the new notion of quantum alternation that can support the paradigm of superposition of programs.

1.1 Alternation and Choice in Classical Programming

Recall that an alternation is a collection of guarded commands written as

$$\begin{array}{l} \textbf{if } G_1 \rightarrow P_1 \\ \square\, G_2 \rightarrow P_2 \\ \quad \ldots\ldots \\ \square\, G_n \rightarrow P_n \\ \textbf{fi} \end{array} \tag{1}$$

or

$$\textbf{if } (\square i \cdot G_i \rightarrow P_i)\ \textbf{fi} \tag{2}$$

where for each $1 \leq i \leq n$, the subprogram P_i is guarded by the boolean expression G_i, and P_i will be executed only when G_i is true.

Alternation is also the most widely accepted mechanism for nondeterministic programming. Nondeterminism in alternation (1) or (2) is a consequence of the "overlapping" of the guards $G_1, G_2, ..., G_n$; that is, if more than one guards G_i are true at the same time, the alternation needs to select one from the corresponding commands P_i for execution. In particular, if $G_1 = G_2 = \cdots = G_n = \mathbf{true}$, then alternation (1) or (2) becomes a demonic choice:

$$\square_{i=1}^{n} P_i \tag{3}$$

where the alternatives P_i are chosen unpredictably.

To formalise randomised algorithms, research on probabilistic programming [25, 29] started in 1980's with the introduction of probabilistic choice:

$$\square_{i=1}^{n} C_i @ p_i \tag{4}$$

where $\{p_i\}$ is a probability distribution; that is, $p_i \geq 0$ for all i, and $\sum_{i=1}^{n} p_i = 1$. The probabilistic choice (4) randomly chooses the command C_i with probability p_i for every i, and thus it can be seen as a refinement of the demonic choice (3). A probabilistic choice is often used to represent a decision in forks according to a certain probability distribution in a randomised algorithm.

1.2 Classical Alternation in Quantum Programming

As stated before, the aim of this paper is to define a quantum generalization of alternation. Indeed, a kind of alternation already exists in Sanders and Zuliani's quantum programming language qGCL [35, 50] because qGCL is the probabilistic GCL [29] extended by adding the quantum procedures of unitary transformations and measurements, and thus alternation and probabilistic choice in pGCL are inherited in qGCL. Another kind of measurement-based alternation was introduced by Selinger in his quantum programming language QPL [36]. Let $\overline{q}$ be a family of quantum variables and M a measurement on $\overline{q}$ with possible outcomes $m_1, m_2, ..., m_n$. For each $1 \leq i \leq n$, let P_i be a (quantum) program. Then a generalized form of Selinger's alternation considered in [43] can be written as follows:

$$\begin{array}{lll} \textbf{measure}\ M[\overline{q}] = & m_1 \rightarrow P_1 & \\ \quad \square & m_2 \rightarrow P_2 & \\ & ..... & \\ \quad \square & m_n \rightarrow P_n & \\ \textbf{end} & & \end{array} \tag{5}$$

or

$$\textbf{measure}\ (\square i \cdot M[\overline{q}] = m_i \rightarrow P_i)\ \textbf{end} \tag{6}$$

Alternation (5) or (6) selects a command according to the outcome of measurement M: if the outcome is m_i, then the corresponding command P_i will be executed. The alternations defined in both qGCL and QPL can be appropriately termed as *classical alternation of quantum programs* because the selection of commands in it based on classical information - the outcomes of quantum measurements. However, our intention is to introduce the notion of *quantum alternation of (quantum) programs*. Do we actually need quantum alternation in quantum programming? A role for programming languages is to provide ways of organizing computations [37]. So, to answer this question, let's look at the basic design ideas of several popular quantum algorithms.

1.3 From Superposition of Data to Superposition of Programs

1.3.1 Superposition of Data

It has been realized well that one major source of the power of quantum computation [33] is superposition of data. To see this, let's consider a function

$$f(x_1, ..., x_n) : \{0,1\}^n \to \{0,1\}$$

with $n-$bit input and one-bit output, we want to compute $f(x)$ for multiple inputs $x = (x_1, ..., x_n) \in \{0,1\}^n$ simultaneously. Classical parallelism is meant to build *multiple* circuits all for computing the same function f and to execute them in parallel for different inputs. However, quantum parallelism allows us to computer $f(x)$ for all different inputs $x \in \{0,1\}^n$ simultaneously with a *single* quantum circuit implementing the oracle unitary operator:

$$U_f : |x, y\rangle \to |x, y \oplus f(x)\rangle \tag{7}$$

where $x \in \{0,1\}^n$, $y \in \{0,1\}$ and $\oplus$ is addition module 2. This quantum circuit has $n+1$ qubits: the n qubits $x = (x_1, ..., x_n)$ are called the data register, and the qubit y is called the target register. Initially, we prepare the data and target registers in computational basis state $|0\rangle^{\otimes n}|0\rangle$. The superposition

$$|\psi\rangle \stackrel{\triangle}{=} \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle = H^{\otimes n}|0\rangle^{\otimes n} \tag{8}$$

of the basis states of n qubits can be created by n Hadamard gates $H^{\otimes n}$ acting in parallel on the data register, where

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{9}$$

is the Hadamard gate. Then we apply unitary operator U_f:

$$|\psi\rangle|0\rangle \xrightarrow{U_f} \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|f(x)\rangle \stackrel{\triangle}{=} |\varphi\rangle.$$

The different terms of $|\varphi\rangle$ contain information about values $f(x)$ for all 2^n inputs $x \in \{0,1\}^n$. Thus, we have computed $f(x)$ for all $x \in \{0,1\}^n$ simultaneously by employing U_f only once. Such a technique of applying a single circuit U_f to the superposition (8) of data is a key step of a large class of quantum algorithms, including the Grover search algorithm [21], the Deustch-Josza algorithm [15] and the Bernstein-Vazirani algorithm [9].

1.3.2 Superposition of Programs

After understanding superposition of data, a question naturally arises: is there any other form of superposition that is useful in quantum computing? A superposition of evolutions (rather than that of states) of a quantum systems was considered by physicists Aharonov, Anandan, Popescu and Vaidman [3] as early as in 1990, and they proposed to introduce

an external system in order to implement the superposition. The idea of using such an external system was rediscovered by Aharonov, Ambainis, Bach, Kempe, Nayak, Vazirani, Vishwanath and Watrous in defining quantum walks [5, 2]. Let's consider a simple example of the quantum walk on a graph:

Example 1.1 *A quantum walk is the quantum counterpart of a random walk. Let $G = (V, E)$ be an $n-$regular directed graph; that is, a graph where each vertex has n neighbors. Then we can label each edge with a number between 1 and n such that for each $1 \leq i \leq n$, the directed edges labeled i form a permutation. A random walk on G is defined as follows: the vertices v's of G are used to represent the states of the walk, and for each state v the walk goes from v to its every neighbor with a certain probability. To define a quantum walk on G, let $\mathcal{H}_V$ be the Hilbert space spanned by states $\{|v\rangle\}_{v\in V}$ corresponding to the vertices of the graph. Then for each $1 \leq i \leq n$, we can define a shift operator S_i on $\mathcal{H}_V$:*

$$S_i|v\rangle = |\text{the } i\text{th neighbour of } v\rangle$$

for any $v \in V$. We introduce an auxiliary quantum system with the state Hilbert space $\mathcal{H}_C$ spanned by $\{|i\rangle\}_{i=1}^n$. This auxiliary system is usually called a "quantum coin", and the space $\mathcal{H}_C$ is referred to as the "coin space". Now we are able to combine these unitary operators S_i $(1 \leq i \leq n)$ along the "coin" to form a whole shift operator S on $\mathcal{H}_C \otimes \mathcal{H}_V$:

$$S|i, v\rangle = |i\rangle S_i|v\rangle \tag{10}$$

for any $1 \leq i \leq n$ and $v \in V$. If we further choose a unitary operator C on $\mathcal{H}_C$, e.g. the Hadamard gate defined by equation (9), called the "coin-tossing operator", then a single step of a coined quantum walk on graph G can be modelled by the unitary operator:

$$W \stackrel{\triangle}{=} S(C \otimes I_{\mathcal{H}_V}) \tag{11}$$

where $I_{\mathcal{H}_V}$ is the identity operator on $\mathcal{H}_V$. The quantum walk is then an iteration of the single-step walk operator W.

Let's carefully observe the behavior of the quantum walk step W. The shift operators $S_1, S_2, ..., S_n$ can be seen as a collection of programs independent to each other. Then the whole shift operator S can be seen as a kind of alternation of $S_1, S_2, ..., S_n$ because S selects one of them for execution. But the defining equation (10) of S clearly indicates that this alternation is different from the alternation (5) or (6): the selection in equation (10) is made according to the basis state $|i\rangle$ of the "coin space", which is quantum information rather than classical information. Thus, we can appropriately call S an *quantum alternation*. Furthermore, the "coin-tossing operator" C can be seen as another program. From equation (11) we see that the quantum walk step W first runs "coin-tossing" program C to create a superposition of the execution paths of programs $S_1, S_2, ..., S_n$, and then the quantum alternation S follows. During the execution of alternation S, each S_i is running along its own path within the whole superposition of execution paths of $S_1, S_2, ..., S_n$. Then the quantum walk step W is indeed a *quantum choice* of shift programs $S_1, S_2, ..., S_n$ through the "coin-tossing" program C. Therefore, the superposition in W is a higher-level superposition - the *superposition of programs $S_1, S_2, ..., S_n$*.

1.4 Design Decision of the Paper

Quantum walks have been shown to be a very powerful tool for the development of a large class of quantum algorithms, in particular for simulation of quantum systems (see [41] for a comprehensive review). Moreover, they were proved to be a universal model of computation [14, 28]. This motivates us to develop the idea of superposition of programs embedded in quantum walk-based algorithms as a *quantum programming paradigm*. As suggested by Example 1.1, the following two steps are needed toward a general form of superposition of programs.

1.4.1 Quantum Alternation

The key step is to define quantum alternation. This is exactly the major aim of this paper. The defining equation (10) of the shift operator S of a quantum walk already provides us with a basic idea for defining quantum alternation. Let $P_1, P_2, ..., P_n$ be a collection of (quantum) programs whose state spaces are the same Hilbert space $\mathcal{H}$. We introduce a new family of quantum variables $\overline{q}$ that do not appear in $P_1, P_2, ..., P_n$. These variables are used to denote an external "coin" system. Assume that the state space of system $\overline{q}$ is an $n-$dimensional Hilbert space $\mathcal{H}_C$ and $\{|i\rangle\}_{i=1}^n$ is an orthonormal basis of it. Then it seems that a quantum alternation P of programs $P_1, P_2, ..., P_n$ can be defined by combining them along the basis $\{|i\rangle\}$, simply mimicking the shift operator S. More precisely, the semantic operator $[\![P]\!]$ of alternation P should be defined on the tensor product $\mathcal{H}_C \otimes \mathcal{H}$, and

$$[\![P]\!](|i\rangle|\varphi\rangle) = |i\rangle([\![P_i]\!]|\varphi\rangle) \tag{12}$$

for every $1 \leq i \leq n$ and $|\varphi\rangle \in \mathcal{H}$, where $[\![P_i]\!]$ is the semantic operator of P_i. We write the alternation P as

$$\begin{array}{ll} \textbf{qif } [\overline{q}]: & |1\rangle \rightarrow P_1 \\ \square & |2\rangle \rightarrow P_2 \\ & ..... \\ \square & |n\rangle \rightarrow P_n \\ \textbf{fiq} & \end{array} \tag{13}$$

or

$$\textbf{qif } [\overline{q}](\square i \cdot |i\rangle \rightarrow P_i) \textbf{ fiq} \tag{14}$$

(Whenever the family $\overline{q}$ of quantum variables can be recognised from the context, it can be dropped from the above notation.) The control flow of program in the above alternation is determined by quantum variables $\overline{q}$. For each $1 \leq i \leq n$, P_i is guarded by the basis state $|i\rangle$. A superposition of these basis states yields a quantum control flow - superposition of control flows:

$$[\![P]\!]\left(\sum_{i=1}^n \alpha_i |i\rangle|\varphi_i\rangle\right) = \sum_{i=1}^n \alpha_i |i\rangle([\![P_i]\!]|\varphi_i\rangle)$$

for all $|\varphi_i\rangle \in \mathcal{H}$ and complex numbers α_i $(1 \leq i \leq n)$. This is very different from the classical alternation (5) or (6) of quantum programs where the guards in an alternation cannot be superposed.

Quantum alternation is a convenient notion for describing quantum algorithms; for example, the shift operator of a quantum walk can be written as a quantum alternation:

$$S = \mathbf{qif}\ (\square i \cdot |i\rangle \rightarrow S_i)\ \mathbf{fiq}$$

It is interesting to note that even superposition of data can be seen as a special case of superposition of programs: for each $x \in \{0,1\}^n$, let

$$V_x : |y\rangle \rightarrow |y \oplus f(x)\rangle$$

for $y = 0$ or 1. Clearly, V_x is a unitary operator on the $2-$dimensional Hilbert space. Then the oracle operator U_f defined in (7) can be written as a quantum alternation:

$$U_f = \mathbf{qif}\ (\square x \in \{0,1\}^n \cdot |x\rangle \rightarrow V_x)\ \mathbf{fiq}$$

1.4.2 Quantum Choice

Following the idea of defining equation (11) of quantum walk operator W, a general form of quantum choice can be easily defined in terms of quantum alternation. Let $P_1, P_2, ..., P_n$ be a collection of (quantum) programs, $\overline{q}$ a new family of quantum variables that do not appear in $P_1, P_2, ..., P_n$, and P a quantum program acting on $\overline{q}$. Assume that $\{|i\rangle\}$ is an orthonormal basis of the state Hilbert space of the "coin" system denoted by $\overline{q}$. Then the quantum choice of $P_1, P_2, ..., P_n$ along $\{|i\rangle\}$ with "coin-tossing" program P is defined as follows:

$$[P]\left(\bigoplus_{i=1}^{n} P_i\right) \stackrel{\triangle}{=} P; \mathbf{qif}\ [\overline{q}](\square i \cdot |i\rangle \rightarrow P_i)\ \mathbf{fiq} \tag{15}$$

Intuitively, quantum choice (15) first runs program P to produce a superposition of the respective execution paths of programs P_i $(1 \leq i \leq n)$, and then enters the quantum alternation of $P_1, P_2, ..., P_n$ where each P_i is running along its own path within the superposition of paths generated by P.

It is interesting to compare quantum choice with probabilistic choice (4). A probabilistic choice is a resolution of nondeterminism where we can simply say that the choice is made according to a certain probability distribution. However, when defining a quantum choice, a "device" that can actually perform the choice, namely a "quantum coin", has to be explicitly introduced.

1.5 Technical Contributions of the Paper

At the first glance, it seems that the defining equation (10) of shift operator S in Example 1.1 can be smoothly generalized to equation (12) to define the denotational semantics of a general quantum alternation P of programs $P_1, P_2, ..., P_n$. But there is actually a major difficulty in equation (12). For the case where no quantum measurement occur in any

P_i $(1 \leq i \leq n)$, the operational semantics of each P_i is simply a sequence of unitary operators, and equation (12) is not problematic at all. Whenever some P_i contains quantum measurements, however, its semantic structure becomes a tree of linear operators with branching happening at the points where the measurements are performed. Then equation (12) becomes meaningless within the framework of quantum mechanics, and defining the semantics of quantum alternation P requires to properly combine a collection of trees of quantum operations such that the relevant quantum mechanical principles are still obeyed. This problem will be circumvented in Sections 3 and 4 by introducing a semi-classical semantics in terms of operator-valued functions as a middle step toward a purely quantum denotational semantics of programs. Based on this, we systematically develop a theory of quantum programming with quantum alternation and choice. In particular, a set of programming laws for quantum alternation and choice are established.

1.6 Organisation of the Paper

We assume that the readers are familiar with the basics of quantum theory including density operator description of mixed quantum states and the super-operator formalism of dynamics of (open) quantum systems; a reader who has no basic knowledge about quantum theory can consult a standard quantum computation textbook [33] or the preliminary sections of several influential quantum programming papers [36, 35, 16] and survey [19] as well as the authors' recent papers [43, 46].

This paper is organized as follows. A new quantum programming language QGCL is defined in Section 2 to support quantum programming with quantum alternation. Section 3 prepares several key ingredients needed in defining the denotational semantics of QGCL, including guarded composition of various quantum operations. The denotational semantics and weakest precondition semantics of QGCL are presented in Section 4. In Section 5, quantum choice is defined in terms of quantum guarded command, and probabilistic choice is implemented by quantum choice through introducing local variables. It should be pointed out that a quantum implementation of probabilistic choice was already given by Zuliani [50] in a different way by using a quantum measurement. A family of algebraic laws for QGCL programs are presented in Section 6. Several examples are given in Section 7 to illustrate the expressive power of the language QGCL. For readability, some more technical materials are postponed to the appendices. A discussion about the choice of coefficients in the definition of guarded composition of quantum operations is presented in Appendix A. Quantum alternation defined in Sections 2 is guarded by an orthonormal basis of the "coin" space. In Appendix B, we show that the notion of quantum alternation can be generalized to the case where guards are orthogonal subspaces of the "coin" space. All the proofs of lemmas, propositions and theorems are deferred to Appendix C.

2 QGCL: A Language with Quantum Alternation

We first define the syntax of quantum programming language QGCL. It is essentially an extension of Sanders and Zuliani's qGCL [35] obtained by adding quantum alternation.

But the presentation of QGCL is quite different from qGCL due to the complications in the semantics of quantum alternation. QGCL also borrows some ideas from Selinger's language QPL [36]. We assume a countable set $qVar$ of quantum variables ranged over by $q, q_1, q_2, ...$ For simplicity of the presentation, we only consider a purely quantum programming language, but we include a countably infinite set Var of classical variables ranged over by $x, y, ...$ so that we can use them to record the outcomes of quantum measurements. However, classical computation described by, for example, the assignment statement $x := e$ in a classical programming language, is excluded. It is required that the sets of classical and quantum variables are disjoint. For each classical variable $x \in Var$, its type is assumed to be a non-empty set D_x; that is, x takes values from D_x. In applications, if x is used to store the outcome of quantum measurement M, then $spec(M)$ (the set of all possible outcomes of M) should be a subset of D_x. For each quantum variable $q \in qVar$, its type is a Hilbert space $type(q) = \mathcal{H}_q$, which is the state space of the quantum system denoted by q. For a sequence $\overline{q} = q_1, q_2, \cdots$ of distinct quantum variables, we write:

$$type(\overline{q}) = \mathcal{H}_{\overline{q}} = \bigotimes_{i \geq 1} \mathcal{H}_{q_i}.$$

So, $type(\overline{q})$ is the state Hilbert space of the composed system denoted by $\overline{q}$. Similarly, for any set $V \subseteq qVar$, we write:

$$type(V) = \mathcal{H}_V = \bigotimes_{q \in V} \mathcal{H}_q \tag{16}$$

for the state Hilbert space of the composed system denoted by V. In particular, we write $\mathcal{H}_{all}$ for $type(qVar)$. To simplify the notation, we often identify a sequence of variables with the set of these variables provided they are distinct.

