CHAPTER

Syntax and semantics of quantum programs

3

In Section 2.3, several quantum algorithms were presented at the very low-level model of quantum computing – quantum circuits. How can we design and implement higher-level programming languages for quantum computers? From this chapter on, we systematically develop the foundations of quantum programming.

As the first step, let us see how a classical programming language can be directly extended for programming a quantum computer. As pointed out in Sections 1.1 and 1.2, this issue had been the main concern of early research on quantum programming. This chapter studies a class of simple quantum generalizations of classical programs – quantum programs with classical control: i.e., programs in the superposition-of-data paradigm. The design idea of this class of quantum programs was briefly introduced in Subsection 1.2.1. The control flow of these programs will be further discussed shortly.

The chapter is divided into three parts:

- The **while**-language constitutes the "kernel" of many classical programming languages. The first part of this chapter introduces a quantum extension of the **while**-language. This part consists of Sections 3.1 to 3.3: Section 3.1 defines the syntax of the quantum **while**-language. Sections 3.2 and 3.3 present its operational and denotational semantics, respectively.

  Along the way, we briefly prepare a theory of quantum domains needed for a characterization of the denotational semantics of the loop in the quantum **while**-language. For readability, the lengthy proofs of some lemmas about quantum domains are postponed to a separate section – Section 3.6 – at the end of the chapter.
- The second part – Section 3.4 – extends the quantum **while**-language by adding recursive quantum programs (with classical control). The operational and denotational semantics of recursive quantum programs are defined. Here, the theory of quantum domains is also needed to deal with the denotational semantics.
- The third part – Section 3.5 – presents an illustrative example showing how the Grover quantum search can be programmed in the language defined in this chapter.

Foundations of Quantum Programming. http://dx.doi.org/10.1016/B978-0-12-802306-8.00003-3
Copyright © 2016 Elsevier Inc. All rights reserved.

3.1 SYNTAX

In this section, we define the syntax of a quantum extension of classical **while**-language. Recall that