Definition 2.1 *For each QGCL program P, we write $var(P)$ for the set of its classical variables $qvar(P)$ for its quantum variables and $cvar(P)$ for its "coin" variables. Then QGCL programs are inductively defined as follows:*

1. $\mathbf{abort}$ *and* $\mathbf{skip}$ *are programs, and*

$$var(\mathbf{abort}) = var(\mathbf{skip}) = \emptyset,$$

$$qvar(\mathbf{abort}) = qvar(\mathbf{skip}) = \emptyset,$$

$$cvar(\mathbf{abort}) = cvar(\mathbf{skip}) = \emptyset.$$

2. *If $\overline{q}$ is a sequence of distinct quantum variables, and U is a unitary operator on $type(\overline{q})$, then $U[\overline{q}]$ is a program, and*

$$var(U[\overline{q}]) = \emptyset, \quad qvar(U[\overline{q}]) = \overline{q}, \quad cvar(U[\overline{q}]) = \emptyset.$$

3. *If $\overline{q}$ is a sequence of distinct quantum variables, x is a classical variable, $M = \{M_m\}$ is a quantum measurement on $type(\overline{q})$ such that $spec(M) \subseteq D_x$, where $spec(M) =$*

$\{m\}$ is the spectrum of M; that is, the set of all possible outcomes of M, and $\{P_m\}$ is a family of programs indexed by the outcomes m of measurement M such that $x \notin \bigcup_m var(P_m)$, then the classical alternation of P_m's guarded by measurement outcomes m's:

$$P \stackrel{\triangle}{=} \textbf{measure}\ (\square m \cdot M[\overline{q} : x] = m \to P_m)\ \textbf{end} \tag{17}$$

is a program, and

$$var(P) = \{x\} \cup \bigcup_m var(P_m),$$

$$qvar(P) = \overline{q} \cup \bigcup_m qvar(P_m),$$

$$cvar(P) = \bigcup_m cvar(P_m).$$

4. *If $\overline{q}$ is a sequence of distinct quantum variables, $\{|i\rangle\}$ is an orthonormal basis of $type(\overline{q})$, and $\{P_i\}$ is a family of programs indexed by the basis states $|i\rangle$'s such that*

$$\overline{q} \cap \left(\bigcup_i qVar(P_i) \right) = \emptyset,$$

then the quantum alternation of P_i's guarded by basis states $|i\rangle$'s:

$$P \stackrel{\triangle}{=} \textbf{qif}\ [\overline{q}]\ (\square i \cdot\ |i\rangle \to P_i)\ \textbf{fiq} \tag{18}$$

is a program, and

$$var(P) = \bigcup_i var(P_i),$$

$$qvar(P) = \overline{q} \cup \bigcup_i qvar(P_i),$$

$$cvar(P) = \overline{q} \cup \bigcup_i cvar(P_i).$$

5. *If P_1 and P_2 are programs such that $var(P_1) \cap var(P_2) = \emptyset$, then $P_1; P_2$ is a program, and*

$$var(P_1; P_2) = var(P_1) \cup var(P_2),$$

$$qvar(P_1; P_2) = qvar(P_1) \cup qvar(P_2),$$

$$cvar(P_1; P_2) = cvar(P_1) \cup cvar(P_2).$$

The meanings of **abort** and **skip** are the same as in a classical programming language. Two kinds of statements are introduced in the above definition to describe basic quantum operations, namely unitary transformation and measurement. In the unitary transformation $U[\overline{q}]$, only quantum variables $\overline{q}$ but no classical variables appear, and the transformation is

applied to $\overline{q}$. In statement (17), a measurement M is first performed on quantum variables $\overline{q}$ with the outcome stored in classical variable x, and then whenever outcome m is reported, the corresponding subprogram P_m is executed. It is required in statement (17) that $x \notin \bigcup_m var(P_m)$. This means that the classical variables already used to record the outcomes of the measurements in P_m's are not allowed to store the outcome of a new measurement. This technical requirement is cumbersome, but it can significantly simplify the presentation of the semantics of QGCL. On the other hand, it is not required that the measured quantum variables $\overline{q}$ do not occur in P_m. So, measurement M can be performed not only on an external system but also on some quantum variables within P_m. The statement (17) and classical alternation (5) or (6) (of quantum programs) are essentially the same, and the only difference between them is that a classical variable x is added in (17) to record the measurement outcome. The intuitive meaning of quantum alternation (18) was already carefully explained in Section 1. Only one thing is worthy to mention: it is required that the variables in $\overline{q}$ do not appear in any P_i's. This indicates that the "coin system" $\overline{q}$ is external to programs P_i's. Whenever the sequence $\overline{q}$ of quantum variables can be recognized from the context, then it can be dropped from statement (18). The sequential composition $P_1; P_2$ is similar to that in a classical language, and the requirement $var(P_1) \cap var(P_2) = \emptyset$ means that the outcomes of measurements performed at different points are stored in different classical variables. Such a requirement is mainly for technical convenience, and it will considerably simplify the presentation. Obviously, all "coin" are quantum variables: $cvar(P) \subseteq qvar(P)$ for all programs P. The set $cvar(P)$ of "coin" variables of program P will be needed in defining a kind of equivalence between quantum programs. The syntax of QGCL can be summarised as follows:

$$\begin{aligned} P := {} & \mathbf{abort} \mid \mathbf{skip} \mid P_1; P_2 \\ & \mid U[\overline{q}] && (\text{unitary transformation}) \\ & \mid \mathbf{measure}\ (\square m \cdot M[\overline{q} : x] = m \to P_m)\ \mathbf{end} && (\text{classical alternation}) \\ & \mid \mathbf{qif}\ [\overline{q}](\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq} && (\text{quantum alternation}) \end{aligned} \tag{19}$$

3 Guarded Compositions of Quantum Operations

A major difficulty in defining the semantics of QGCL comes from the treatment of quantum alternation. This section provides the key mathematical tool for defining the semantics of quantum alternation, namely guarded composition of quantum operations.

3.1 Guarded Composition of Unitary Operators

To ease the understanding of a general definition of guarded composition, we start with a special case of the guarded composition of unitary operators, which is a straightforward generalisation of the quantum walk shift operator S in Example 1.1.

Definition 3.1 *For each $1 \leq i \leq n$, let U_i be an unitary operator in Hilbert space $\mathcal{H}$. Let $\mathcal{H}_C$ be an auxiliary Hilbert space, called the "coin space", with $\{|i\rangle\}$ as an orthonormal*

basis. Then we define a linear operator:

$$U \stackrel{\triangle}{=} \square_{i=1}^{n} |i\rangle \rightarrow U_i$$

in $\mathcal{H}_C \otimes \mathcal{H}$ *by*

$$U(|i\rangle|\psi\rangle) = |i\rangle(U_i|\psi\rangle) \tag{20}$$

for any $|\psi\rangle \in \mathcal{H}$ *and for any* $1 \leq i \leq n$*. Then by linearity we have:*

$$U\left(\sum_{i,j} \alpha_{ij}|i\rangle|\psi_j\rangle\right) = \sum_{i,j} \alpha_{ij}|i\rangle\left(U_i|\psi_j\rangle\right) \tag{21}$$

for any $|\psi_j\rangle \in \mathcal{H}$ *and complex numbers* α_{ij}*. The operator* U *is called the guarded composition of* U_i $(1 \leq i \leq n)$ *along the basis* $\{|i\rangle\}$*.*

A routine calculation yields:

Lemma 3.1 *1. The guarded composition* $\square_i |i\rangle \rightarrow U_i$ *is an unitary operator in* $\mathcal{H}_C \otimes \mathcal{H}$*.*

2. For any two orthonormal basis $\{|i\rangle\}$ *and* $\{|\varphi_i\rangle\}$ *of the "coin space"* $\mathcal{H}_C$*, there exists an unitary operator* U_C *such that* $|\varphi_i\rangle = U_C|i\rangle$ *for all* i*, and the two compositions along different bases* $\{|i\rangle\}$ *and* $\{|\varphi_i\rangle\}$ *are related to each other by*

$$\square_i|\varphi_i\rangle \rightarrow U_i = (U_C \otimes I_{\mathcal{H}})(\square_i|i\rangle \rightarrow U_i)(U_C^{\dagger} \otimes I_{\mathcal{H}})$$

where $I_{\mathcal{H}}$ *is the identity operator in* $\mathcal{H}$*.*

Clause (1) of the above lemma indicates that the guarded composition of unitary operators is well defined, and clause (2) shows that the choice of orthonormal basis of the "coin space" is not essential for the definition of guarded composition.

The guarded composition of unitary operators is nothing new; it is just a quantum multiplexor introduced in [38] as a useful tool in the synthesis of quantum logic circuits.

Example 3.1 *A quantum multiplexor (QMUX for short) is a quantum generalisation of multiplexor, a well-known notion in digit logic. A QMUX* U *with* k *select qubits and* $d-$*qubit-wide data bus can be represented by a block-diagonal matrix:*

$$U = diag(U_0, U_1, ..., U_{2^k-1}) = \begin{pmatrix} U_0 & & & \\ & U_1 & & \\ & & ... & \\ & & & U_{2^k-1} \end{pmatrix}.$$

Multiplexing $U_0, U_1, ..., U_{2^k-1}$ *with* k *select qubits is exactly the guarded composition*

$$\square_{i=0}^{2^k-1}|i\rangle \rightarrow U_i$$

along the computational basis $\{|i\rangle\}$ *of* k *qubits.*

3.2 Operator-Valued Functions

A general form of guarded composition of quantum operations cannot be defined by a straightforward generalization of Definition 3.1. Instead, we need an auxiliary notion of operator-valued function. For any Hilbert space $\mathcal{H}$, we write $\mathcal{L}(\mathcal{H})$ for the space of (bounded linear) operators in $\mathcal{H}$.

Definition 3.2 *Let Δ be a nonempty set. Then a function $F : \Delta \to \mathcal{L}(\mathcal{H})$ is called an operator-valued function in $\mathcal{H}$ over Σ if*

$$\sum_{\delta \in \Delta} F(\delta)^\dagger \cdot F(\delta) \sqsubseteq I_{\mathcal{H}}, \tag{22}$$

where $I_{\mathcal{H}}$ is the identity operator in $\mathcal{H}$, and $\sqsubseteq$ stands for the Löwner order; that is, $A \sqsubseteq B$ if and only if $B - A$ is a positive operator. In particular, F is said to be full when equation (22) becomes equality.

The simplest examples of operator-valued function are unitary operators and quantum measurements.

Example 3.2 *1. A unitary operator U in Hilbert space $\mathcal{H}$ can be seen as a full operator-valued function over a singleton $\Delta = \{\epsilon\}$. This function maps ϵ to U.*

2. A quantum measurement $M = \{M_m\}$ in Hilbert space $\mathcal{H}$ can be seen as a full operator-valued function over its spectrum $spec(M) = \{m\}$ (the set of possible measurement outcomes). This function maps each measurement outcome m to the corresponding measurement operator M_m.

More generally, a super-operator (or quantum operation) defines a family of operator-valued functions. Let $\mathcal{E}$ be a super-operator in Hilbert space $\mathcal{H}$. Then $\mathcal{E}$ has the Kraus operator-sum representation:

$$\mathcal{E} = \sum_i E_i \circ E_i^\dagger,$$

meaning:

$$\mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger$$

for all density operators ρ in $\mathcal{H}$ (see [33], Chapter 8). For such a representation, we set $\Delta = \{i\}$ for the set of indexes, and define an operator-valued function over Δ by

$$F(i) = E_i$$

for every i. Since operator-sum representation of $\mathcal{E}$ is not unique, $\mathcal{E}$ defines not only a single operator-valued function. We write $\mathbb{F}(\mathcal{E})$ for the family of operator-valued functions defined by all Kraus operator-sum representations of $\mathcal{E}$. Conversely, an operator-valued function determines uniquely a super-operator.

Definition 3.3 *Let F be an operator-valued function in Hilbert space $\mathcal{H}$ over set Δ. Then F defines a super-operator $\mathcal{E}(F)$ in $\mathcal{H}$ as follows:*

$$\mathcal{E}(F) = \sum_{\delta \in \Delta} F(\delta) \circ F(\delta)^\dagger;$$

that is,

$$\mathcal{E}(F)(\rho) = \sum_{\delta \in \Delta} F(\delta) \rho F(\delta)^\dagger$$

for every density operator ρ.

For a family $\mathbb{F}$ of operator-valued functions, we write:

$$\mathcal{E}(\mathbb{F}) = \{\mathcal{E}(F) : F \in \mathbb{F}\}.$$

It is obvious that $\mathcal{E}(\mathbb{F}(\mathcal{E})) = \{\mathcal{E}\}$ for each super-operator $\mathcal{E}$. On the other hand, for any operator-valued function F over $\Delta = \{\delta_1, ..., \delta_k\}$, it follows from Theorem 8.2 in [33] that $\mathbb{F}(\mathcal{E}(F))$ consists of all operator-valued functions G over some $\Gamma = \{\gamma_1, ..., \gamma_l\}$ such that

$$G(\gamma_i) = \sum_{j=1}^{n} u_{ij} \cdot F(\delta_j)$$

for each $1 \leq i \leq n$, where $n = \max(k, l)$, $U = (u_{ij})$ is an $n \times n$ unitary matrix, $F(\delta_i) = G(\gamma_j) = 0_{\mathcal{H}}$ for all $k + 1 < i \leq n$ and $l + 1 < j \leq n$, and $0_{\mathcal{H}}$ is the zero operator in $\mathcal{H}$.

3.3 Guarded Composition of Operator-Valued Functions

We need to introduce a notation before defining guarded composition of operator-valued functions. Let Δ_i be a nonempty set for every $1 \leq i \leq n$. Then the superposition of Δ_i $(1 \leq i \leq n)$ is defined as follows:

$$\bigoplus_{i=1}^{n} \Delta_i = \{\oplus_{i=1}^{n} \delta_i : \delta_i \in \Delta_i \text{ for every } 1 \leq i \leq n\}. \tag{23}$$

Here, $\oplus_{i=1}^{n} \delta_i$ is simply a notation indicating a combination of δ_i $(1 \leq i \leq n)$, and we do not need to care its meaning.

Definition 3.4 *For each $1 \leq i \leq n$, let F_i be an operator-valued function in Hilbert space $\mathcal{H}$ over set Δ_i. Let $\mathcal{H}_C$ be a "coin" Hilbert space with $\{|i\rangle\}$ as an orthonormal basis. Then the guarded composition*

$$F \stackrel{\triangle}{=} \square_{i=1}^{n} |i\rangle \rightarrow F_i$$

of F_i $(1 \leq i \leq n)$ along the basis $\{|i\rangle\}$ is defined to be the operator-valued function

$$F : \bigoplus_{i=1}^{n} \Delta_i \rightarrow \mathcal{L}(\mathcal{H}_C \otimes \mathcal{H})$$

in $\mathcal{H}_C \otimes \mathcal{H}$ over $\bigoplus_{i=1}^n \Delta_i$. For any $\delta_i \in \Delta_i$ $(1 \leq i \leq n)$, $F(\oplus_{i=1}^n \delta_i)$ is an operator in $\mathcal{H}_C \otimes \mathcal{H}$ defined as follows: for each $|\Psi\rangle \in \mathcal{H}_C \otimes \mathcal{H}$, there is a unique tuple $(|\psi_1\rangle, ..., |\psi_n\rangle)$ such that $|\psi_1\rangle, ..., |\psi_n\rangle \in \mathcal{H}$ and $|\Psi\rangle$ can be written as

$$|\Psi\rangle = \sum_{i=1}^{n} |i\rangle|\psi_i\rangle,$$

and then we define

$$F(\oplus_{i=1}^n \delta_i)|\Psi\rangle = \sum_{i=1}^{n} \left(\prod_{k \neq i} \lambda_{k\delta_k} \right) |i\rangle(F_i(\delta_i)|\psi_i\rangle) \tag{24}$$

where

$$\lambda_{k\delta_k} = \sqrt{\frac{trF_k(\delta_k)^\dagger F_k(\delta_k)}{\sum_{\tau_k \in \Delta_k} trF_k(\tau_k)^\dagger F_k(\tau_k)}}. \tag{25}$$

In particular, if F_k is full and $d = \dim \mathcal{H} < \infty$, then

$$\lambda_{k\delta_k} = \sqrt{\frac{trF_k(\delta_k)^\dagger F_k(\delta_k)}{d}}$$

for any $\delta_k \in \Delta_k$ $(1 \leq k \leq n)$.

Intuitively, the square $\lambda_{k\delta_k}^2$ of the coefficients defined in equation (25) can be understood as a kind of conditional probability. A further discussion on the choice of coefficients in equation (25) is given in Appendix A. The following lemma shows that the guarded composition of operator-valued functions is well-defined, and the choice of orthonormal basis of the "coin space" is not essential in its definition.

Lemma 3.2 *1. The guarded composition $F \stackrel{\triangle}{=} \square_{i=1}^n |i\rangle \rightarrow F_i$ is an operator-valued function in $\mathcal{H}_C \otimes \mathcal{H}$ over $\bigoplus_{i=1}^n \Delta_i$. In particular, if all F_i $(1 \leq i \leq n)$ are full, then so is F.*

2. For any two orthonormal bases $\{|i\rangle\}$ and $\{|\varphi_i\rangle\}$ of the "coin space" $\mathcal{H}_C$, there exists an unitary operator U_C such that $|\varphi_i\rangle = U_C|i\rangle$ for all i, and the two compositions along different bases $\{|i\rangle\}$ and $\{|\varphi_i\rangle\}$ are related to each other by

$$\square_{i=1}^n |\varphi_i\rangle \rightarrow F_i = (U_C \otimes I_{\mathcal{H}}) \cdot (\square_{i=1}^n |i\rangle \rightarrow F_i) \cdot (U_C^\dagger \otimes I_{\mathcal{H}});$$

that is,

$$(\square_{i=1}^n |\varphi_i\rangle \rightarrow F_i)(\oplus_{i=1}^n \delta_i) = (U_C \otimes I_{\mathcal{H}})(\square_{i=1}^n |i\rangle \rightarrow F_i)(\oplus_{i=1}^n \delta_i)(U_C^\dagger \otimes I_{\mathcal{H}})$$

for any $\delta_1 \in \Delta_1, ..., \delta_n \in \Delta_n$.

It is easy to see that whenever Δ_i is a singleton for all $1 \leq i \leq n$, then all $\lambda_{k\delta_k} = 1$ and equation (24) degenerates to (21). So, the above definition is a generalisation of guarded composition of unitary operators introduced in Definition 3.1. On the other hand, it can also be used to compose quantum measurements as shown in the following simple example.

Example 3.3 *We consider a guarded composition of two simplest quantum measurements. Let $M^{(0)}$ be the measurement on a qubit (the principal qubit) q in the computational basis $|0\rangle, |1\rangle$, i.e. $M^{(0)} = \{M_0^{(0)}, M_1^{(0)}\}$, where $M_0^{(0)} = |0\rangle\langle 0|$, $M_1^{(0)} = |1\rangle\langle 1|$, and let $M^{(1)}$ be the measurement of the same qubit but in a different basis:*

$$|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle),$$

i.e. $M^{(1)} = \{M_+^{(1)}, M_-^{(1)}\}$, where $M_+^{(1)} = |+\rangle\langle +|, M_-^{(1)} = |-\rangle\langle -|$. Then the guarded composition of $M^{(0)}$ and $M^{(1)}$ along the computational basis of another qubit (the "coin qubit") q_C is the measurement

$$M = (|0\rangle \to M^{(0)}) \square (|1\rangle \to M^{(1)}) = \{M_{0+}, M_{0-}, M_{1+}, M_{1-}\}$$

on two qubits q and q_C, where ij is an abbreviation of $i \oplus j$, and

$$M_{ij}(|0\rangle_{q_C}|\psi_0\rangle_q + |1\rangle_{q_C}|\psi_1\rangle_q) = \frac{1}{\sqrt{2}}(|0\rangle_{q_C} M_i^{(0)}|\psi_0\rangle_q + |1\rangle_{q_C} M_j^{(1)}|\psi_1\rangle_q)$$

for any states $|\psi_0\rangle, |\psi_1\rangle$ of the principal qubit q and $i \in \{0,1\}, j \in \{+,-\}$. Furthermore, for each state $|\Psi\rangle$ of two qubits q, q_C and for any $i \in \{0,1\}, j \in \{+,-\}$, a routine calculation yields that the probability that the outcome is ij when performing the guarded composition M of $M^{(0)}$ and $M^{(1)}$ on the two qubit system $q_C q$ in state $|\Psi\rangle$ is

$$p(i,j||\Psi\rangle, M) = \frac{1}{2}\left[p(i|_{q_C}\langle 0|\Psi\rangle, M^{(0)}) + p(j|_{q_C}\langle 1|\Psi\rangle, M^{(1)})\right],$$

where:

1. *if $|\Psi\rangle = |0\rangle_{q_C}|\psi_0\rangle_q + |1\rangle_{q_C}|\psi_1\rangle_q$, then ${}_{q_C}\langle k|\Psi\rangle = |\psi_k\rangle_q$ is the "conditional" state of the principal qubit q given that the two qubit system $q_C q$ is in state $|\Psi\rangle$ and the "coin" qubit q_C is in the basis state $|k\rangle$ for $k = 0, 1$;*

2. *$p(i|_{q_C}\langle 0|\Psi\rangle, M^{(0)})$ is the probability that the outcome is i when performing measurement $M^{(0)}$ on qubit q in state ${}_{q_C}\langle 0|\Psi\rangle$;*

3. *$p(j|_{q_C}\langle 1|\Psi\rangle, M^{(1)})$ is the probability that the outcome is j when performing measurement $M^{(1)}$ on qubit q in state ${}_{q_C}\langle 1|\Psi\rangle$.*

3.4 Guarded Composition of Super-Operators

Now the guarded composition of a family of super-operators can be defined through the guarded composition of the operator-valued functions generated from them.

Definition 3.5 *For each $1 \leq i \leq n$, let $\mathcal{E}_i$ be a super-operator in Hilbert space $\mathcal{H}$. Let $\mathcal{H}_C$ be a "coin" Hilbert space with $\{|i\rangle\}$ as an orthonormal basis. Then the guarded composition of $\mathcal{E}_i$ $(1 \leq i \leq n)$ along the basis $\{|i\rangle\}$ is defined to be the family of super-operators in $\mathcal{H}_C \otimes \mathcal{H}$:*

$$\square_{i=1}^n |i\rangle \rightarrow \mathcal{E}_i = \{\mathcal{E}(\square_{i=1}^n |i\rangle \rightarrow F_i) : F_i \in \mathbb{F}(\mathcal{E}_i) \text{ for every } 1 \leq i \leq n\},$$

where $\mathbb{F}(\mathcal{F})$ stands for the family of operator-valued functions defined by all Kraus operator-sum representations of an super-operator $\mathcal{F}$, and $\mathcal{E}(F)$ is the super-operator defined by an operator-valued function F (see Definition 3.3).

It is easy to see that if $n = 1$ then the above guarded composition of super-operators consists of only $\mathcal{E}_1$. For $n > 1$, however, it is usually not a singleton, as shown by the following:

Example 3.4 *For any unitary operator U in a Hilbert space $\mathcal{H}$, we write $\mathcal{E}_U = U \circ U^\dagger$ for the super-operator defined by U; that is,*

$$\mathcal{E}_U(\rho) = U\rho U^\dagger$$

for all density operators ρ in $\mathcal{H}$. Now suppose that U_0 and U_1 are two unitary operators in $\mathcal{H}$. Let U be the composition of U_0 and U_1 guarded by the computational basis $|0\rangle, |1\rangle$ of a qubit:

$$U = (|0\rangle \rightarrow U_0)\square(|1\rangle \rightarrow U_1).$$

Then $\mathcal{E}_U$ is an element of the guarded composition

$$\mathcal{E} = (|0\rangle \rightarrow \mathcal{E}_{U_0})\square(|1\rangle \rightarrow \mathcal{E}_{U_1})$$

of super-operators $\mathcal{E}_{U_0}$ and $\mathcal{E}_{U_1}$. But $\mathcal{E}$ contains more than one element. Indeed, it holds that

$$\mathcal{E} = \{\mathcal{E}_{U_\theta} = U_\theta \circ U_\theta^\dagger : 0 \leq \theta < 2\pi\},$$

where

$$U_\theta = (|0\rangle \rightarrow U_0)\square(|1\rangle \rightarrow e^{i\theta}U_1).$$

The non-uniqueness of the members of the guarded composition $\mathcal{E}$ is caused by the relative phase θ between U_0 and U_1.

For any two super-operators $\mathcal{E}_1$ and $\mathcal{E}_2$ in a Hilbert space $\mathcal{H}$, their sequential composition $\mathcal{E}_2; \mathcal{E}_1$ is the super-operator in $\mathcal{H}$ defined by

$$(\mathcal{E}_2; \mathcal{E}_1)(\rho) = \mathcal{E}_2(\mathcal{E}_1(\rho))$$

for any density operator ρ in $\mathcal{H}$. For any super-operator $\mathcal{E}$ and any set Ω of super-operators in Hilbert space $\mathcal{H}$, we define the sequential composition of Ω and $\mathcal{E}$ by

$$\mathcal{E}; \Omega = \{\mathcal{E}; \mathcal{F} : \mathcal{F} \in \Omega\}.$$

The following lemma can be easily derived from Lemma 3.2 (2), and it shows that the choice of orthonormal basis of the "coin space" is not essential for the guarded composition of super-operators.

Lemma 3.3 *For any two orthonormal bases $\{|i\rangle\}$ and $\{|\varphi_i\rangle\}$ of the "coin space" $\mathcal{H}_C$, there exists an unitary operator U_C such that $|\varphi_i\rangle = U_C|i\rangle$ for all i, and the two compositions along different bases $\{|i\rangle\}$ and $\{|\varphi_i\rangle\}$ are related to each other by*

$$\square_{i=1}^{n}|\varphi_i\rangle \to \mathcal{E}_i = \mathcal{E}_{U_C\otimes I_{\mathcal{H}}}; \left[(\square_{i=1}^{n}|i\rangle \to \mathcal{E}_i); \mathcal{E}_{U_C^\dagger\otimes I_{\mathcal{H}}}\right],$$

where $\mathcal{E}_{U_C\otimes I_{\mathcal{H}}}$ and $\mathcal{E}_{U_C^\dagger\otimes I_{\mathcal{H}}}$ are the super-operators in $\mathcal{H}_C\otimes\mathcal{H}$ defined by unitary operators $U_C\otimes I_{\mathcal{H}}$ and $U_C^\dagger\otimes I_{\mathcal{H}}$, respectively.

4 Semantics of QGCL

With the preparation in Section 3, we are ready to define the semantics of language QGCL. We first introduce several notations needed in this section. Let $\mathcal{H}$ and $\mathcal{H}'$ be two Hilbert spaces, and let E be an operator in $\mathcal{H}$. Then the cylindrical extension of E in $\mathcal{H}\otimes\mathcal{H}'$ is defined to be the operator $E\otimes I_{\mathcal{H}'}$, where $I_{\mathcal{H}'}$ is the identity operator in $\mathcal{H}'$. For simplicity, we will write E for $E\otimes I_{\mathcal{H}'}$ whenever confusion does not happen. Let F be an operator-valued function in $\mathcal{H}$ over Δ. Then the cylindrical extension of F in $\mathcal{H}\otimes\mathcal{H}'$ is the operator-valued function $\overline{F}$ in $\mathcal{H}\otimes\mathcal{H}'$ over Δ defined by

$$\overline{F}(\delta) = F(\delta)\otimes I_{\mathcal{H}'}$$

for every $\delta\in\Delta$. For simplicity, we often write F for $\overline{F}$ whenever confusion can be excluded from the context. Furthermore, let $\mathcal{E} = \sum_i E_i\circ E_i^\dagger$ be a super-operator in $\mathcal{H}$. Then the cylindrical extension $\overline{\mathcal{E}}$ of $\mathcal{E}$ in $\mathcal{H}\otimes\mathcal{H}'$ is defined to be the super-operator:

$$\overline{\mathcal{E}} = \sum_i (E_i\otimes I_{\mathcal{H}'})\circ(E_i^\dagger\otimes I_{\mathcal{H}'}).$$

For simplicity, $\mathcal{E}$ will be used to denote its extension $\overline{\mathcal{E}}$ when no confusion occurs. In particular, if E is an operator in $\mathcal{H}$, and ρ is a density operator in $\mathcal{H}\otimes\mathcal{H}'$, then $E\rho E^\dagger$ should be understood as $(E\otimes I_{\mathcal{H}'})\rho(E^\dagger\otimes I_{\mathcal{H}'})$.

4.1 Classical States

We now define the states of classical variables in QGCL. As already stated in Section 2, they will be only used to record the outcomes of quantum measurements.

Definition 4.1 *The (partial) classical states and their domains are inductively defined as follows:*

1. *ϵ is a classical state, called the empty state, and $dom(\epsilon)=\emptyset$;*
2. *If $x\in Var$ is a classical variable, and $a\in D_x$ is an element of the domain of x, then $[x\leftarrow a]$ is a classical state, and $dom([x\leftarrow a])=\{x\}$;*

3. *If both δ_1 and δ_2 are classical states, and $dom(\delta_1) \cap dom(\delta_2) = \emptyset$, then $\delta_1\delta_2$ is a classical state, and $dom(\delta_1\delta_2) = dom(\delta_1) \cup dom(\delta_2)$;*

4. *If δ_i is a classical state for every $1 \leq i \leq n$, then $\oplus_{i=1}^{n}\delta_i$ is a classical state, and*

$$dom(\oplus_{i=1}^{n}\delta_i) = \bigcup_{i=1}^{n} dom(\delta_i).$$

Intuitively, a classical state δ defined by clauses (1) to (3) in the above definition can be seen as a (partial) assignment to classical variables; more precisely, δ is an element of $\prod_{x\in dom(\delta)} D_x$; that is, a choice function:

$$\delta : dom(\delta) \to \bigcup_{x\in dom(\delta)} D_x$$

such that $\delta(x) \in D_x$ for every $x \in dom(\delta)$. In particular, ϵ is the empty function. Since $\prod_{x\in\emptyset} D_x = \{\epsilon\}$, ϵ is the only possible state with empty domain. The state $[x \leftarrow a]$ assigns value a to variable x but the values of the other variables are undefined. The composed state $\delta_1\delta_2$ can be seen as the assignment to variables in $dom(\delta_1) \cup dom(\delta_2)$ given by

$$(\delta_1\delta_2)(x) = \begin{cases} \delta_1(x) & \text{if } x \in dom(\delta_1), \\ \delta_2(x) & \text{if } x \in dom(\delta_2). \end{cases} \tag{26}$$

Equation (26) is well-defined since it is required that $dom(\delta_1)\cap dom(\delta_2) = \emptyset$. In particular, $\epsilon\delta = \delta\epsilon = \delta$ for any state δ, and if $x \notin dom(\delta)$ then $\delta[x \leftarrow a]$ is the assignment to variables in $dom(\delta) \cup \{x\}$ given by

$$\delta[x \leftarrow a](y) = \begin{cases} \delta(y) & \text{if } y \in dom(\delta), \\ a & \text{if } y = x. \end{cases}$$

Hence, $[x_1 \leftarrow a_1]\cdots[x_n \leftarrow a_n]$ is a classical state that assigns value a_i to variable x_i for all $1 \leq i \leq n$. It will be abbreviated to $[x_1 \leftarrow a_1, \cdots, x_n \leftarrow a_n]$ in the sequel. The state $\oplus_{i=1}^{n}\delta_i$ defined by clause (4) in Definition 4.1 can be thought of as a kind of superposition of δ_i $(1 \leq i \leq n)$. It will be used in defining the semantics of quantum alternation.

4.2 Semi-Classical Denotational Semantics

The semi-classical semantics of QGCL is a step stone for defining its purely quantum semantics. For each QGCL program P, we write $\Delta(P)$ for the set of all possible states of its classical variables. The semi-classical denotational semantics $\lceil P \rceil$ of P will be defined as an operator-valued function in $\mathcal{H}_{qvar(P)}$ over $\Delta(P)$, where $\mathcal{H}_{qvar(P)}$ is the type of quantum variables occurring in P. In particular, if $qvar(P) = \emptyset$; for example $P = \mathbf{abort}$ or $\mathbf{skip}$, then $\mathcal{H}_{qvar(P)}$ is a one-dimensional space $\mathcal{H}_\emptyset$, and an operator in $\mathcal{H}_\emptyset$ can be identified with a complex number; for instance the zero operator is number 0 and the identity operator is number 1. For any set $V \subseteq qVar$ of quantum variables, we write I_V for the identity operator in Hilbert space $\mathcal{H}_V$ (see equation (16)).

Definition 4.2 *The classical states $\Delta(P)$ and semi-classical semantic function $\lceil P \rceil$ of a QGCL program P are inductively defined as follows:*

1. $\Delta(\mathbf{abort}) = \{\epsilon\}$, *and* $\lceil \mathbf{abort} \rceil(\epsilon) = 0$;
2. $\Delta(\mathbf{skip}) = \{\epsilon\}$, *and* $\lceil \mathbf{skip} \rceil(\epsilon) = 1$;
3. $\Delta(U[\overline{q}]) = \{\epsilon\}$, *and* $\lceil U[\overline{q}] \rceil(\epsilon) = U_{\overline{q}}$, *where $U_{\overline{q}}$ is the unitary operator U acting in $\mathcal{H}_{\overline{q}}$;*
4. *If P is a classical alternation:*

$$P \stackrel{\triangle}{=} \mathbf{measure}\ (\square m \cdot M[\overline{q} : x] = m \to P_m)\ \mathbf{end},$$

where quantum measurement $M = \{M_m\}$, then

$$\Delta(P) = \bigcup_m \{\delta[x \leftarrow m] : \delta \in D(P_m)\},$$

$$\lceil P \rceil(\delta[x \leftarrow m]) = (\lceil P_m \rceil(\delta) \otimes I_{V \setminus qvar(P_m)}) \cdot (M_m \otimes I_{V \setminus \overline{q}})$$

for every $\delta \in \Delta(P_m)$ and for every outcome m, where $V = \overline{q} \cup \bigcup_m qvar(P_m)$;

5. *If P is a quantum alternation:*

$$P \stackrel{\triangle}{=} \mathbf{qif}\ [\overline{q}]\ (\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq},$$

then

$$\Delta(P) = \bigoplus_i \Delta(P_i),$$

$$\lceil P \rceil = \square_i\ |i\rangle \to \lceil P_i \rceil, \tag{27}$$

where operation $\bigoplus$ is defined by equation (23), and $\square$ in equation (27) stands for the guarded composition of operator-valued functions (see Definition 3.4);

6. *If $P = P_1; P_2$, then*

$$\Delta(P) = \Delta(P_1); \Delta(P_2) = \{\delta_1\delta_2 : \delta_1 \in \Delta(P_1) \text{ and } \delta_2 \in \Delta(P_2)\}, \tag{28}$$

$$\lceil P \rceil(\delta_1\delta_2) = (\lceil P_2 \rceil(\delta_2) \otimes I_{V \setminus qvar(P_2)}) \cdot (\lceil P_1 \rceil(\delta_1) \otimes I_{V \setminus qvar(P_1)})$$

where $V = qvar(P_1) \cup qvar(P_2)$;

Intuitively, if a quantum program P does not contain any quantum alternation, then its semantic structure can be seen as a tree with its nodes labelled by basic commands and its edges by linear operators. This tree grows up from the root in the following way: if the current node is labelled by a unitary transformation U, then a single edge stems from the node and it is labelled by U; and if the current node is labelled by a measurement $M = \{M_m\}$, then for each possible outcome m, an edge stems from the node and it is

labelled by the corresponding measurement operator M_m. Obviously, branching in the semantic tree comes from the different possible outcomes of a measurement in P. Each classical state $\delta \in \Delta(P)$ is corresponding to a branch in the semantic tree of P, and it denotes a possible path of execution. Furthermore, the value of semantic function $\lceil P \rceil$ in state δ is the (sequential) composition of the operators labelling the edges of δ. This can be clearly seen from clauses (3), (4) and (6) of the above definition. Since it is required in Definition 2.1 that $var(P_1) \cap var(P_2) = \emptyset$ in the sequential composition $P_1; P_2$, we have $dom(\delta_1) \cap dom(\delta_2) = \emptyset$ for any $\delta_1 \in \Delta(P_1)$ and $\delta_2 \in \Delta(P_2)$. Thus, equation (28) is well-defined.

The semantic structure of a quantum program P with quantum alternations is much more complicated. We can imagine it as a tree with superpositions of nodes that generate superpositions of branches. The value of semantic function $\lceil P \rceil$ in a superposition of branches is then defined as the guarded composition of the values in these branches.

4.3 Purely Quantum Denotational Semantics

Now the purely quantum semantics of a quantum program can be naturally defined as the super-operator induced by its semi-classical semantic function.

Definition 4.3 *For each QGCL program P, its purely quantum denotational semantics is the super-operator $[\![P]\!]$ in $\mathcal{H}_{qvar(P)}$ defined as follows:*

$$[\![P]\!] = \mathcal{E}(\lceil P \rceil) = \sum_{\delta \in \Delta(P)} \lceil P \rceil(\delta) \circ \lceil P \rceil(\delta)^\dagger. \tag{29}$$

The following proposition presents a representation of the purely quantum semantics of a program in terms of its subprograms.

Proposition 4.1 *1.* $[\![\mathbf{abort}]\!] = 0$;

2. $[\![\mathbf{skip}]\!] = 1$;

3. $[\![P_1; P_2]\!] = [\![P_1]\!]; [\![P_2]\!]$;

4. $[\![U[\overline{q}]]\!] = U_{\overline{q}} \circ U_{\overline{q}}^\dagger$;

5.

$$[\![\mathbf{measure}\ (\square m \cdot M[\overline{q} : x] = m \to P_m)\ \mathbf{end}]\!] = \sum_m \left[(M_m \circ M_m^\dagger); [\![P_m]\!] \right].$$

Here, $[\![P_m]\!]$ should be seen as a cylindrical extension in $\mathcal{H}_V$ from $\mathcal{H}_{qvar(P_m)}$, $M_m \circ M_m^\dagger$ is seen as a cylindrical extension in $\mathcal{H}_V$ from $\mathcal{H}_{\overline{q}}$, and $V = \overline{q} \cup \bigcup_m qvar(P_m)$;

6.

$$[\![\mathbf{qif}\ [\overline{q}]\ (\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq}]\!] \in \square_i\ |i\rangle \to [\![P_i]\!]. \tag{30}$$

Here $[\![P_i]\!]$ should be understood as a cylindrical extension in $\mathcal{H}_V$ from $\mathcal{H}_{qvar(P_i)}$ for every $1 \leq i \leq n$, and $V = \overline{q} \cup \bigcup_i qvar(P_i)$.

The above proposition shows that the purely quantum denotational semantics is almost compositional, but it is not completely compositional because the symbol "$\in$" appears in equation (30) of the above proposition. The symbol "$\in$" can be understood as a refinement relation. It is worth noting that in general "$\in$" cannot be replaced by equality. This is exactly the reason that the purely quantum semantics of a program has to be derived through its semi-classical semantics but cannot be defined directly by a structural induction.

It should be stressed that the symbol "$\in$" in equation (30) does not mean that the purely quantum denotational semantics of quantum alternation "$\mathbf{qif}\ [\overline{q}]\ (\square i \cdot |i\rangle \rightarrow P_i)\ \mathbf{fiq}$" is not well-defined. In fact, it is uniquely defined by equations (27) and (29) as a super-operator. The right-hand side of equation (30) is not the denotational semantics of any program. It is the guarded composition of the denotational semantics of programs P_i. Since it is the guarded composition of a family of super-operators, it can be a set consisting of more than one super-operator, as shown in Example 3.4. The semantics of quantum alternation "$\mathbf{qif}\ [\overline{q}]\ (\square i \cdot |i\rangle \rightarrow P_i)\ \mathbf{fiq}$" is one member of the set of super-operators in the right-hand side of equation (30).

Equivalence relation between quantum programs can be introduced based on their purely quantum denotational semantics.

Definition 4.4 *Let P and Q be two QGCL programs. Then:*

1. *We say that P and Q are equivalent and write $P \equiv Q$ if*

$$[\![P]\!] \otimes \mathcal{I}_{Q \setminus P} = [\![Q]\!] \otimes \mathcal{I}_{P \setminus Q},$$

   where $\mathcal{I}_{Q \setminus P}$ is the identity super-operator in $\mathcal{H}_{qvar(Q) \setminus qvar(P)}$ and $\mathcal{I}_{P \setminus Q}$ the identity super-operator in $\mathcal{H}_{qvar(P) \setminus qvr(Q)}$.

2. *The "coin-free"equivalence $P \equiv_{CF} Q$ holds if*

$$tr_{\mathcal{H}_{cvar(P) \cup cvar(Q)}}([\![P]\!] \otimes \mathcal{I}_{Q \setminus P}) = tr_{\mathcal{H}_{cvar(P) \cup cvar(Q)}}([\![Q]\!] \otimes \mathcal{I}_{P \setminus Q}). \tag{31}$$

If $qvar(P) = qvar(Q)$, then $P \equiv Q$ if and only if $[\![P]\!] = [\![Q]\!]$, and $P \equiv_{CF} Q$ if and only if $tr_{\mathcal{H}_{cvar(P)}}([\![P]\!]) = tr_{\mathcal{H}_{cvar(P)}}[\![Q]\!]$. The symbol "$tr$" in equation (36) denotes partial trace, which is defined s follows: let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two Hilbert spaces and let $\{|\varphi_i\rangle\}$ be an orthonormal basis of $\mathcal{H}_1$. Then for any density operator ρ in $\mathcal{H}_1 \otimes \mathcal{H}_2$,

$$tr_{\mathcal{H}_1}(\rho) = \sum_i \langle \varphi_i | \rho | \varphi_i \rangle \tag{32}$$

is a density operator in $\mathcal{H}_2$. Furthermore, for any super-operator $\mathcal{E}$ on $\mathcal{H}_1 \otimes \mathcal{H}_2$, $tr_{\mathcal{H}_1}(\mathcal{E})$ is a super-operator from $\mathcal{H}_1 \otimes \mathcal{H}_2$ to $\mathcal{H}_2$ defined by $tr_{\mathcal{H}_1}(\mathcal{E})(\rho) = tr_{\mathcal{H}_1}(\mathcal{E}(\rho))$ for all density operators ρ in $\mathcal{H}_1 \otimes \mathcal{H}_2$. In a sense, "coin" variables are only used to realize superposition of programs. The computational outcome of a program P is stored in the "principal" state space $\mathcal{H}_{qvar(P) \setminus cvar(P)}$. This is exactly the reason why we introduce the notion of "coin-free"equivalence. Obviously, $P \equiv Q$ implies $P \equiv_{CF} Q$.

4.4 Weakest Precondition Semantics

The notions of Hoare triple for a quantum program and quantum weakest precondition were proposed by D'Hondt and Panangaden in [16]. We now recall their definitions from [16].

Definition 4.5 *Let P be a program, and let N_1 and N_2 be (bounded) positive (Hermitian) operators in $\mathcal{H}_{qvar(P)}$.*

1. *If*
$$tr(N_1\rho) \leq tr(N_2[\![P]\!](\rho))$$
for all density operators ρ in $\mathcal{H}_{qvar(P)}$, then N_1 is called a precondition of N_2 and N_2 a postcondition of N_1 with respect to P, and we write
$$\{N_1\}P\{N_2\}. \tag{33}$$
Equation (33) is called a (quantum) Hoare triple.

2. *N_2 is called the weakest precondition of N_1 with respect to P, written $N_2 = wp.P.N_1$ if*
   (a) *N_2 is a precondition of N_1 with respect to P; and*
   (b) *$N' \sqsubseteq N_2$ whenever N' is a also precondition of N_1 with respect to P, where $\sqsubseteq$ stands for the Löwner order.*

Remark 4.1 *In the original definition of quantum weakest precondition in [16], N_1 and N_2 are required to be so-called quantum predicates; i.e. Hermitian operators whose eigenvalues are in the unit interval $[0, 1]$. However, this constraint is not essential, and thus it was removed in the above definition. Allowing N_1 and N_2 to be any (bounded) positive operators is indeed consistent with the treatment of probabilistic weakest preconditions in [25, 29], where a probabilistic precondition or postcondition was understood as the expectation of a random variable.*

For each program P, $wp.P$ can be seen as the super-operator in $\mathcal{H}_{qvar(P)}$ defined as follows: for any positive operator N,

$$(wp.P)(N) = wp.P.N$$

is given by clause (2) of the above definition, and $wp.P$ can be extended to the whole space of bounded operators in $\mathcal{H}_{qvar(P)}$ by linearity.

The weakest precondition semantics of QGCL programs can be derived from Proposition 4.1, and they are given in the next proposition.

Proposition 4.2 1. $wp.\mathbf{abort} = 0$;

2. $wp.\mathbf{skip} = 1$;

3. $wp.(P_1; P_2) = wp.P_2; wp.P_1$;

4. $wp.U[\overline{q}] = U_{\overline{q}}^{\dagger} \circ U_{\overline{q}}$;

5. $wp.\mathbf{measure}\ (\square m \cdot M[\overline{q} : x] = m \to P_m)\ \mathbf{end} = \sum_m \left[wp.P_m; (M_m^{\dagger} \circ M_m)\right]$;

6. $wp.\mathbf{qif}\ [\overline{q}]\ (\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq} \in \square_i\ |i\rangle \to wp.P_i$.

Some cylindrical extensions of super-operators are used but unspecified in the above proposition because they can be recognised from the context. Again, "$\in$" in the above clause (6) cannot be replaced by equality because the right-hand side of clause (6) is a set that may contain more than one super-operator.

We can define the refinement relation between quantum programs in terms of their weakest precondition semantics. To this end, we first generalize the Löwner order to the case of super-operators: for any two super-operators $\mathcal{E}$ and $\mathcal{F}$ in Hilbert space $\mathcal{H}$, $\mathcal{E} \sqsubseteq \mathcal{F}$ if and only if $\mathcal{E}(\rho) \sqsubseteq \mathcal{F}(\rho)$ for all density operators ρ in $\mathcal{H}$.

Definition 4.6 *Let P and Q be two programs. Then we say that P is refined by Q and write $P \sqsubseteq Q$ if*

$$wp.P \otimes \mathcal{I}_{Q \setminus P} \sqsubseteq wp.Q \otimes \mathcal{I}_{P \setminus Q},$$

where $\mathcal{I}_{Q \setminus P}$ and $\mathcal{I}_{P \setminus Q}$ are the same as in Deinition 4.4.

It is easy to see that $P \sqsubseteq Q$ and $Q \sqsubseteq P$ implies $P \equiv Q$. Here, we are not going to further consider how can refinement technique be used in quantum programming, but leave it as a topic for future research.

4.5 An Example

To conclude this section, we present a simple example that helps us to understand the semantic notions introduced above.

Example 4.1 *Let q be a qubit variable and x, y two classical variables. Consider the*

QGCL program

$$
\begin{aligned}
P \stackrel{\triangle}{=} \mathbf{qif}\ |0\rangle \rightarrow\ & H[q]; \\
& \mathbf{measure}\ M^{(0)}[x \leftarrow q] = 0 \rightarrow X[q]; \\
& \quad \square \qquad\qquad\quad 1 \rightarrow Y[q] \\
& \mathbf{end} \\
\square\ |1\rangle \rightarrow\ & S[q]; \\
& \mathbf{measure}\ M^{(1)}[x \leftarrow q] = 0 \rightarrow Y[q] \\
& \quad \square \qquad\qquad\quad 1 \rightarrow Z[q] \\
& \mathbf{end}; \\
& X[q]; \\
& \mathbf{measure}\ M^{(0)}[y \leftarrow q] = 0 \rightarrow Z[q] \\
& \quad \square \qquad\qquad\quad 1 \rightarrow X[q] \\
& \mathbf{end} \\
\mathbf{fiq}
\end{aligned}
$$

where $M^{(0)}, M^{(1)}$ are the measurements on a qubit in computational basis $|0\rangle, |1\rangle$ and basis $|\pm\rangle$, respectively (see Example 3.3), H is the Hadamard gate,

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},\ Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

are the Pauli matrices, and

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

is the phase gate. The program P is a quantum alternation between two subprograms P_0 and P_1. The first subprogram P_0 is the Hadamard gate followed by the measurement in the computational basis, where whenever the outcome is 0, then the gate X follows; whenever the outcome is 1, then the gate Y follows. The second subprogram P_1 is the gate S followd by the measurement in basis $|\pm\rangle$, the gate X, and the measurement in the computational basis.

We write a for classical state $[x \leftarrow a]$ of program P_0 and bc for classical state $[x \leftarrow b, y \leftarrow c]$ of program P_1 for any $a, c \in \{0, 1\}$ and $b \in \{+, -\}$. Then the semi-classical semantic functions of P_0 and P_1 are given as follows:

$$\begin{cases} \lceil P_0 \rceil(0) = X \cdot |0\rangle\langle 0| \cdot H = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \\ \lceil P_0 \rceil(1) = Y \cdot |1\rangle\langle 1| \cdot H = \frac{i}{\sqrt{2}} \begin{pmatrix} -1 & 1 \\ 0 & 0 \end{pmatrix}, \end{cases}$$

$$\begin{cases} \lceil P_1 \rceil(+0) = Z \cdot |0\rangle\langle 0| \cdot X \cdot Y \cdot |+\rangle\langle +| \cdot S = \frac{1}{2} \begin{pmatrix} i & -1 \\ 0 & 0 \end{pmatrix}, \\ \lceil P_1 \rceil(+1) = X \cdot |1\rangle\langle 1| \cdot X \cdot Y \cdot |+\rangle\langle +| \cdot S = \frac{1}{2} \begin{pmatrix} -i & 1 \\ 0 & 0 \end{pmatrix}, \\ \lceil P_1 \rceil(-0) = Z \cdot |0\rangle\langle 0| \cdot X \cdot Z \cdot |-\rangle\langle -| \cdot S = \frac{1}{2} \begin{pmatrix} 1 & -i \\ 0 & 0 \end{pmatrix}, \\ \lceil P_1 \rceil(-1) = X \cdot |1\rangle\langle 1| \cdot X \cdot Z \cdot |-\rangle\langle -| \cdot S = \frac{1}{2} \begin{pmatrix} 1 & -i \\ 0 & 0 \end{pmatrix}. \end{cases}$$

The semi-classical semantic function of P is an operator-valued function in the state space of two qubits over classical states $\Delta(P) = \{a \oplus bc : a, c \in \{0,1\} \text{ and } b \in \{+,-\}\}$. It follows from equation (24) that

$$\lceil P \rceil(a \oplus bc)(|0\rangle|\varphi\rangle) = \lambda_{1(bc)}|0\rangle(\lceil P_0 \rceil(a)|\varphi\rangle),$$
$$\lceil P \rceil(a \oplus bc)(|1\rangle|\varphi\rangle) = \lambda_{0a}|1\rangle(\lceil P_1 \rceil(bc)|\varphi\rangle),$$

where $\lambda_{0a} = \frac{1}{\sqrt{2}}$ and $\lambda_{1(bc)} = \frac{1}{2}$ for $a, c \in \{0,1\}$ and $b \in \{+,-\}$. Using

$$\lceil P \rceil(a \oplus bc) = \sum_{i,j \in 0,1} (\lceil P \rceil(a \oplus bc)|ij\rangle)\langle ij|,$$

we can compute:

$$\lceil P \rceil(0 \oplus +0) = \frac{1}{2\sqrt{2}} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}, \quad \lceil P \rceil(0 \oplus +1) = \frac{1}{2\sqrt{2}} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -i & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$$\lceil P \rceil(0 \oplus -0) = \lceil P \rceil(0 \oplus -1) = \frac{1}{2\sqrt{2}} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -i & 0 \end{pmatrix},$$

$$\lceil P \rceil(1 \oplus +0) = \frac{1}{2\sqrt{2}} \begin{pmatrix} -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}, \quad \lceil P \rceil(1 \oplus +1) = \frac{1}{2\sqrt{2}} \begin{pmatrix} -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -i & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$$\lceil P \rceil(1 \oplus -0) = \lceil P \rceil(1 \oplus -1) = \frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -i & 0 \end{pmatrix}.$$

Then the purely quantum semantics of program P is the super-operator:

$$[\![P]\!] = \sum_{a,c \in \{0,1\} \text{ and } b \in \{+,-\}} E_{abc} \circ E_{abc}^\dagger,$$

and it follows from the proof of Proposition 3.3 in [16] that the weakest precondition semantics of P is the super-operator

$$wp.P = \sum_{a,c\in\{0,1\} \text{ and } b\in\{+,-\}} E_{abc}^{\dagger} \circ E_{abc},$$

where $E_{abc} = \lceil P \rceil (a \oplus bc)$.

5 Quantum Choice

As discussed in Subsection 1.4, quantum alternation and choice are two ingredients in the realization of the quantum programming paradigm of superposition of programs. But only quantum alternation was introduced as a primitive program construct in the syntax of QGCL. Indeed, as already explained in Subsection 1.4, quantum choice may be easily defined as a derived program construct from quantum alternation.

Definition 5.1 *Let P be a program such that $\overline{q} = qvar(P)$, and let P_i be programs for all i . If $\{|i\rangle\}$ is an orthonormal basis of $\mathcal{H}_{\overline{q}}$, and $\overline{q} \cap \bigcup_i qvar(P_i) = \emptyset$, then the quantum choice of P_i's according to P along the basis $\{|i\rangle\}$ is defined as*

$$[P] \left(\bigoplus_i |i\rangle \rightarrow P_i \right) \stackrel{\triangle}{=} P; \mathbf{qif}\ [\overline{q}]\ (\square i \cdot |i\rangle \rightarrow P_i)\ \mathbf{fiq}.$$

In particular, if $n = 2$, then the quantum choice will be abbreviated to $P_0\ {}_P \oplus P_1$.

Since the quantum choice of $P_1, ..., P_n$ is defined in terms of their quantum alternation, the semantics of the former can be directly derived from that of the latter.

5.1 Quantum Implementation of Probabilistic Choice

The relationship between probabilistic choice and quantum choice was briefly discussed at the end of Subsection 1.4. Now it is the right time to examine this relationship in a more precise way. To this end, we first expand the syntax and semantics of QGCL to include probabilistic choice [29].

Definition 5.2 *Let P_i be a QGCL program for each $1 \leq i \leq n$, and let $\{p_i\}_{i=1}^n$ be a sub-probability distribution; that is, $p_i > 0$ for each $1 \leq i \leq n$ and $\sum_{i=1}^n p_i \leq 1$. Then*

1. *The probabilistic choice of $P_1, ..., P_n$ according to $\{p_i\}_{i=1}^n$ is*

$$\sum_{i=1}^{n} P_i @ p_i.$$

2. *The quantum variables of the choice are:*

$$qvar\left(\sum_{i=1}^{n} P_i@p_i\right) = \bigcup_{i=1}^{n} qvar(P_i).$$

3. *The purely quantum denotational semantics of the choice is:*

$$\left[\!\left[\sum_{i=1}^{n} P_i@p_i\right]\!\right] = \sum_{i=1}^{n} p_i \cdot [\![P_i]\!]. \tag{34}$$

The right-hand side of equation (34) is the probabilistic combination of super-operators $[\![P_i]\!]$ according to distribution $\{p_i\}$; that is,

$$\left(\sum_{i=1}^{n} p_i \cdot [\![P_i]\!]\right)(\rho) = \sum_{i=1}^{n} p_i \cdot [\![P_i]\!](\rho)$$

for all density operators ρ. It is obvious that $\sum_{i=1}^{n} p_i \cdot [\![P_i]\!]$ is a super-operator too.

A clear description about the relationship between probabilistic choice and quantum choice requires us to further expand the syntax and semantics of QGCL by introducing block command with local quantum variables.

Definition 5.3 *Let P be a QGCL program, let $\overline{q} \subseteq qvar(P)$ be a sequence of quantum variables, and let ρ be a density operator in $\mathcal{H}_{\overline{q}}$. Then*

1. *The block command defined by P restricted to $\overline{q} = \rho$ is:*

$$\mathbf{begin\ local}\ \overline{q} := \rho; P\ \mathbf{end}. \tag{35}$$

2. *The quantum variables of the block command are:*

$$qvar\left(\mathbf{begin\ local}\ \overline{q} := \rho; P\ \mathbf{end}\right) = qvar(P) \setminus \overline{q}.$$

3. *The purely quantum denotational semantics of the block command is give as follows:*

$$[\![\mathbf{begin\ local}\ \overline{q} := \rho; P\ \mathbf{end}]\!](\sigma) = tr_{\mathcal{H}_{\overline{q}}}([\![P]\!](\sigma \otimes \rho)) \tag{36}$$

for any density operator σ in $\mathcal{H}_{qvar(P)\setminus\overline{q}}$.

The intuitive meaning of block command (35) is that program P is running in the environment where $\overline{q}$ are local variables and they are initialized in state ρ before the execution of P. The symbol "tr" in equation (35) is partial trace defined by equation (32). So, after executing P, the auxiliary system denoted by the local variables $\overline{q}$ is discarded.

We present a simple example to illustrate the above two definitions.

Example 5.1 *(Continuation of Example 3.3; Probabilistic mixture of measurements) It is often required in quantum cryptographic protocols like BB84 to randomly choose between the measurement $M^{(0)}$ on a qubit in the computational basis and the measurement $M^{(1)}$ in the basis $|\pm\rangle$. Here we consider a simplified version of random choice between $M^{(0)}$ and $M^{(1)}$. If we perform measurement $M^{(0)}$ on qubit q in state $|\psi\rangle$ and discard the outcomes of measurement, then we get*

$$\rho_0 = M_0^{(0)}|\psi\rangle\langle\psi|M_0^{(0)} + M_1^{(0)}|\psi\rangle\langle\psi|M_1^{(0)},$$

and if we perform measurement $M^{(1)}$ on $|\psi\rangle$ and discard the outcomes, then we get

$$\rho_1 = M_+^{(1)}|\psi\rangle\langle\psi|M_+^{(1)} + M_-^{(1)}|\psi\rangle\langle\psi|M_-^{(1)}.$$

We now take the unitary matrix

$$U = \begin{pmatrix} \sqrt{p} & \sqrt{r} \\ \sqrt{r} & -\sqrt{p} \end{pmatrix}$$

where $p, r \geq 0$ and $p + r = 1$, and introduce a "coin" qubit q_C. Let

$$\begin{aligned} P_i \stackrel{\triangle}{=} \mathbf{measure}\ M^{(i)}[x \leftarrow q] = 0 &\rightarrow \mathbf{skip} \\ \square \qquad\qquad\quad\ \ 1 &\rightarrow \mathbf{skip} \\ \mathbf{end} & \end{aligned}$$

for $i = 0, 1$, and put quantum choice of P_0 and P_1 according the "coin tossing operator" U into a block with the "coin" qubit q_C as a local variable:

$$P \stackrel{\triangle}{=} \mathbf{begin\ local}\ q_C := |0\rangle; P_0\ {}_{U[q_C]} \oplus P_1\ \mathbf{end}$$

Then for any $|\psi\rangle \in \mathcal{H}_q$, $i \in \{0, 1\}$ and $j \in \{+, -\}$, we have:

$$|\psi_{ij}\rangle \stackrel{\triangle}{=} M_{ij}(U|0\rangle|\psi\rangle) = \sqrt{\frac{p}{2}}|0\rangle M_i^{(0)}|\psi\rangle + \sqrt{\frac{r}{2}}|1\rangle M_j^{(1)}|\psi\rangle,$$

$$\begin{aligned} [\![P]\!](|\psi\rangle\langle\psi|) &= tr_{\mathcal{H}_{q_C}} \left(\sum_{i\in\{0,1\} \text{ and } j\in\{+,-\}} |\psi_{ij}\rangle\langle\psi_{ij}| \right) \\ &= 2 \sum_{i\in\{0,1\}} \frac{p}{2} M_i^{(0)}|\psi\rangle\langle\psi|M_i^{(0)} + 2 \sum_{j\in\{+,-\}} \frac{r}{2} M_j^{(1)}|\psi\rangle\langle\psi|M_j^{(1)} \\ &= p\rho_0 + r\rho_1. \end{aligned}$$

So, program P can be seen as a probabilistic mixture of measurements $M^{(0)}$ and $M^{(1)}$.

Now we are ready to precisely characterize the relationship between probabilistic choice and quantum choice. Roughly speaking, if the "coin" variables are treated as local variables, then a quantum choice degenerates to a probabilistic choice.

Theorem 5.1 *Let $qvar(P) = \overline{q}$. Then we have:*

$$\mathbf{begin\ local}\ \overline{q} := \rho; [P] \left(\bigoplus_{i=1}^{n} |i\rangle \rightarrow P_i \right)\ \mathbf{end} \equiv \sum_{i=1}^{n} P_i @ p_i \tag{37}$$

where $p_i = \langle i|[\![P]\!](\rho)|i\rangle$ for every $1 \leq i \leq n$.

The inverse of the above theorem is also true. For any probability distribution $\{p_i\}_{i=1}^n$, we can find an $n \times n$ unitary operator U such that $p_i = |U_{i0}|^2$ $(1 \leq i \leq n)$. So, it follows immediately from the above theorem that a probabilistic choice $\sum_{i=1}^n P_i @ p_i$ can always be implemented by a quantum choice:

$$\mathbf{begin\ local}\ \overline{q} := |0\rangle; [U[\overline{q}]] \left(\bigoplus_{i=1}^{n} |i\rangle \rightarrow P_i \right)\ \mathbf{end}$$

where $\overline{q}$ is a family of new quantum variables with an $n-$dimensional state space. As said in Subsection 1.1, probabilistic choice (4) can be thought of as a refinement of nondeterministic choice (3). Since for a given probability distribution $\{p_i\}$, there are more than one "coin program" P to implement the probabilistic choice $\sum_{i=1}^n P_i @ p_i$ in equation (37), a quantum choice can be further seen as a refinement of a probabilistic choice where a specific "device" (quantum "coin") is explicitly given for generating the distribution $\{p_i\}$.

6 Algebraic Laws

In this section, we present a group of basic algebraic laws for quantum alternation and choice, which will be useful for verification, transformation and compilation of quantum programs. The laws given in the following theorem shows that quantum alternation is idempotent, commutative and associative and sequential composition is distributive over quantum alternation from the right.

Theorem 6.1 *(Laws for Quantum Alternation)*

1. *Idempotent Law: If $P_i \equiv P$ for all i, then*

$$\mathbf{qif}\ (\square i \cdot |i\rangle \rightarrow P_i)\ \mathbf{fiq} \equiv P.$$

2. *Commutative Law: For any permutation τ of $\{1, ..., n\}$, we have:*

$$\mathbf{qif}\ [\overline{q}]\ \left(\square_{i=1}^n i \cdot |i\rangle \rightarrow P_{\tau(i)}\right)\ \mathbf{fiq} \equiv U_{\tau^{-1}}[\overline{q}]; \mathbf{qif}\ [\overline{q}]\ (\square_{i=1}^n i \cdot |i\rangle \rightarrow P_i)\ \mathbf{fiq}; U_\tau[\overline{q}],$$

where τ^{-1} is the inverse of τ, i.e. $\tau^{-1}(i) = j$ if and only if $\tau(j) = i$ for $i, j \in \{1, ..., n\}$, and U_τ (resp. $U_{\tau^{-1}}$) is the unitary operator permutating the basis $\{|i\rangle\}$ of $\mathcal{H}_{\overline{q}}$ with τ (resp. τ^{-1}); that is, $U_\tau(|i\rangle) = |\tau(i)\rangle$ (resp. $U_{\tau^{-1}}(|i\rangle) = |\tau^{-1}(i)\rangle$) for every $1 \leq i \leq n$.

3. *Associative Law:*

$$\mathbf{qif}\ (\square i\cdot|i\rangle\rightarrow \mathbf{qif}\ (\square j_i\cdot|j_i\rangle\rightarrow P_{ij_i})\ \mathbf{fiq})\ \mathbf{fiq}\equiv \mathbf{qif}\ (\overline{\alpha})\,(\square i,j_i\cdot|i,j_i\rangle\rightarrow P_{ij_i})\ \mathbf{fiq}$$

for some family $\overline{\alpha}$ of parameters, where the right-hand side is a parameterized quantum alternation defined in Appendix A.

4. *Distributive Law: If $\overline{q}\cap qvar(Q)=\emptyset$, then*

$$\mathbf{qif}\ [\overline{q}]\,(\square i\cdot|i\rangle\rightarrow P_i)\ \mathbf{fiq};Q\equiv_{CF}\mathbf{qif}\ (\overline{\alpha})[\overline{q}]\,(\square i\cdot|i\rangle\rightarrow (P_i;Q))\,\mathbf{fiq}$$

for some family $\overline{\alpha}$ of parameters, where the right-hand side is a parameterized quantum alternation. In particular, if we further assume that Q contains no measurements, then

$$\mathbf{qif}\ [\overline{q}]\,(\square i\cdot|i\rangle\rightarrow P_i)\ \mathbf{fiq};Q\equiv \mathbf{qif}\ [\overline{q}]\,(\square i\cdot|i\rangle\rightarrow (P_i;Q))\,\mathbf{fiq}.$$

A quantum choice is defined as a "coin" program followed by a quantum alternation. A natural question would be: is it possible to move the "coin" program to the end of a quantum alternation? The following theorem positively answers this question under the condition that encapsulation in a block with local variables is allowed.

Theorem 6.2 *For any programs P_i and unitary operator U, we have:*

$$[U[\overline{q}]]\left(\bigoplus_{i=1}^{n}|i\rangle\rightarrow P_i\right)\equiv \mathbf{qif}\ (\square i\cdot U_{\overline{q}}^{\dagger}|i\rangle\rightarrow P_i)\ \mathbf{fiq};U[\overline{q}]. \tag{38}$$

More generally, for any programs P_i and P with $\overline{q}=qvar(P)$, there are new quantum variables $\overline{r}$, a pure state $|\varphi_0\rangle\in\mathcal{H}_{\overline{r}}$, an orthonormal basis $\{|\psi_{ij}\rangle\}$ of $\mathcal{H}_{\overline{q}}\otimes\mathcal{H}_{\overline{r}}$, programs Q_{ij}, and a unitary operator U in $\mathcal{H}_{\overline{q}}\otimes\mathcal{H}_{\overline{r}}$ such that

$$\begin{aligned}[P]\left(\bigoplus_{i=1}^{n}|i\rangle\rightarrow P_i\right)\equiv\ &\mathbf{begin\ local}\ \overline{r}:=|\varphi_0\rangle;\\ &\mathbf{qif}\ (\square i,j\cdot|\psi_{ij}\rangle\rightarrow Q_{ij})\ \mathbf{fiq};\\ &U[\overline{q},\overline{r}]\\ \mathbf{end}.&\end{aligned} \tag{39}$$

The next theorem shows that quantum choice is also idempotent, commutative and associative and sequential composition is distributive over quantum choice from the right.

Theorem 6.3 *(Laws for Quantum Choice)*

1. *Idempotent Law: If $qvar(Q)=\overline{q}$, $tr[\![Q]\!](\rho)=1$ and $P_i\equiv P$ for all $1\leq i\leq n$, then*

$$\mathbf{begin\ local}\ \overline{q}:=\rho;[Q]\left(\bigoplus_{i=1}^{n}|i\rangle\rightarrow P_i\right)\ \mathbf{end}\equiv P.$$

2. *Commutative Law: For any permutation τ of $\{1, ..., n\}$, we have:*

$$[P]\left(\bigoplus_{i=1}^{n}|i\rangle \to P_{\tau(i)}\right) \equiv [P; U_\tau[\overline{q}]]\left(\bigoplus_{i=1}^{n}|i\rangle \to P_i\right); U_{\tau^{-1}}[q],$$

where $qvar(P) = \overline{q}$, and U_τ, $U_{\tau^{-1}}$ are the same as in Theorem 6.1 (2).

3. *Associative Law: Let $\Gamma = \{(i, j_i) : 1 \leq i \leq m \text{ and } 1 \leq j_i \leq n_i\} = \bigcup_{i=1}^{m}(\{i\} \times \{1, ..., n_i\})$, and*

$$R = [P]\left(\bigoplus_{i=1}^{n}|i\rangle \to Q_i\right).$$

Then

$$[P]\left(\bigoplus_{i=1}^{m}|i\rangle \to [Q_i]\left(\bigoplus_{j_i=1}^{n_i}|j_i\rangle \to R_{ij_i}\right)\right) \equiv [R(\overline{\alpha})]\left(\bigoplus_{(i,j_i)\in\Gamma}|i, j_i\rangle \to R_{ij_i}\right),$$

for some family $\overline{\alpha}$ of parameters, where the right-hand side is a parameterized quantum choice defined in Appendix A.

4. *Distributive Law: If $qvar(P) \cap qvar(Q) = \emptyset$, then*

$$[P]\left(\bigoplus_{i=1}^{n}|i\rangle \to P_i\right); Q \equiv_{CF} [P(\overline{\alpha})]\left(\bigoplus_{i=1}^{n}|i\rangle \to (P_i; Q)\right)$$

for some family $\overline{\alpha}$ of parameters, where the right-hand side is a parameterized quantum choice. In particular, if we further assume that Q contains no measurements, then

$$[P]\left(\bigoplus_{i=1}^{n}|i\rangle \to P_i\right); Q \equiv [P]\left(\bigoplus_{i=1}^{n}|i\rangle \to (P_i; Q)\right).$$

7 Illustrative Examples

The design of the language QGCL, in particular the definition of quantum alternation and choice, was inspired by the construction of some simplest quantum walks. A large number of variants and generalizations of quantum walks have been introduced in the last decade. Quantum walks have been widely used in the development of quantum algorithms including quantum simulation. It was proved that quantum walks are indeed universal for quantum computation [14, 28]. Furthermore, experimental implementations of quantum walks have also been conducting in the laboratories over the world. Various extended quantum walks in the literature can be conveniently written as QGCL programs with quantum alternation and choice. Here, we present several simple examples of quantum walks to further show the expressive power of QGCL.

Example 7.1 *One of the simplest random walks is the one-dimensional walk where a walker moves to the left with probability $\frac{1}{2}$ and moves to the right with the same probability. The Hadamard walk considered in [5] is a quantum generalization of this random walk. Let p, c be the quantum variables for position and coin, respectively. The type of variable p is the infinite-dimensional Hilbert space*

$$\mathcal{H}_p = span\{|n\rangle : n \in \mathbb{Z}\ (\text{integers})\} = \{\sum_{\mathrm{n}=-\infty}^{\infty} \alpha_{\mathrm{n}}|\mathrm{n}\rangle : \sum_{\mathrm{n}=-\infty}^{\infty} |\alpha_{\mathrm{n}}|^2 < \infty\},$$

and the type of c is the $2-$dimensional Hilbert space $\mathcal{H}_c = span\{|L\rangle, |R\rangle\}$, where L, R stand for Left and Right, respectively. So, the state space of a walker on a line is $\mathcal{H} = \mathcal{H}_c \otimes \mathcal{H}_p$. We write $I_{\mathcal{H}_p}$ for the identity operator in $\mathcal{H}_p$. Let H be the 2×2 Hadamard matrix (see equation (9)), and let T_L, T_R be left- and right-translation, respectively; that is,

$$T_L|n\rangle = |n-1\rangle, \qquad T_R|n\rangle = |n+1\rangle$$

for every $n \in \mathbb{Z}$. Then a single step of the Hadamard walk can be described by the unitary operator

$$W = (|L\rangle\langle L| \otimes T_L + |R\rangle\langle R| \otimes T_R)(H \otimes I_{\mathcal{H}_p}). \tag{40}$$

It can also be written as the QGCL program:

$$T_L[p]_{H[c]} \oplus T_R[p].$$

This program is the quantum choice of the left-translation T_L and the right-translation T_R according to the "coin" program $H[c]$. The Hadamard walk continuously runs this programs. The following are several variants of this walk considered in the recent physics literature.

1. *A simple variant of the above Hadamard walk is the unidirectional quantum walk examined in [30], where the walk either moves to the right or stays in the previous position. So, the left-translation T_L should be replaced by the program* **skip** *whose semantics is the identity operator $I_{\mathcal{H}_p}$, and a single step of the new quantum walk can be written as the QGCL program:*

   $$\mathbf{skip}_{H[c]} \oplus T_R[p].$$

   It is a quantum choice of **skip** *and the right-translation T_R.*

2. *A feature of the original one-dimensional quantum walk and its unidirectional variant is that the coin operator H is independent of the position and time. A new kind of quantum walk was employed in [26] to implement quantum measurement. The coin tossing operator of this walk depends on both position n and time t:*

   $$C(n,t) = \frac{1}{\sqrt{2}} \begin{pmatrix} c(n,t) & s(n,t) \\ s^*(n,t) & -e^{i\theta}c(n,t) \end{pmatrix}.$$

Then for a given time t, step t of the walk can be written as the QGCL program:

$$W_t \stackrel{\triangle}{=} \mathbf{qif}\ [p](\square n \cdot |n\rangle \to C(n,t)[c])\ \mathbf{fiq};$$
$$\mathbf{qif}\ [c](|L\rangle \to T_L[p])\square(|R\rangle \to T_R[p])\ \mathbf{fiq}.$$

The program W_t is a sequential composition of two quantum alternations. Since W_t may be different for different time points t, the first T steps can be written as the program:

$$W_1; W_2; ...; W_T.$$

3. *Another simple generalization of the original and unidirectional one-dimensional quantum walk is the quantum walk with three coin states considered in [22]. The coin space of this walk is a $3-$dimensional Hilbert space $\mathcal{H}_c = span\{|L\rangle, |0\rangle, |R\rangle\}$, where L and R are used to indicate moving to the left and to the right, respectively, as before, but 0 means staying at the previous position. The "coin tossing" operator is the unitary*

$$U = \frac{1}{3}\begin{pmatrix} -1 & 2 & 2 \\ 2 & -1 & 2 \\ 2 & 2 & -1 \end{pmatrix}.$$

Then a single step of the walk can be written as the QGCL program:

$$[U[c]]\,(|L\rangle \to T_L[p] \oplus |0\rangle \to \mathbf{skip} \oplus |R\rangle \to T_R[p])\,.$$

This is the quantum choice of $\mathbf{skip}$*, the left- and right-translations according to the "coin" program $U[c]$.*

The quantum walks in the above example have only a single walker as well as a single "coin". In the following two examples, we consider some more complicated quantum walks in which multiple walkers participate and multiple "coins" are equipped to control the walkers.

Example 7.2 *A one-dimensional quantum walk driven by multiple coins was defined in [11]. In this walk, there is still a single walker, but it is controlled by M different "coins". Each of these "coins" has its own state space, but the "coin tossing" operator for all of them are the same, namely the 2×2 Hadamard matrix. Now let variable p, space $\mathcal{H}_p$, $\mathcal{H}_c$ and operators T_L, T_R, H are the same as in Example 7.1, and let $c_1, ..., c_M$ be the quantum variables for the M coins. Then the state space of the walk is*

$$\mathcal{H} = \mathcal{H}_p \otimes \bigotimes_{m=1}^{M} \mathcal{H}_{c_m},$$

where $\mathcal{H}_{c_m} = \mathcal{H}_c$ for all $1 \leq m \leq M$. We write

$$W_m = (T_L[p]_{H[c_1]} \oplus T_R[p]); ...; (T_L[p]_{H[c_m]} \oplus T_R[p])$$

for $1 \leq m \leq M$. If we cycle among the M coins, starting from the coin c_1, then the first T steps of the walk can be written in the language QGCL as follows:

$$W_M; ...; W_M; W_r$$

where W_M is iterated for $d = \lfloor T/M \rfloor$ times, and $r = T - Md$ is the remainder of T divided by M. This program is a sequential compositions of T quantum choices of the left- and right translations controlled different "coins".

Example 7.3 *A quantum walk consisting of two walkers on a line sharing coins was introduced in [42]. In this walk, the two walkers have different state spaces, and each of the two walkers has its own "coin". So, the state Hilbert space of the whole quantum walk is $\mathcal{H}_p \otimes \mathcal{H}_p \otimes \mathcal{H}_c \otimes \mathcal{H}_c$. If the two walkers are completely independent, then the step operator of this walk is $W \otimes W$, where W is defined by equation (40). But more interesting is the case where a two-qubit unitary operator U is introduced to entangle the two coins. This case can be thought of as that the two walkers are sharing coins. A step of this quantum walk can be written as a QGCL program as follows:*

$$U[c_1, c_2]; (T_L[q_1]_{H[c_1]} \oplus T_R[q_1]); (T_L[q_2]_{H[c_2]} \oplus T_R[q_2])$$

where q_1, q_2 are the position variables and c_1, c_2 the coin variables of the two walkers, respectively.

8 Conclusions

In this paper, we introduce the notions of quantum alternation and choice by employing the idea of "coin" systems used in the construction of quantum walks. They are quantum counterparts of the popular program construct of alternation, case statement or switch statement in classical programming languages and probabilistic choice in probabilistic programming languages. Based on them, a new quantum programming language, called QGCL, is defined. This language can be seen as a quantum generalization of Dijkstra's language GCL of guarded commands and Morgan et al.'s probabilistic programming language pGCL. It is also an extension of Sanders and Zuliani's quantum programming language qGCL. A salient feature of QGCL that all the previous quantum programming languages do not enjoy is that it can fully support a novel quantum programming paradigm - superposition of programs - which has been implicitly but widely used in the design of quantum walk-based algorithms. We believe that from the programming language point of view, the paradigm of superposition of programs will be a significant step to further exploit the power of quantum computing. This paper presents the denotational and weakest precondition semantics of the language QGCL, and establishes a group of basic algebraic laws that are useful in verification, transformation and compilation of QGCL programs.

We have developed a preliminary theory of quantum programming with quantum alternation and choice, but also leave a series of problems unsolved. Here, we list some of them for the future studies:

- The recursive construct of iteration (or while loop) can be conveniently defined in terms of alternation in classical programming languages. A kind of while loop for quantum programming was considered in [36, 46] based on the classical alternation (5) of quantum programs, and it can be appropriately called *classical controlled quantum loop*. How can we define *quantum controlled loop* - loop based on quantum alternation introduced in this paper? One of the key design ideas of almost all existing quantum programming languages can be summarised by the influential slogan "quantum data, classical control" proposed by Selinger [36], meaning that the control flow of a quantum program is still classical, but the program operates on quantum data. An exception is Altenkirch and Grattage's functional language QML [4], where "quantum control" flow was introduced. It seems that quantum alternation and choice together with quantum controlled loop will provide a much more general structure of control flows for quantum programming.

- A quantum Floyd-Hoare logic was developed in [13, 23, 43] for quantum programs with only classical control flows. So, a further interesting problem would be to extend this logic so that it can also be used to reasoning about programs with quantum control flows.

- Of course, another important problem for further research is the implementation of the new quantum programming language QGCL. It is interesting to notice that recently physicists [49, 6] started to research on the physical implementation of a kind of control of quantum operations, which is similar to the guarded composition of two super-operators considered in Section 3.

References

[1] Abhari, A. J., Faruque, A., Dousti, M. J. et al. 2012. *Scaffold: Quantum Programming Language*. Technical Report-934-12, Department of Computer Science, Princeton University, 2012.

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

[3] Aharonov, Y., Anandan, J., Popescu, S. and Vaidman, L. 1990. Superpositions of time evolutions of a quantum system and quantum time-translation machine. *Plysical Review Letters 64*, 2965–2968.

[4] Altenkirch, T. and Grattage, J. 2005. A functional quantum programming language. In *Proc. of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS)*. 249–258.

[5] Ambainis, A., Bach, E., Nayak, A., Vishwanath, A. and Watrous, J. 2001. One-dimensional quantum walks. In *Proc. of the 33rd ACM Symposium on Theory of Computing (STOC)*. 37–49.

[6] Araújo, M., Feix, A., Costa, F. and Brukner, C. 2013. *Quantum circuits cannot control unknown operations*. arXiv:1309.7976 [quant-ph].

[7] Barman, S., Bodík, R., Chandra, S., Galenson, J., Kimelman, D., Rodarmor, C. and Tung, N. 2010. Programming with angelic nondeterminism. In *Proc. of the 37th ACM Symposium on Principles of Programming Languages (POPL)*. 339–351.

[8] Baltag, A and Smets, S. 2006. LQP: The dynamic logic of quntum information. *Mathematical Structures in Computer Science 16*, 491–525.

[9] Bernstein, E. and Vazirani, U. 1993. Quantum complexity theory. In *Proc. of the 25th Annual ACM Symposium on Theory of Computing (STOC)*. 11–20.

[10] Birkhoff, G. and von Neumann, J. 1936. The logic of quantum mechanics. *Annals of Mathematics 37*, 823–843.

[11] Brun, T. A., Carteret, H. A. and Ambainis, A. 2003. Quantum walks driven by many coins. *Physical Review A 67*, art. no. 052317.

[12] Brunet, O. and Jorrand, P. 2004. Dynamic quantum logic for quantum progrms. *International Journal of Quantum Information 2*, 45–54.

[13] Chadha, R., Mateus, P. and Sernadas, A. 2006. Reasoning about imperative quntum progrms. *Electronic Notes in Theoretical Computer Science 158*, 19–39.

[14] Childs, A. M. 2009. Universal computation by quantum walk. *Physical Review Letters 102*, art. no. 180501.

[15] Deutsch, D. and Jozsa, R. 1992. Rapid solutions of problems by quantum computation. *Proceedings of the Royal Society of London A439*, 553–558.

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

[17] Dijkstra, E. W. 1975. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM 18*, 453–457.

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

[19] Gay, S. 2006. Quantum programming languages: survey and bibliography. *Mathematical Structures in Computer Science 16*, 581–600.

[20] Green, A. S., LeFanu Lumsdaine, P., Ross, N. J., Selinger, P. and Valiron, B. 2013. Quipper: a scalable quantum programming language. In *Proc. of the 34th ACM Conference on Programming Language Design and Implementation (PLDI)*. 333–342.

[21] Grover, L. K. 1996. A fast quantum mechanical algorithm for database search. In *Proc. of the 28th Annual ACM Symposium on the Theory of Computing (STOC)*. 212–219.

[22] Inui, N., Konno, N. and Segawa, E. 2005. One-dimensional three-state quantum walk. *Physical Review E 72*, art. no. 056112.

[23] Kakutani, Y. 2009. A logic for formal verification of quantum programs. *Proc. of the 13th Asian Computing Science Conference (ASIAN)*. LNCS 5913, 79–93.

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

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

[26] Kurzyński, P. and Wójcik, A. 2013. Quantum walk as a generalized measure device. *Physical Review Letters 110*, art. no. 200404.

[27] Lapets, A., da Silva, M. P., Thome, M., Adler, A., Beal, J. and Rötteler, M. 2013. QuaFL: a typed DSL for quantum programming. In *Proc. of the 1st annual workshop on Functional programming concepts in domain-specific languages (FPCDSL)*. 19–26.

[28] Lovett, N. B., Cooper, S., Everitt, M., Trevers, M. and Kendon, V. 2010. Universal quantum computation using the discrete-time quantum walk. *Physicl Review A 81*, art. no. 042330.

[29] McIver, A. and Morgan, C. 2005. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer, New York.

[30] Montero, M. 2013. Unidirectional quantum walks: Evolution and exit times. *Physical Review A 88*, art. no. 012333.

[31] Morgan, C. 1998. *Programming from Specifications*. Prentice Hall, Hertfordshire.

[32] Nagarajan, R., Papanikolaou, N. and Williams, D. 2007. Simulating and compiling code for the Sequential Quantum Random Access Machine. *Electronic Notes in Theoretical Computer Science 170*, 101–124.

[33] Nielsen, M. A. and Chuang, I. L. 2000. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge.

[34] Ömer, B. 2003. *Structural quantum programming*. Ph.D. Thesis, Technical University of Vienna.

[35] Sanders, J. W. and Zuliani, P. 2000. Quantum programming. In *Proc. of 5th International Conference on Mathematics of Program Construction (MPC)*. LNCS 1837. 88–99.

[36] Selinger, P. 2004. Towards a quantum programming language. *Mathematical Structures in Computer Science 14*, 527–586.

[37] Sethi, R. 2002. *Programming Languages: Concepts and Constructs*. Addison-Wesley.

[38] Shende, V. V., Bullock, S. S. and Markov, I. L. 2006. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25*, 1000–1010.

[39] Svore, K. M., Aho, . V., Cross, A. W., Chuang, I. L. and Markov, I. L. 2009. A layered software architecture for quntum computing desgin tools. In *IEEE Computer 39*, 74–83.

[40] Tafliovich, A. and Hehner, E. C. R. 2006. Quantum predicative progrmming. In *Proc. of the 8th International Conference on Mathematics of Program Construction (MPC)*. 433–454.

[41] Venegas-Andraca, S. E. 2012. Quantum walks: a comprehensive review. *Quantum Information Processing 11*, 1015–1106.

[42] Xue, P. and Sanders, B. C. 2012. Two quantum walkers sharing coins. *Physical Review A 85*, art. no. 022307.

[43] Ying, M. S. 2011. Floyd-Hoare logic for quantum programs. *ACM Transactions on Programming Languages and Systems 39*, art. no. 19.

[44] Ying, M. S. 2010. Foundations of quantum programming (Invited talk). In *Proc. of the 8th Asian Symposium on Programming Lnguages and Systems (APLAS)*. LNCS 6461, 16–20.

[45] Ying, M. S., Duan, R. Y., Feng, Y. and Ji, Z. F. 2010. Predicate transformer semantics of quantum progrms. *Semantic Techniques in Quntum Computaton* I. Mackie and S. Gay, eds., Cambridge University Press, 311–360.

[46] Ying, M. S. and Feng, Y. 2010. Quantum loop programs. *Acta Informatica 47*, 221–250.

[47] Ying, M. S. and Feng, Y. 2011. A flowchart language for quantum programming. *IEEE Transactions on Software Engineering 37*, 466–485.

[48] Ying, M. S., Feng, Y., Duan, R. Y., Li, Y. J. and Yu, N. K. 2012. Quantum programming: From theories to implementations. *Chinese Science Bulletin 57*, 1903–1909.

[49] Zhou, X. Q., Ralph, T. C., Kalasuwan, P., Zhang, M., Peruzzo, A., Lanyon, B. P. and O'Brien, J. L. 2011. Adding control to arbitrary unkown quantum operations. *Nature Communications 2*, 413.1–8.

[50] Zuliani, P. 2001. *Quantum Programming*. D.Phil. Thesis, University of Oxford, 2001.

A Choice of the Coefficients in Guarded Compositions of Quantum Operations

The coefficients in the right-hand side of the defining equation (24) of guarded composition of operator-valued function are chosen in a very special way with a physical interpretation in terms of conditional probability. This Appendix shows that other choices of these coefficients are possible. Let's first consider the guarded composition

$$U \stackrel{\triangle}{=} \square_{k=1}^{n} |k\rangle \rightarrow U_k$$

of unitary operators U_k $(1 \leq k \leq n)$ in a Hilbert space $\mathcal{H}$ along an orthonormal basis $\{|k\rangle\}$ of a "coin" Hilbert space $\mathcal{H}_C$. If for each $1 \leq k \leq n$, we add a relative phase θ_k into the defining equation (20) of U:

$$U(|k\rangle|\psi\rangle) = e^{i\theta_k}|k\rangle(U_k|\psi\rangle) \tag{41}$$

for all $|\psi\rangle \in \mathcal{H}$, then equation (21) is changed to

$$U\left(\sum_{k,j} \alpha_{kj}|k\rangle|\psi_j\rangle\right) = \sum_{k,j} \alpha_{kj} e^{i\theta_k}|k\rangle(U_k|\psi_j\rangle). \tag{42}$$

It is easy to see that the new operator U defined by equation (41) or (42) is still unitary.

The idea of adding relative phases also applies to the guarded composition of operator-valued functions. Consider

$$F \stackrel{\triangle}{=} \square_{k=1}^{n} |k\rangle \rightarrow F_k$$

where $\{|k\rangle\}$ is an orthonormal basis of $\mathcal{H}_C$, and F_k is an operator-valued function in $\mathcal{H}$ over Δ_k for every $1 \leq k \leq n$. We arbitrarily choose a sequence $\theta_1, ..., \theta_n$ of real numbers and change the defining equation (24) of F to

$$F(\oplus_{k=1}^{n}\delta_k)|\Psi\rangle = \sum_{k=1}^{n} e^{i\theta_k} \left(\prod_{l\neq k} \lambda_{l\delta_l}\right) |k\rangle(F_k(\delta_k)|\psi_k\rangle \tag{43}$$

for any $|\Psi\rangle = \sum_{k=1}^{n} |k\rangle|\psi_k\rangle \in \mathcal{H}_C \otimes \mathcal{H}$, where $\lambda_{l\delta_l}$'s are the same as in Definition 3.4. Then it is clear that F defined by equation (43) is still an operator-valued function. Indeed, this conclusion is true for a much more general guarded composition of operator-valued functions. Let F_k be an operator-valued function in $\mathcal{H}$ over Δ_k for each $1 \leq k \leq n$, and let

$$\overline{\alpha} = \left\{\alpha^{(k)}_{\delta_1,...,\delta_{k-1},\delta_{k+1},...,\delta_n} : 1 \leq k \leq n \text{ and } \delta_l \in \Delta_l \text{ for } l = 1, ..., k-1, k+1, ..., n\right\} \tag{44}$$

be a family of complex numbers satisfying the normalization condition:

$$\sum_{\delta_1\in\Delta_1,...,\delta_{k-1}\in\Delta_{k-1},\delta_{k+1}\in\Delta_{k+1},...,\delta_n\in\Delta_n} \left|\alpha^{(k)}_{\delta_1,...,\delta_{k-1},\delta_{k+1},...,\delta_n}\right|^2 = 1 \tag{45}$$

for every $1 \leq k \leq n$. Then we can define the $\overline{\alpha}-$guarded composition

$$F \stackrel{\triangle}{=} (\overline{\alpha}) \left(\square_{k=1}^{n} |i\rangle \rightarrow F_k\right)$$

of F_k $(1 \leq k \leq n)$ along an orthonormal basis $\{|k\rangle\}$ of $\mathcal{H}_C$ by

$$F\left(\oplus_{k=1}^{n} \delta_k\right) \left(\sum_{k=1}^{n} |k\rangle|\psi_k\rangle\right) = \sum_{k=1}^{n} \alpha^{(k)}_{\delta_1,\ldots,\delta_{k-1},\delta_{k+1},\ldots,\delta_n} |k\rangle \left(F_k(\delta_k)|\psi_k\rangle\right) \tag{46}$$

for any $|\psi_1\rangle, ..., |\psi_n\rangle \in \mathcal{H}$ and for any $\delta_k \in \Delta_k$ $(1 \leq k \leq n)$. Note that coefficient $\alpha^{(k)}_{\delta_1,\ldots,\delta_{k-1},\delta_{k+1},\ldots,\delta_n}$ does not contain parameter δ_k. This independence together with condition (45) guarantees that the $\overline{\alpha}-$guarded composition is an operator-valued function, as can be seen from the proof of Lemma 3.2 presented in Appendix C.1.

Example A.1 *1. Definition 3.4 is a special case of $\overline{\alpha}-$guarded composition because if for any $1 \leq i \leq n$ and $\delta_k \in \Delta_k$ $(k = 1, ..., i-1, i+1, ..., n)$, we set*

$$\alpha^{i}_{\delta_1,\ldots,\delta_{i-1},\delta_{i+1},\ldots,\delta_n} = \prod_{k \neq i} \lambda_{k\delta_k},$$

where $\lambda_{k\delta_k}$'s are given by equation (25), then equation (46) degenerates to (24).

2. Another possible choice of $\overline{\alpha}$ is

$$\alpha^{i}_{\delta_1,\ldots,\delta_{i-1},\delta_{i+1},\ldots,\delta_n} = \frac{1}{\sqrt{\prod_{k \neq i} |\Delta_k|}}$$

for all $1 \leq i \leq n$ and $\delta_k \in \Delta_k$ $(k = 1, ..., i-1, i+1, ..., n)$. Obviously, for this family $\overline{\alpha}$ of coefficients, the $\overline{\alpha}-$guarded composition cannot be obtained by modifying Definition 3.4 with relative phases.

Now we are able to define parameterized quantum alternation and choice, which are needed in the presentation of some algebraic laws in Section 6.

Definition A.1 *1. Let $\overline{q}$, $\{|i\rangle\}$ and $\{P_i\}$ be as in Definition 2.1 (4). Furthermore, let the classical states $\Delta(P_i) = \Delta_i$ for every i, and let $\overline{\alpha}$ be a family of parameters satisfying condition (45), as in equation (44). Then the $\overline{\alpha}-$quantum alternation of $P_1, ..., P_n$ guarded by basis states $|i\rangle$'s is*

$$P \stackrel{\triangle}{=} \mathbf{qif}\ (\overline{\alpha})[\overline{q}]\ (\square i \cdot\ |i\rangle \rightarrow P_i)\ \mathbf{fiq} \tag{47}$$

and its semi-classical semantics is

$$\lceil P \rceil = (\overline{\alpha})(\square_{i=1}^{n} |i\rangle \rightarrow \lceil P_i \rceil).$$

2. *Let P, $\{|i\rangle\}$ and P_i's be as in Definition 5.1, and let $\overline{\alpha}$ be as above. Then the $\overline{\alpha}-$quantum choice of P_i's according to P along the basis $\{|i\rangle\}$ is defined as*

$$[P(\overline{\alpha})]\left(\bigoplus_i |i\rangle \to P_i\right) \stackrel{\triangle}{=} P; \mathbf{qif}\ (\overline{\alpha})[\overline{q}]\,(\square i \cdot |i\rangle \to P_i)\ \mathbf{end}.$$

The symbol $[\overline{q}]$ in quantum alternation (47) can be dropped whenever quantum variables $\overline{q}$ can be recognized from the context. At the first glance, it seems unreasonable that the parameters $\overline{\alpha}$ in the syntax (47) of $\overline{\alpha}-$quantum alternation are indexed by the classical states of P_i. But this is not problematic at all because the classical states of P_i are completely determined by the syntax of P_i. The purely quantum denotational semantics of the $\overline{\alpha}-$quantum alternation can be obtained from its semi-classical semantics according to Definition 4.3, and the semantics of $\overline{\alpha}-$quantum choice can be derived from the semantics of $\overline{\alpha}-$quantum alternation.

B Quantum Alternation Guarded by Subspaces

A major difference between alternation (2) of classical programs and quantum alternation (18) can be revealed by a comparison between their guards: the guards G_i in the former are propositions about the program variables, whereas the guards $|i\rangle$ in the latter are basis states of the "coin" space $\mathcal{H}_C$. However, this difference is not as big as we imagine at the first glance. In the Birkhoff-von Neumann quantum logic [10], a proposition about a quantum system is expressed by a closed subspace of the state Hilbert space of the system. This observation leads us to a way to define quantum alternation guarded by propositions about the "coin" system instead of basis states of the "coin" space.

Definition B.1 *Let $\overline{q}$ be a sequence of quantum variables and $\{P_i\}$ be a family of programs such that*

$$\overline{q} \cap \left(\bigcup_i qVar(P_i)\right) = \emptyset.$$

Suppose that $\{X_i\}$ is a family of propositions about the "coin" system $\overline{q}$, i.e. closed subspaces of the "coin" space $\mathcal{H}_{\overline{q}}$, satisfying the following two conditions:

1. *X_i's are pairwise orthogonal, i.e. $X_{i_1} \perp X_{i_2}$ provided $i_1 \neq i_2$;*
2. *$\bigoplus_i X_i \stackrel{\triangle}{=} span\,(\bigcup_i X_i) = \mathcal{H}_{\overline{q}}$.*

Then

1. *The quantum alternation of P_i's guarded by subspaces X_i's:*

$$P \stackrel{\triangle}{=} \mathbf{qif}\ [\overline{q}]\,(\square i \cdot X_i \to P_i)\ \mathbf{fiq} \tag{48}$$

is a program.

2. *The quantum variables of the alternation are:*

$$qVar(P) = \overline{q} \cup \left(\bigcup_i qVar(P_i) \right).$$

3. *The purely quantum denotational semantics of the alternation is:*

$$[\![P]\!] = \{[\![\mathbf{qif}\ [\overline{q}]\ (\square i, j_i \cdot |\varphi_{ij_i}\rangle \rightarrow P_{ij_i})\ \mathbf{fiq}]\!] : \{|\varphi_{ij_i}\rangle\} \text{ is an orthonormal basis of } X_i \text{ for each } i, \text{ and } P_{ij_i} = P_i \text{ for every } i, j_i\}. \tag{49}$$

For simplicity, the variables $\overline{q}$ in quantum alternation (48) can be dropped if they can be recognized from or irrelevant in the context. It is clear that the union $\bigcup_i \{|\varphi_{ij_i}\rangle\}$ of the bases of subspaces X_i's in equation (49) is an orthonormal basis of the whole "coin" space $\mathcal{H}_C$. Note that the purely quantum semantics of alternation (48) guarded by subspaces is a set of super-operators rather than a single super-operator. So, alternation (48) is a nondeterministic program, and its nondeterminism comes from different choices of the bases of guard subspaces. Furthermore, an alternation guarded by basis states of these subspaces is a refinement of alternation (48).

The notion of program equivalence in Definition 4.4 can be easily generalized to the case of nondeterministic programs provided we make the following conventions:

- If Ω is a set of super-operators and $\mathcal{F}$ a super-operator, then

$$\Omega \otimes \mathcal{F} = \{\mathcal{E} \otimes \mathcal{F} : \mathcal{E} \in \Omega\};$$

- We identify a single super-operator with the set containing only this super-operator.

Some basic properties of quantum alternation guarded by subspaces are given in the following:

Proposition B.1 *1. If P_i does not contain any measurement for all i, then for any orthonormal basis $\{|\varphi_{ij_i}\rangle\}$ of X_i $(1 \leq i \leq n)$, we have:*

$$\mathbf{qif}\ (\square i \cdot X_i \rightarrow P_i)\ \mathbf{fiq} \equiv \mathbf{qif}\ (\square i, j_i \cdot |\varphi_{ij_i}\rangle \rightarrow P_{ij_i})\ \mathbf{fiq}$$

where $P_{ij_i} = P_i$ for every i, j_i. In particular, if U_i is an unitary operator in $\mathcal{H}_{\overline{q}}$ for all i, then

$$\mathbf{qif}\ [\overline{q}_C](\square i \cdot X_i \rightarrow U_i[\overline{q}])\ \mathbf{fiq} \equiv U[\overline{q}_C, \overline{q}]$$

where $U = \sum_i (I_{X_i} \otimes U_i)$ is an unitary operator in $\mathcal{H}_{\overline{q}_C \cup \overline{q}}$.

2. *Let U be a unitary operator in $\mathcal{H}_{\overline{q}}$. If for every i, X_i is an invariant subspace of U, i.e. $UX_i = \{U|\psi\rangle : |\psi\rangle \in X_i\} \subseteq X_i$, then*

$$U[\overline{q}]; \mathbf{qif}\ [\overline{q}]\ (\square i \cdot X_i \rightarrow P_i)\ \mathbf{fiq}; U^\dagger[\overline{q}] \equiv \mathbf{qif}\ [\overline{q}]\ (\square i \cdot X_i \rightarrow P_i)\ \mathbf{fiq}.$$

C Proofs of Lemmas, Propositions and Theorems

C.1 Proof of Lemma 3.2

Clause (2) can be proved by a routine calculation, which is omitted here. To prove clause (1), we write:

$$\overline{F} \stackrel{\triangle}{=} \sum_{\delta_1 \in \Delta_1, \ldots, \delta_n \in \Delta_n} F(\oplus_{i=1}^n \delta_i)^\dagger \cdot F(\oplus_{i=1}^n \delta_i).$$

Our purpose is to show that $\overline{F} \sqsubseteq I_{\mathcal{H}_C \otimes \mathcal{H}}$, and $\overline{F} = I_{\mathcal{H}_C \otimes \mathcal{H}}$ whenever all F_i's are full. To do this, we start with an auxiliary equality. For any $|\Phi\rangle, |\Psi\rangle \in \mathcal{H}_C \otimes \mathcal{H}$, we can write:

$$|\Phi\rangle = \sum_{i=1}^n |i\rangle|\varphi_i\rangle, \qquad |\Psi\rangle = \sum_{i=1}^n |i\rangle|\psi_i\rangle$$

where $|\varphi_i\rangle, |\psi_i\rangle \in \mathcal{H}$ for each $1 \leq i \leq n$. Then we have:

$$\begin{aligned}
\langle \Phi|\overline{F}|\Psi\rangle &= \sum_{\delta_1,\ldots,\delta_n} \langle \Phi|F(\oplus_{i=1}^n \delta_i)^\dagger \cdot F(\oplus_{i=1}^n \delta_i)|\Psi\rangle \\
&= \sum_{\delta_1,\ldots,\delta_n} \sum_{i,i'=1}^n \left(\prod_{k\neq i} \lambda^*_{k\delta_k}\right)\left(\prod_{k\neq i'} \lambda_{k\delta_k}\right) \langle i|i'\rangle \langle \varphi_i|F_i(\delta_i)^\dagger F_{i'}(\delta_{i'})|\psi_{i'}\rangle \\
&= \sum_{\delta_1,\ldots,\delta_n} \sum_{i=1}^n \left(\prod_{k\neq i} |\lambda_{k\delta_k}|^2\right) \langle \varphi_i|F_i(\delta_i)^\dagger F_i(\delta_i)|\psi_i\rangle \\
&= \sum_{i=1}^n \left[\sum_{\delta_1,\ldots,\delta_{i-1},\delta_{i+1},\ldots,\delta_n} \left(\prod_{k\neq i} |\lambda_{k\delta_k}|^2\right) \cdot \sum_{\delta_i} \langle \varphi_i|F_i(\delta_i)^\dagger F_i(\delta_i)|\psi_i\rangle\right] \\
&= \sum_{i=1}^n \sum_{\delta_i} \langle \varphi_i|F_i(\delta_i)^\dagger F_i(\delta_i)|\psi_i\rangle \\
&= \sum_{i=1}^n \langle \varphi_i| \sum_{\delta_i} F_i(\delta_i)^\dagger F_i(\delta_i)|\psi_i\rangle
\end{aligned} \tag{50}$$

because for each k, we have:

$$\sum_{\delta_k} |\lambda_{k\delta_k}|^2 = 1,$$

and thus

$$\sum_{\delta_1,\ldots,\delta_{i-1},\delta_{i+1},\ldots,\delta_n} \left(\prod_{k\neq i} |\lambda_{k\delta_k}|^2\right) = \prod_{k\neq i} \left(\sum_{\delta_k} |\lambda_{k\delta_k}|^2\right) = 1. \tag{51}$$

Now we are ready to prove our conclusions by using equation (50).

(1) We first prove that $\overline{F} \sqsubseteq I_{\mathcal{H}_C \otimes \mathcal{H}}$, i.e. F is an operator-valued function in $\mathcal{H}_C \otimes \mathcal{H}$ over $\bigoplus_{i=1}^{n} \Delta_n$. It suffices to show that $\langle \Phi | \overline{F} | \Phi \rangle \leq \langle \Phi | \Phi \rangle$ for each $|\Phi\rangle \in \mathcal{H}_C \otimes \mathcal{H}$. In fact, for each $1 \leq i \leq n$, since F_i is an operator-valued function, we have:

$$\sum_{\delta_i} F_i(\delta_i)^\dagger F_i(\delta_i) \sqsubseteq I_{\mathcal{H}}.$$

Therefore, it holds that

$$\langle \varphi_i | \sum_{\delta_i} F_i(\delta_i)^\dagger F_i(\delta_i) | \varphi_i \rangle \leq \langle \varphi_i | \varphi_i \rangle.$$

Then it follows immediately from equation (50) that

$$\langle \Phi | \overline{F} | \Phi \rangle \leq \sum_{i=1}^{n} \langle \varphi_i | \varphi_i \rangle = \langle \Phi | \Phi \rangle.$$

So, F is an operator-valued function.

(2) Secondly, we prove that F is full for the case where all F_i $(1 \leq i \leq n)$ are full. It requires us to show that $\overline{F} = I_{\mathcal{H}_C \otimes \mathcal{H}}$. In fact, for every $1 \leq i \leq n$, we have:

$$\sum_{\delta_i} F_i(\delta_i)^\dagger F_i(\delta_i) = I_{\mathcal{H}}$$

because F_i is full. Thus, it follows from equation (50) that for any $|\Phi\rangle, |\Psi\rangle \in \mathcal{H}_C \otimes \mathcal{H}$,

$$\langle \Phi | \overline{F} | \Psi \rangle = \sum_{i=1}^{n} \langle \varphi_i | \psi_i \rangle = \langle \Phi | \Psi \rangle.$$

So, it holds that $\overline{F} = I_{\mathcal{H} \otimes \mathcal{H}_s}$ by arbitrariness of $|\Phi\rangle$ and $\Psi\rangle$, and F is full.

C.2 Proof of Proposition 4.1

Clauses (1) - (4) are obvious. To prove clause (5), let

$$P \stackrel{\triangle}{=} \mathbf{measure}\ (\square m \cdot M[\overline{q} : x] = m \to P_m)\ \mathbf{end}.$$

Then by Definitions 4.2 and 4.3, for any partial density operator ρ in $\mathcal{H}_{qvar(P)}$, we have:

$$\begin{aligned}
[\![P]\!](\rho) &= \sum_m \sum_{\delta\in\Delta(P_m)} \lceil P\rceil(\delta[x\leftarrow m])\rho\lceil P\rceil(\delta[x\leftarrow m])^\dagger \\
&= \sum_m \sum_{\delta\in\Delta(P_m)} \left(\lceil P_m\rceil(\delta)\otimes I_{qvar(P)\setminus qvar(P_m)}\right)\left(M_m\otimes I_{qvar(P)\setminus\overline{q}}\right) \\
&\qquad\qquad \rho\left(M_m^\dagger\otimes I_{qvar(P)\setminus\overline{q}}\right)\left(\lceil P_m\rceil(\delta)^\dagger\otimes I_{qvar(P)\setminus qvar(P_m)}\right) \\
&= \sum_m \sum_{\delta\in\Delta(P_m)} \left(\lceil P_m\rceil(\delta)\otimes I_{qvar(P)\setminus qvar(P_m)}\right)\left(M_m\rho M_m^\dagger\right) \\
&\qquad\qquad \left(\lceil P_m\rceil(\delta)^\dagger\otimes I_{qvar(P)\setminus qvar(P_m)}\right) \\
&= \sum_m [\![P_m]\!]\left(M_m\rho M_m^\dagger\right) \\
&= \left(\sum_m \left[(M_m\circ M_m^\dagger);[\![P_m]\!]\right]\right)(\rho).
\end{aligned}$$

Finally, we prove clause (6). For simplicity of the presentation, we write:

$$P \stackrel{\triangle}{=} \mathbf{qif}\ [\overline{q}](\square i\cdot|i\rangle\to P_i)\ \mathbf{fiq}.$$

By Definitions 4.2, we obtain:

$$\lceil P\rceil = \square_i|i\rangle\to\lceil P_i\rceil.$$

Note that $\lceil P_i\rceil\in\mathbb{F}([\![P_i]\!])$ for every $1\le i\le n$, where $\mathbb{F}(\cdot)$ is defined as in the paragraph before Definition 3.3. Therefore, it follows from Definition 4.3 that

$$[\![P]\!] = \mathcal{E}(\lceil P\rceil)\in\{\mathcal{E}(\square_i|i\rangle\to F_i) : F_i\in\mathbb{F}([\![P_i]\!]) \text{ for every } i\} = \square_i|i\rangle\to[\![P_i]\!].$$

C.3 Proof of Proposition 4.2

The proof is based on the following key lemma by D'Hondt and Panangaden [16].

Lemma C.1 *If the semantic function $[\![P]\!]$ of program P has the Kraus operator-sum representation:*

$$[\![P]\!] = \sum_j E_j\circ E_j^\dagger,$$

then we have:

$$wp.P = \sum_j E_j^\dagger\circ E_j.$$

Now we start to prove Proposition 4.2. Clauses (1) - (4) are immediate corollaries of Proposition 4.1 and Lemma C.1. Clause (5) can be directly proved by the transformation between the purely quantum semantics of a program and its weakest precondition semantics given by Lemma C.1. We write:

$$P \stackrel{\triangle}{=} \mathbf{measure}\ (\square m \cdot M[\overline{q} : x] = m \to P_m)\ \mathbf{end},$$

and suppose that for every m,

$$[\![P_m]\!] = \sum_m E_{mi_m} \circ E_{mi_m}^\dagger.$$

Then by Proposition 4.1 (5) we have:

$$\begin{aligned}
[\![P]\!] &= \sum_m \Big[(M_m \circ M_m^\dagger); [\![P_m]\!]\Big] \\
&= \sum_m \left[(M_m \circ M_m^\dagger); \sum_{i_m} \Big(E_{mi_m} \circ E_{mi_m}^\dagger\Big)\right] \\
&= \sum_m \sum_{i_m} \Big[(E_{mi_m} M_m) \circ (M_m^\dagger E_{mi_m}^\dagger)\Big] \\
&= \sum_m \sum_{i_m} \Big[(E_{mi_m} M_m) \circ (E_{mi_m} M_m)^\dagger\Big].
\end{aligned}$$

Using Lemma C.1 we obtain:

$$\begin{aligned}
wp.P &= \sum_m \sum_{i_m} \Big[(E_{mi_m} M_m)^\dagger \circ (E_{mi_m} M_m)\Big] \\
&= \sum_m \sum_{i_m} \Big[(M_m^\dagger E_{mi_m}^\dagger) \circ (E_{mi_m} M_m)\Big] \\
&= \sum_m \left[\sum_{i_m} \Big(E_{mi_m}^\dagger \circ E_{mi_m}\Big); \Big(M_m^\dagger \circ M_m\Big)\right] \\
&= \sum_m \Big[wp.P_m; (M_m^\dagger \circ M_m)\Big].
\end{aligned}$$

The proof technique of clause (6) is different from that of clause (5). To prove clause (6), it is enough to consider the purely quantum semantics of the involved programs. Instead, we have to go to the semi-classic semantics. For each $1 \leq i \leq n$, assume that the semi-classical semantics of P_i is the function $\lceil P_i \rceil$ over $\Delta = \{j_i\}$ such that

$$\lceil P_i \rceil (j_i) = E_{ij_i}$$

for every j_i. Then by Definition 4.3 we obtain:

$$[\![P_i]\!] = \sum_{j_i} E_{ij_i} \circ E_{ij_i}^\dagger,$$

and it follows from Lemma C.1 that

$$wp.P_i = \sum_{j_i} E_{ij_i}^\dagger \circ E_{ij_i}.$$

Now we compute the guarded composition of these operator-valued functions. For any state

$$|\varphi\rangle = \sum_{i=1}^{n} |i\rangle|\varphi_i\rangle$$

in $\mathcal{H}_C \otimes \mathcal{H}$, where $|\varphi_i\rangle \in \mathcal{H}_{qvar(P_i)}$ $(1 \leq i \leq n)$, we define:

$$G_{j_1 \ldots j_n}(|\varphi\rangle) = \sum_{i=1}^{n} \zeta_i |i\rangle (E_{ij_i}^\dagger |\varphi_i\rangle),$$
$$\zeta_i = \prod_{k \neq i} \delta_{kj_k}$$

for any $j_1, ..., j_n$ and i, where

$$\delta_{kj_k} = \sqrt{\frac{tr(E_{kj_k}^\dagger)^\dagger E_{kj_k}^\dagger}{\sum_{l_k} (E_{kl_k}^\dagger)^\dagger E_{kl_k}^\dagger}}.$$

It is obvious that

$$\delta_{kj_k} = \sqrt{\frac{tr E_{kj_k}^\dagger E_{kj_k}}{\sum_{l_k} E_{kl_k}^\dagger E_{kl_k}}} = \lambda_{kj_k} \tag{52}$$

and λ_{kj_k}'s are defined by equation (25). By Definitions 3.4 and 3.5 we have:

$$\sum_{j_1,\ldots,j_n} G_{j_1 \ldots j_n} \circ G_{j_1 \ldots j_n}^\dagger \in \square_{i=1}^{n} |i\rangle \to wp.P_i.$$

On the other hand, we write

$$P \stackrel{\triangle}{=} \mathbf{qif}\ (\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq}.$$

Then by Definitions 4.2 (5) and 4.3 we have:

$$[\![P]\!] = \sum_{j_1,\ldots,j_n} F_{j_1 \ldots j_n} \circ F_{j_1 \ldots j_n}^\dagger$$

where $F_{j_1 \ldots j_n}$'s are defined by equation (24). Applying Lemma C.1 once again, we obtain:

$$wp.P = \sum_{j_1,\ldots,j_n} F_{j_1 \ldots j_n}^\dagger \circ F_{j_1 \ldots j_n}.$$

So, we complete the proof of clause (6) if we are able to prove that

$$G_{j_1 \ldots j_n} = F_{j_1 \ldots j_n}^\dagger$$

for all $j_1, ..., j_n$. In fact, we can prove the above equality by a straightforward calculation: for any state

$$|\varphi\rangle = \sum_{i=1}^{n} |i\rangle|\varphi_i\rangle, \qquad |\psi\rangle = \sum_{i=1}^{n} |i\rangle|\psi_i\rangle$$

with $|\varphi\rangle, |\psi_i\rangle \in \mathcal{H}_{qvar(P_i)}$ $(1 \leq i \leq n)$, it holds that

$$\begin{aligned}
(G_{j_1\ldots j_n}|\varphi\rangle, |\psi\rangle) &= \left(\sum_{i=1}^{n} \zeta_i |i\rangle(E_{ij_i}^{\dagger}|\varphi_i\rangle), \sum_{i=1}^{n} |i\rangle|\psi_i\rangle\right) \\
&= \sum_{i,i'} \zeta_i^* \langle i|i'\rangle (E_{ij_i}^{\dagger}|\varphi_i\rangle, |\psi_{i'}\rangle) \\
&= \sum_{i} \zeta_i (E_{ij_i}^{\dagger}|\varphi_i\rangle, |\psi_i\rangle) \\
&= \sum_{i} \zeta_i (|\varphi_i\rangle, E_{ij_i}|\psi_i\rangle) \\
&= \sum_{i,i'} \zeta_i \langle i|i'\rangle (|\varphi_i\rangle, E_{i'j_{i'}}|\psi_{i'}\rangle) \\
&= \left(\sum_{i=1}^{n} |i\rangle|\varphi_i\rangle, \sum_{i=1}^{n} \zeta_i |i\rangle(E_{ij_i}|\psi_i\rangle)\right) \\
&= (|\varphi\rangle, F_{j_1\ldots j_n}|\psi\rangle)
\end{aligned}$$

because ζ_i's are real numbers, and it follows from equation (52) that

$$\zeta_i = \prod_{k \neq i} \lambda_{kj_k}$$

for each $1 \leq i \leq n$. Thus, we complete the proof.

C.4 Proof of Theorem 5.1

To simplify the presentation, we write:

$$R \stackrel{\triangle}{=} \mathbf{qif}\ [\overline{q}](\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq}.$$

We need to work at the level of semi-classical semantics first, and then lift it to the purely quantum semantics. Assume that the semi-classical semantics $\lceil P_i \rceil$ is the operator-valued function over Δ_i such that $\lceil P_i \rceil(\delta_i) = E_{i\delta_i}$ for each $\delta_i \in \Delta_i$. Let states

$$|\psi\rangle \in \mathcal{H}_{\bigcup_{i=1}^{n} qvar(P_i)}$$

and $|\varphi\rangle \in \mathcal{H}_{\overline{q}}$. We can write:

$$|\varphi\rangle = \sum_{i=1}^{n} \alpha_i |i\rangle$$

for some complex numbers α_i $(1 \leq i \leq n)$. Then for any $\delta_i \in \Delta_i$ $(1 \leq i \leq n)$, we have:

$$\begin{aligned}|\Psi_{\delta_1 \ldots \delta_n}\rangle &\stackrel{\triangle}{=} \lceil R \rceil (\oplus_{i=1}^n \delta_i)(|\varphi\rangle|\psi\rangle) \\ &= \lceil R \rceil (\oplus_{i=1}^n \delta_i) \left(\sum_{i=1}^n \alpha_i |i\rangle |\psi\rangle \right) \\ &= \sum_{i=1}^n \alpha_i \left(\prod_{k \neq i} \lambda_{k\delta_k} \right) |i\rangle (E_{i\sigma_i}|\psi\rangle)\end{aligned}$$

where $\lambda_{i\delta_i}$'s are defined as in equation (25). We continue to compute:

$$|\Psi_{\delta_1 \ldots \delta_n}\rangle\langle\Psi_{\delta_1 \ldots \delta_n}| = \sum_{i,j=1}^n \left[\alpha_i \alpha_j^* \left(\prod_{k \neq i} \lambda_{k\delta_k} \right) \left(\prod_{k \neq j} \lambda_{k\delta_k} \right) |i\rangle\langle j| \otimes E_{i\delta_i} |\psi\rangle\langle\psi| E_{j\delta_j}^\dagger \right],$$

and it follows that

$$tr_{\mathcal{H}_{\overline{q}}} |\Psi_{\delta_1 \ldots \delta_n}\rangle\langle\Psi_{\delta_1 \ldots \delta_n}| = \sum_{i=1}^n |\alpha_i|^2 \left(\prod_{k \neq i} \lambda_{k\delta_k} \right)^2 E_{i\delta_i} |\psi\rangle\langle\psi| E_{i\delta_i}^\dagger.$$

Using equation (51), we obtain:

$$\begin{aligned}tr_{\mathcal{H}_{\overline{q}}} [\![R]\!](|\varphi\psi\rangle\langle\psi\varphi|) &= tr_{\mathcal{H}_{\overline{q}}} \left(\sum_{\delta_1, \ldots, \delta_n} |\Psi_{\delta_1 \ldots \delta_n}\rangle\langle\Psi_{\delta_1 \ldots \delta_n}| \right) \\ &= \sum_{\delta_1, \ldots, \delta_n} tr_{\mathcal{H}_{\overline{q}}} |\Psi_{\delta_1 \ldots \delta_n}\rangle\langle\Psi_{\delta_1 \ldots \delta_n}| \\ &= \sum_{i=1}^n |\alpha_i|^2 \left[\sum_{\delta_1, \ldots, \delta_{i-1}, \delta_{i+1}, \ldots, \delta_n} \left(\prod_{k \neq i} \lambda_{k\delta_k} \right)^2 \right] \cdot \left[\sum_{\delta_i} E_{i\delta_i} |\psi\rangle\langle\psi| E_{i\delta_i}^\dagger \right] \\ &= \sum_{i=1}^n |\alpha_i|^2 [\![P_i]\!](|\psi\rangle\langle\psi|).\end{aligned} \tag{53}$$

Now we do spectral decomposition for $[\![P]\!](\rho)$, which is a density operator, and assume that

$$[\![P]\!](\rho) = \sum_l s_l |\varphi_l\rangle\langle\varphi_l|.$$

We further write:

$$|\varphi_l\rangle = \sum_i \alpha_{li} |i\rangle$$

for every l. For any density operator σ in $\mathcal{H}_{\bigcup_{i=1}^n qvar(P_i)}$, we can write σ in the form of

$$\sigma = \sum_m r_m |\psi_m\rangle\langle\psi_m|.$$

Then using equation (53), we get:

$$\begin{aligned}
[\![\mathbf{begin\ local}\ \overline{q} := \rho;[P]\left(\bigoplus_{i=1}^{n}|i\rangle \to P_i\right)\ \mathbf{end}]\!](\sigma) &= tr_{\mathcal{H}_{\overline{q}}}[\![P;R]\!](\sigma\otimes\rho)\\
&= tr_{\mathcal{H}_{\overline{q}}}[\![R]\!](\sigma\otimes[\![P]\!](\rho))\\
&= tr_{\mathcal{H}_{\overline{q}}}[\![R]\!]\left(\sum_{m,l} r_m s_l |\psi_m\varphi_l\rangle\langle\varphi_l\psi_m|\right)\\
&= \sum_{m,l} r_m s_l tr_{\mathcal{H}_{\overline{q}}}[\![R]\!](|\psi_m\varphi_l\rangle\langle\varphi_l\psi_m|)\\
&= \sum_{m,l} r_m s_l \sum_{i=1}^{n} |\alpha_{li}|^2 [\![P_i]\!](|\psi_m\rangle\langle\psi_m|)\\
&= \sum_{l}\sum_{i=1}^{n} s_l|\alpha_{li}|^2 [\![P_i]\!]\left(\sum_m r_m|\psi_m\rangle\langle\psi_m|\right)\\
&= \sum_{l}\sum_{i=1}^{n} s_l|\alpha_{li}|^2 [\![P_i]\!](\sigma)\\
&= \sum_{i=1}^{n}\left(\sum_l s_l|\alpha_{li}|^2\right)[\![P_i]\!](\sigma)\\
&= \left[\!\!\left[\sum_{i=1}^{n} P_i@p_i\right]\!\!\right](\sigma),
\end{aligned}$$

where

$$p_i = \sum_l s_l|\alpha_{li}|^2 = \sum_l s_l\langle i|\varphi_l\rangle\langle\varphi_l|i\rangle = \langle i|\left(\sum_l s_l|\varphi_l\rangle\langle\varphi_l|\right)|i\rangle = \langle i|[\![P]\!](\rho)|i\rangle.$$

C.5 Proof of Theorem 6.2

We first prove equation (38). Let LHS and RHS stand for the left and right hand side of equation (38), respectively. What we want to prove is $[\![LHS]\!] = [\![RHS]\!]$. But we need to work with the semi-classical semantics, and show that $\lceil LHS\rceil = \lceil RHS\rceil$. Assume that $\lceil P_i\rceil$ is the operator-valued function over Δ_i such that

$$\lceil P_i\rceil(\delta_i) = F_{i\delta_i}$$

for each $\delta_i \in \Delta_i$ $(1 \le i \le n)$. We write:

$$P \stackrel{\triangle}{=} \mathbf{qif}\ (\square i \cdot U_{\overline{q}}^{\dagger}|i\rangle \to P_i)\ \mathbf{fiq}.$$

Then for any state

$$|\psi\rangle = \sum_{i=1}^{n} |i\rangle|\psi_i\rangle,$$

where $|\psi_i\rangle \in \mathcal{H}_V$ $(1 \leq i \leq n)$, and $V = \bigcup_{i=1}^{n} qvar(P_i)$, we have:

$$\begin{aligned}
\lceil P \rceil(\oplus_{i=1}^{n}\delta_i)|\psi\rangle &= \lceil P \rceil(\oplus_{i=1}^{n}\delta_i)\left[\sum_{i=1}^{n}\left(\sum_{j=1}^{n} U_{ij}(U_{\overline{q}}^{\dagger}|j\rangle)\right)|\psi_i\rangle\right] \\
&= \lceil P \rceil(\oplus_{i=1}^{n}\delta_i)\left[\sum_{j=1}^{n}(U_{\overline{q}}^{\dagger}|j\rangle)\left(\sum_{i=1}^{n} U_{ij}|\psi_i\rangle\right)\right] \\
&= \sum_{j=1}^{n}\left(\prod_{k\neq j}\lambda_{k\delta_k}\right)(U_{\overline{q}}^{\dagger}|j\rangle)F_{j\delta_j}\left(\sum_{i=1}^{n} U_{ij}|\psi_i\rangle\right),
\end{aligned}$$

where $\lambda_{k\delta_k}$'s are defined by equation (25). Then it holds that

$$\begin{aligned}
\lceil RHS \rceil(\oplus_{i=1}^{n}\delta_i)|\psi\rangle &= U_{\overline{q}}(\lceil P \rceil(\oplus_{i=1}^{n}\delta_i)|\psi\rangle) \\
&= \sum_{j=1}^{n}\left(\prod_{k\neq j}\lambda_{k\delta_k}\right)|j\rangle F_{j\delta_j}\left(\sum_{i=1}^{n} U_{ij}|\psi_i\rangle\right) \\
&= \lceil P \rceil(\oplus_{i=1}^{n}\delta_i)\left[\sum_{j=1}^{n}|j\rangle\left(\sum_{i=1}^{n} U_{ij}|\psi_i\rangle\right)\right] \\
&= \lceil P \rceil(\oplus_{i=1}^{n}\delta_i)\left[\sum_{i=1}^{n}\left(\sum_{j=1}^{n} U_{ij}|j\rangle\right)|\psi_i\rangle\right] \\
&= \lceil P \rceil(\oplus_{i=1}^{n}\delta_i)\left(\sum_{i=1}^{n}(U_{\overline{q}}|i\rangle)|\psi_i\rangle\right) \\
&= \lceil LHS \rceil(\oplus_{i=1}^{n}\delta_i)|\psi\rangle.
\end{aligned}$$

So, we complete the proof of equation (38).

Now we are ready to prove equation (39). The basic idea is to use equation (38) that we just proved above to prove the more general equation (39). So, we need to turn the general "coin" program P into a special "coin" program which is a unitary transformation. The technique that we used before to deal with super-operators is always the Kraus operator-sum representation. Here, however, we have to employ the system-environment model of super-operators (see equation (8.38) in [33]). Since $[\![P]\!]$ is a super-operator in $\mathcal{H}_{\overline{q}}$, there must be a family of quantum variables $\overline{r}$, a pure state $|\varphi_0\rangle \in \mathcal{H}_{\overline{r}}$, a unitary operator U in $\mathcal{H}_{\overline{q}} \otimes \mathcal{H}_{\overline{r}}$, and a projection operator K onto some closed subspace $\mathcal{K}$ of $\mathcal{H}_{\overline{r}}$ such that

$$[\![P]\!](\rho) = tr_{\mathcal{H}_{\overline{r}}}(KU(\rho \otimes |\varphi_0\rangle\langle\varphi_0|)U^{\dagger}K) \tag{54}$$

for all density operators ρ in $\mathcal{H}_{\overline{q}}$. We choose an orthonormal basis of $\mathcal{K}$ and then extend it to an orthonormal basis $\{|j\rangle\}$ of $\mathcal{H}_{\overline{r}}$. Define pure states $|\psi_{ij}\rangle = U^\dagger|ij\rangle$ for all i, j and programs

$$Q_{ij} = \begin{cases} P_i & \text{if } |j\rangle \in \mathcal{K}, \\ \mathbf{abort} & \text{if } |j\rangle \notin \mathcal{K}. \end{cases}$$

Then by a routine calculation we have:

$$[\![\mathbf{qif}\ (\square i, j \cdot |ij\rangle \to Q_{ij})\ \mathbf{fiq}]\!](\sigma) = [\![\mathbf{qif}\ (\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq}]\!](K\sigma K) \tag{55}$$

for any $\sigma \in \mathcal{H}_{\overline{q}\cup\overline{r}\cup V}$, where $V = \bigcup_{i=1}^{n} qvar(P_i)$. We now write RHS for the right hand side of equation (39). Then we have:

$$\begin{aligned}
[\![RHS]\!](\rho) &= tr_{\mathcal{H}_{\overline{r}}}\Big([\![\mathbf{qif}\ (\square i, j \cdot U^\dagger|ij\rangle \to Q_{ij})\ \mathbf{fiq}; U[\overline{q}, \overline{r}]]\!](\rho \otimes |\varphi_0\rangle\langle\varphi_0|\Big) \\
&= tr_{\mathcal{H}_{\overline{r}}}\left(\left[\!\!\left[[U[\overline{q}, \overline{r}]]\left(\bigoplus_{i,j} |ij\rangle \to Q_{ij}\right)\right]\!\!\right](\rho \otimes |\varphi_0\rangle\langle\varphi_0|)\right) \\
&= tr_{\mathcal{H}_{\overline{r}}}\Big([\![\mathbf{qif}\ (\square i, j \cdot |ij\rangle \to Q_{ij})\ \mathbf{fiq}]\!](U(\rho \otimes |\varphi_0\rangle\langle\varphi_0|)U^\dagger)\Big) \\
&= tr_{\mathcal{H}_{\overline{r}}}[\![\mathbf{qif}\ (\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq}]\!](KU(\rho \otimes |\varphi_0\rangle\langle\varphi_0|)U^\dagger K) \\
&= [\![\mathbf{qif}\ (\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq}]\!](tr_{\mathcal{H}_{\overline{r}}}(KU(\rho \otimes |\varphi_0\rangle\langle\varphi_0|)U^\dagger K)) \\
&= [\![\mathbf{qif}\ (\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq}]\!]([\![P]\!](\rho)) \\
&= \left[\!\!\left[[P]\left(\bigoplus_i |i\rangle \to P_i\right)\right]\!\!\right](\rho)
\end{aligned}$$

for all density operators ρ in $\mathcal{H}_{\overline{q}}$. Here, the second equality is obtained by using equation (38), the fourth equality comes from (55), the fifth equality holds because $\overline{r} \cap qvar(\mathbf{qif}\ (\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq}) = \emptyset$, and the sixth equality follows from equation (54). Therefore, equation (39) is proved.

C.6 Proof of Theorem 6.1 and Theorem 6.3

The proof of Theorem 6.1 is similar to but simpler than the proof of Theorem 6.3. So, here we only prove Theorem 6.3.

(1) Clause (1) is immediate from Theorem 5.1.

(2) To prove clause (2), we write:

$$Q \stackrel{\triangle}{=} \mathbf{qif}\ [\overline{q}](\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq},$$
$$R \stackrel{\triangle}{=} \mathbf{qif}\ [\overline{q}](\square i \cdot |i\rangle \to P_{\tau(i)})\ \mathbf{fiq}.$$

By definition, we have $LHS = P; R$ and $RHS = P; U_\tau[\overline{q}]; Q; U_{\tau^{-1}}[\overline{q}]$. So, it suffices to show that $R \equiv U_\tau[\overline{q}]; Q; U_{\tau^{-1}}[\overline{q}]$. Again, we first need to deal with the semi-classical semantics of the two sides of this equality. Assume that $\lceil P_i \rceil$ is the operator-valued function

over Δ_i with $\lceil P_i \rceil(\delta_i) = E_{i\delta_i}$ for each $\delta_i \in \Delta_i$ $(1 \leq i \leq n)$. For each state $|\Psi\rangle \in \mathcal{H}_{\overline{q} \cup \bigcup_{i=1}^n qvar(P_i)}$, we can write:

$$|\Psi\rangle = \sum_{i=1}^{n} |i\rangle |\psi_i\rangle$$

for some $|\psi_i\rangle \in \mathcal{H}_{\bigcup_{i=1}^n qvar(P_i)}$ $(1 \leq i \leq n)$. Then for any $\delta_1 \in \Delta_{\tau(1)}, ..., \delta_n \in \Delta_{\tau(n)}$, it holds that

$$\begin{aligned} |\Psi_{\delta_1 \ldots \delta_n}\rangle &\stackrel{\triangle}{=} \lceil R \rceil (\oplus_{i=1}^n \delta_i)(|\Psi\rangle) \\ &= \sum_{i=1}^{n} \left(\prod_{k \neq i} \mu_{k\delta_k} \right) |i\rangle (E_{\tau(i)\delta_i} |\psi_i\rangle), \end{aligned}$$

where

$$\mu_{k\delta_k} = \sqrt{\frac{tr E^{\dagger}_{\tau(k)\delta_k} E_{\tau(k)\delta_k}}{\sum_{\theta_k \in \Sigma_{\tau(k)}} tr E^{\dagger}_{\tau(k)\theta_k} E_{\tau(k)\theta_k}}} = \lambda_{\tau(k)\delta_k} \tag{56}$$

for every k and δ_k, and $\lambda_{i\sigma_i}$'s are defined by equation (25). On the other hand, we first observe:

$$|\Psi'\rangle \stackrel{\triangle}{=} (U_\tau)_{\overline{q}}(|\Psi\rangle) = \sum_{i=1}^{n} |\psi_i\rangle |\tau(i)\rangle = \sum_{j=1}^{n} |\psi_{\tau^{-1}(j)}\rangle |j\rangle.$$

Then for any $\delta_1 \in \Delta_1, ..., \delta_n \in \Delta_n$, it holds that

$$\begin{aligned} |\Psi''_{\delta_1 \ldots \delta_n}\rangle &\stackrel{\triangle}{=} \lceil Q \rceil \left(\bigoplus_{i=1}^{n} \delta_i \right) \left(|\Psi'\rangle \right) \\ &= \sum_{j=1}^{n} \left(\prod_{l \neq j} \lambda_{l\delta_{\tau^{-1}(l)}} \right) |j\rangle (E_{j\delta_{\tau^{-1}(j)}} |\psi_{\tau^{-1}(j)}\rangle) \\ &= \sum_{i=1}^{n} \left(\prod_{k \neq i} \lambda_{\tau(k)\delta_k} \right) |\tau(i)\rangle (E_{\tau(i)\delta_i} |\psi_i\rangle). \end{aligned}$$

Furthermore, we have:

$$(U_{\tau^{-1}})_{\overline{q}}(|\Psi''_{\delta_1 \ldots \delta_n}\rangle) = \sum_{i=1}^{n} \left(\prod_{k \neq i} \lambda_{\tau(k)\delta_k} \right) |i\rangle (E_{\tau(i)\delta_i} |\psi_i\rangle).$$

Therefore, we can compute the purely quantum semantics:

$$\begin{aligned} [\![U_\tau[\overline{q}]; Q; U_{\tau^{-1}}[\overline{q}]]\!](|\Psi\rangle\langle\Psi|) &= [\![Q; U_{\tau^{-1}}[\overline{q}](|\Psi'\rangle\langle\Psi'|)]\!] \\ &= (U_{\tau^{-1}})_{\overline{q}} \left(\sum_{\delta_1, \ldots, \delta_n} (|\Psi''_{\delta_1 \ldots \delta_n}\rangle\langle\Psi''_{\delta_1 \ldots \delta_n}|) \right) (U_\tau)_{\overline{q}} \\ &= \sum_{\delta_1, \ldots, \delta_n} |\Psi_{\delta_1 \ldots \delta_n}\rangle\langle\Psi_{\delta_1 \ldots \delta_n}| \\ &= [\![R]\!](\Psi\rangle\langle\Psi|). \end{aligned} \tag{57}$$

Here, the second equality comes from equation (56) and the fact that τ is one-onto-one, and thus $\tau^{-1}(j)$ traverses over $1, ..., n$ as j does. Thus, it follows from equation (57) and spectral decomposition that

$$[\![R]\!](\rho) = [\![U_\tau[\overline{q}]; Q; U_{\tau^{-1}}[\overline{q}]]\!](\rho)$$

for any density operator ρ in $\mathcal{H}_{\overline{q} \cup \bigcup_{i=1}^{n} qvar(P_i)}$, and we complete the proof of clause (2).

(3) To prove clause (3), we write:

$$X_i \stackrel{\triangle}{=} \mathbf{qif}\ (\square j_i \cdot |j_i\rangle \to R_{ij_i})\ \mathbf{fiq},$$
$$Y_i \stackrel{\triangle}{=} [Q_i] \left(\bigoplus_{j_i=1}^{n_i} |j_i\rangle \to R_{ij_i} \right)$$

for every $1 \leq i \leq m$, and we further put:

$$X \stackrel{\triangle}{=} \mathbf{qif}\ (\square i \cdot |i\rangle \to Y_i)\ \mathbf{fiq},$$
$$T \stackrel{\triangle}{=} \mathbf{qif}\ (\square i \cdot |i\rangle \to Q_i)\ \mathbf{fiq},$$
$$Z \stackrel{\triangle}{=} \mathbf{qif}\ (\overline{\alpha})(\square i, j_i \in \Delta \cdot |i, j_i\rangle \to R_{ij_i})\ \mathbf{fiq}.$$

Then by the definition of quantum choice we have $LHS = P; X$ and $RHS = P; T; Z$. So, it suffices to show that $X \equiv T; Z$. To do this, we consider the semi-classical semantics of the involved programs. For each $1 \leq i \leq m$, and for each $1 \leq j_i \leq n_i$, we assume:

- $\lceil Q_i \rceil$ is the operator-valued function over Δ_i such that $\lceil Q_i \rceil(\delta_i) = F_{i\delta_i}$ for every $\delta_i \in \Delta_i$; and
- $\lceil R_{ij_i} \rceil$ is the operator-valued function over Σ_{ij_i} such that $\lceil R_{ij_i} \rceil(\sigma_{ij_i}) = E_{(ij_i)\sigma_{ij_i}}$ for every $\sigma_{ij_i} \in \Sigma_{ij_i}$.

We also assume that state

$$|\Psi\rangle = \sum_{i=1}^{m} |i\rangle |\Psi_i\rangle$$

where each $|\Psi_i\rangle$ is further decomposed into

$$|\Psi_i\rangle = \sum_{j_i=1}^{n_i} |j_i\rangle |\psi_{ij_i}\rangle$$

with $|\psi_{ij_i}\rangle \in \mathcal{H}_{\bigcup_{j_i=1}^{n_i} qvar(R_{ij_i})}$ for every $1 \leq i \leq m$ and $1 \leq j_i \leq n_i$. To simplify the presentation, we use the abbreviation $\overline{\sigma}_i = \oplus_{j_i=1}^{n_i} \sigma_{ij_i}$. Now we compute the semi-classical

semantics of program Y_i:

$$\begin{aligned}
\lceil Y_i \rceil(\delta_i\overline{\sigma}_i)|\Psi_i\rangle &= \lceil X_i \rceil(\overline{\sigma}_i)(\lceil Q_i \rceil(\delta_i)|\Psi_i\rangle) \\
&= \lceil X_i \rceil(\overline{\sigma}_i)\left(\sum_{j_i=1}^{n_i}(F_{i\delta_i}|j_i\rangle)\,|\psi_{ij_i}\rangle\right) \\
&= \lceil X_i \rceil(\overline{\sigma}_i)\left[\sum_{j_i=1}^{n_i}\left(\sum_{l_i=1}^{n_i}\langle l_i|F_{i\delta_i}|j_i\rangle|l_i\rangle\right)|\psi_{ij_i}\rangle\right] \\
&= \lceil X_i \rceil(\overline{\sigma}_i)\left[\sum_{l_i=1}^{n_i}|l_i\rangle\left(\sum_{j_i=1}^{n_i}\langle l_i|F_{i\delta_i}|j_i\rangle|\psi_{ij_i}\rangle\right)\right] \\
&= \sum_{l_i=1}^{n_i}\left[\Lambda_{il_i}\cdot|l_i\rangle\left(\sum_{j_i=1}^{n_i}\langle l_i|F_{i\delta_i}|j_i\rangle E_{(il_i)\sigma_{il_i}}|\psi_{ij_i}\rangle\right)\right]
\end{aligned} \tag{58}$$

where the coefficients:

$$\Lambda_{il_i} = \prod_{l\neq l_i}\lambda_{(il)\sigma_{il}},$$

$$\lambda_{(il)\sigma_{il}} = \sqrt{\frac{trE^{\dagger}_{(il)\sigma_{il}}E_{(il)\sigma_{il}}}{\sum_{k=1}^{n_i}trE^{\dagger}_{(ik)\sigma_{ik}}E_{(ik)\sigma_{ik}}}}$$

for each $1 \leq l \leq n_i$. Then using equation (58), we can further compute the semi-classical semantics of program X:

$$\begin{aligned}
\lceil X \rceil(\oplus_{i=1}^{m}(\delta_i\overline{\sigma}_i))|\Psi\rangle &= \sum_{i=1}^{m}(\Gamma_i\cdot|i\rangle\lceil Y_i \rceil(\delta_i\overline{\sigma}_i)|\Psi_i\rangle) \\
&= \sum_{i=1}^{m}\sum_{l_i=1}^{n_i}\left[\Gamma_i\cdot\Lambda_{il_i}\cdot|il_i\rangle\left(\sum_{j_i=1}^{n_i}\langle l_i|F_{i\delta_i}|j_i\rangle E_{(il_i)\sigma_{il_i}}|\psi_{ij_i}\rangle\right)\right]
\end{aligned} \tag{59}$$

where

$$\Gamma_i = \prod_{h\neq i}\gamma_{h\overline{\sigma}_h},$$

$$\gamma_{i\overline{\sigma}_i} = \sqrt{\frac{tr\lceil Y_i \rceil(\delta_i\overline{\sigma}_i)^{\dagger}\lceil Y_i \rceil(\delta_i\overline{\sigma}_i)}{\sum_{h=1}^{m}tr\lceil Y_h \rceil(\delta_h\overline{\sigma}_h)^{\dagger}\lceil Y_h \rceil(\delta_h\overline{\sigma}_h)}}. \tag{60}$$

On the other hand, we can compute the semi-classical semantics of program T:

$$\begin{aligned}
\lceil T\rceil(\oplus_{i=1}^m\delta_i)|\Psi\rangle &= \lceil T\rceil(\oplus_{i=1}^m\delta_i)\left(\sum_{i=1}^m|i\rangle|\Psi_i\rangle\right)\\
&=\sum_{i=1}^m(\Theta_i\cdot|i\rangle F_{i\delta_i}|\Psi_i\rangle)\\
&=\sum_{i=1}^m\left[\Theta_i\cdot|i\rangle\left(\sum_{j_i=1}^{n_i}(F_{i\delta_i}|j_i\rangle)|\psi_{ij_i}\rangle\right)\right]\\
&=\sum_{i=1}^m\left[\Theta_i\cdot|i\rangle\left(\sum_{j_i=1}^{n_i}\left(\sum_{l_i=1}^{n_i}\langle l_i|F_{i\delta_i}|j_i\rangle|l_i\rangle\right)|\psi_{ij_i}\rangle\right)\right]\\
&=\sum_{i=1}^m\sum_{l_i=1}^{n_i}\left[\Theta_i\cdot|il_i\rangle\left(\sum_{j_i=1}^{n_i}\langle l_i|F_{i\delta_i}|j_i\rangle|\psi_{ij_i}\rangle\right)\right]
\end{aligned}$$

where

$$\Theta_i=\prod_{h\neq i}\theta_{h\delta_h},$$

$$\theta_{i\delta_i}=\sqrt{\frac{trF_{i\delta_i}^\dagger E_{i\delta_i}}{\sum_{h=1}^m trF_{h\delta_h}^\dagger F_{h\delta_h}}}$$

for every $1\leq i\leq m$. Consequently, we obtain the semi-classical semantics of program $T;Z$:

$$\begin{aligned}
&\lceil T;Z\rceil((\oplus_{i=1}^m\delta_i)(\oplus_{i=1}^m\overline{\sigma}_i))|\Psi\rangle\\
&\quad=\lceil Z\rceil(\oplus_{i=1}^m\overline{\sigma}_i)(\lceil T\rceil(\oplus_{i=1}^m\delta_i)|\Psi\rangle)\\
&\quad=\lceil Z\rceil(\oplus_{i=1}^m\oplus_{l_i=1}^{n_i}\sigma_{ij_i})\\
&\qquad\left(\sum_{i=1}^m\sum_{l_i=1}^{n_i}\left[\Theta_i\cdot|il_i\rangle\left(\sum_{j_i=1}^{n_i}\langle l_i|F_{i\delta_i}|j_i\rangle|\psi_{ij_i}\rangle\right)\right]\right)\\
&\quad=\sum_{i=1}^m\sum_{l_i=1}^{n_i}\left[\alpha^{il_i}_{\{\sigma_{jk_j}\}_{(j,k_j)\neq(i,l_i)}}\cdot\Theta_i\cdot|il_i\rangle\left(\sum_{j_i=1}^{n_i}\langle l_i|F_{i\delta_i}|j_i\rangle E_{(il_i)\sigma_{il_i}}|\psi_{ij_i}\rangle\right)\right].
\end{aligned}\tag{61}$$

By comparing equations (59) and (61), we see that it suffices to take

$$\alpha^{il_i}_{\{\sigma_{jk_j}\}_{(j,k_j)\neq(i,l_i)}}=\frac{\Gamma_i\cdot\Delta_{il_i}}{\Theta_i}\tag{62}$$

for all i,l_i and $\{\sigma_{jk_j}\}_{(j,k_j)\neq(i,l_i)}$. What remains to prove is the normalization condition:

$$\sum_{\{\sigma_{jk_j}\}_{(j,k_j)\neq(i,l_i)}}\left|\alpha^{il_i}_{\{\sigma_{jk_j}\}_{(j,k_j)\neq(i,l_i)}}\right|^2=1.\tag{63}$$

To do this, we first compute coefficients $\gamma_{i\overline{\sigma}_i}$. Let $\{|\varphi\rangle\}$ be an orthonormal basis of $\mathcal{H}_{\bigcup_{j_i=1}^{n_i} qvar(R_{ij_i})}$. Then we have:

$$G_{\varphi j_i} \stackrel{\triangle}{=} \lceil Y_i \rceil(\delta_i \overline{\sigma}_i)|\varphi\rangle|j_i\rangle = \sum_{l_i=1}^{n_i} \Lambda_{il_i} \cdot \langle l_i|F_{i\delta_i}|j_i\rangle E_{(il_i)\sigma_{il_i}}|\varphi\rangle|l_i\rangle.$$

It follows that

$$\begin{aligned} G_{\varphi j_i}^\dagger G_{\varphi j_i} &= \sum_{l_i,l_i'=1}^{n_i} \Lambda_{il_i} \cdot \Lambda_{il_i'} \langle j_i|F_{i\delta_i}^\dagger|l_i\rangle\langle l_i'|F_{i\delta_i}|j_i\rangle\langle\varphi|E_{(il_i)\sigma_{il_i}}^\dagger E_{(il_i')\sigma_{il_i'}}|\varphi\rangle\langle l_i|l_i'\rangle \\ &= \sum_{l_i=1}^{n_i} \Lambda_{il_i}^2 \cdot \langle j_i|F_{i\delta_i}^\dagger|l_i\rangle\langle l_i|F_{i\delta_i}|j_i\rangle\langle\varphi|E_{(il_i)\sigma_{il_i}}^\dagger E_{(il_i)\sigma_{il_i}}|\varphi\rangle. \end{aligned}$$

Furthermore, we obtain:

$$\begin{aligned} tr\lceil Y_i \rceil(\delta_i\overline{\sigma}_i)^\dagger \lceil Y_i \rceil(\delta_i\overline{\sigma}_i) &= \sum_{\varphi,j_i} G_{\varphi j_i}^\dagger G_{\varphi j_i} \\ &= \sum_{l_i=1}^{n_i} \Lambda_{il_i}^2 \cdot \left(\sum_{j_i} \langle j_i|F_{i\delta_i}^\dagger|l_i\rangle\langle l_i|F_{i\delta_i}|j_i\rangle\right)\left(\sum_{\varphi} \langle\varphi|E_{(il_i)\sigma_{il_i}}^\dagger E_{(il_i)\sigma_{il_i}}|\varphi\rangle\right) \quad (64) \\ &= \sum_{l_i=1}^{n_i} \Lambda_{il_i}^2 \cdot tr(F_{i\delta_i}^\dagger|l_i\rangle\langle l_i|F_{i\delta_i}) tr(E_{(il_i)\sigma_{il_i}}^\dagger E_{(il_i)\sigma_{il_i}}). \end{aligned}$$

Now a routine but tedious calculation yields equation (63) through substituting equation (64) into (60) and then substituting equations (60) and (62) into (63).

(4) Finally, we prove clause (4). To prove the first equality, we write:

$$\begin{aligned} X &\stackrel{\triangle}{=} \mathbf{qif}\ (\square i \cdot |i\rangle \to P_i)\ \mathbf{fiq}, \\ Y &\stackrel{\triangle}{=} \mathbf{qif}\ (\overline{\alpha})(\square i \cdot |i\rangle \to (P_i; Q))\ \mathbf{fiq}. \end{aligned}$$

Then by definition we have $LHS = P; X; Q$ and $RHS = P; Y$. So, it suffices to show that $X; Q \equiv_{CF} Y$. Suppose that $\lceil P_i \rceil(\sigma_i) = E_{i\sigma_i}$ for every $\sigma_i \in \Delta(P_i)$ and $\lceil Q \rceil(\delta) = F_\delta$ for every $\delta \in \Delta(Q)$, and suppose that

$$|\Psi\rangle = \sum_{i=1}^{n} |i\rangle|\psi_i\rangle,$$

where $|\psi_i\rangle \in \mathcal{H}_{\bigcup_i qvar(P_i)}$ for all i. Then it holds that

$$\begin{aligned} \lceil X; Q \rceil((\oplus_{i=1}^n \sigma_i)\delta)|\Psi\rangle &= \lceil Q \rceil(\delta)(\lceil X \rceil(\oplus_{i=1}^n \sigma_i)|\Psi\rangle) \\ &= F_\delta\left(\sum_{i=1}^{n} \Lambda_i |i\rangle(E_{i\sigma_i}|\psi_i\rangle)\right) \\ &= \sum_{i=1}^{n} \Lambda_i \cdot |i\rangle(F_\delta E_{i\sigma_i}|\psi_i\rangle) \end{aligned}$$

because $qvar(P) \cap qvar(Q) = \emptyset$, where

$$\Lambda_i = \prod_{k \neq i} \lambda_{k\sigma_k},$$

$$\lambda_{i\sigma_i} = \sqrt{\frac{trE_{i\sigma_i}^{\dagger} E_{i\sigma_i}}{\sum_{k=1}^{n} trE_{k\sigma_k}^{\dagger} E_{k\sigma_k}}}. \tag{65}$$

Furthermore, we have:

$$\begin{aligned} & tr_{\mathcal{H}_{qvar(P)}}([\![X;Q]\!](|\Psi\rangle\langle\Psi|)) \\ & \quad = tr_{\mathcal{H}_{qvar(P)}}\left[\sum_{\{\sigma_i\},\delta}\sum_{i,j} \Lambda_i\Lambda_j \cdot |i\rangle\langle j|(F_\delta E_{i\sigma_i}|\psi_i\rangle\langle\psi_j|E_{j\sigma_j}^{\dagger}F_\delta^{\dagger})\right] \\ & \quad = \sum_{\{\sigma_i\},\delta}\sum_{i} \Lambda_i^2 \cdot F_\delta E_{i\sigma_i}|\psi_i\rangle\langle\psi_i|E_{i\sigma_i}^{\dagger}F_\delta^{\dagger}. \end{aligned} \tag{66}$$

On the other hand, we can compute the semi-classical semantics of Y:

$$\begin{aligned} \lceil Y \rceil(\oplus_{i=1}^{n}\sigma_i\delta_i)|\Psi\rangle &= \sum_{i=1}^{n} \alpha^{(i)}_{\{\sigma_k,\delta_k\}_{k\neq i}} \cdot |i\rangle(\lceil P_i;Q\rceil(\sigma_i\delta_i)|\psi_i\rangle) \\ &= \sum_{i=1}^{n} \alpha^{(i)}_{\{\sigma_k,\delta_k\}_{k\neq i}} \cdot |i\rangle(F_{\delta_i}E_{\sigma_i}|\psi_i\rangle). \end{aligned}$$

Furthermore, we obtain:

$$\begin{aligned} & tr_{\mathcal{H}_{qvar(P)}}([\![Y]\!](|\Psi\rangle\langle\Psi|)) \\ & = tr_{\mathcal{H}_{qvar(P)}}\left[\sum_{\{\sigma_i,\delta_i\}}\sum_{i,j} \alpha^{(i)}_{\{\sigma_k,\delta_k\}_{k\neq i}}(\alpha^{(j)}_{\{\sigma_l,\delta_l\}_{l\neq j}})^* \cdot |i\rangle\langle j|(F_{\delta_i}E_{i\sigma_i}|\psi_i\rangle\langle\psi_j|E_{j\sigma_j}^{\dagger}F_{\delta_j}^{\dagger})\right] \\ & = \sum_{\{\sigma_i\},\delta}\sum_{i} \left|\alpha^{(i)}_{\{\sigma_k,\delta_k\}_{k\neq i}}\right|^2 \cdot F_{\delta_i}E_{i\sigma_i}|\psi_i\rangle\langle\psi_i|E_{i\sigma_i}^{\dagger}F_{\delta_i}^{\dagger}. \end{aligned} \tag{67}$$

Comparing equations (66) and (67), we see that

$$tr_{\mathcal{H}_{qvar(P)}}([\![X;Q]\!](|\Psi\rangle\langle\Psi|)) = tr_{\mathcal{H}_{qvar(P)}}([\![Y]\!](|\Psi\rangle\langle\Psi|))$$

if we take

$$\alpha^{(i)}_{\{\sigma_k,\delta_k\}_{k\neq i}} = \frac{\Lambda_i}{\sqrt{|\Delta(Q)|}}$$

for all $i, \{\sigma_k\}$ and $\{\delta_k\}$. Since $qvar(P) \subseteq cvar(X;Q) \cup cvar(Y)$, it follows that

$$tr_{\mathcal{H}_{cvar(X;Q)\cup cvar(Y)}}([\![X;Q]\!](|\Psi\rangle\langle\Psi|)) = tr_{\mathcal{H}_{cvar(X;Q)\cup cvar(Y)}}([\![Y]\!](|\Psi\rangle\langle\Psi|)).$$

Therefore, we can assert that

$$tr_{\mathcal{H}_{cvar(X;Q)\cup cvar(Y)}}([\![X;Q]\!](\rho)) = tr_{\mathcal{H}_{cvar(X;Q)\cup cvar(Y)}}([\![Y]\!](\rho))$$

for all density operator ρ by spectral decomposition, and $X;Q \equiv_{CF} Y$.

For the special case where Q contains no measurements, $\Delta(Q)$ is a singleton, say $\{\delta\}$. We write:

$$Z \stackrel{\triangle}{=} \textbf{qif } (\square i \cdot |i\rangle \rightarrow (P_i;Q)) \textbf{ fiq}.$$

Then

$$\lceil Z \rceil (\oplus_{i=1}^{n} \sigma_i \delta)|\Psi\rangle = \sum_{i=1}^{n} \left(\prod_{k \neq i} \theta_{k\sigma_k} \right) \cdot |i\rangle (F_\delta E_{\sigma_i} |\psi_i\rangle),$$

where

$$\theta_{i\sigma_i} = \sqrt{\frac{tr E_{i\sigma_i}^\dagger F_\delta^\dagger F_\delta E_{i\sigma_i}}{\sum_{k=1}^{n} tr E_{k\sigma_k}^\dagger F_\delta^\dagger F_\delta E_{k\sigma_k}}} = \lambda_{i\sigma_i},$$

where $\lambda_{i\sigma_i}$ is given by equation (65), because $F_\delta^\dagger F_\delta$ is the identity operator. Consequently, $\lceil X;Q \rceil = \lceil Z \rceil$, and we complete the proof of the second equality of clause (4).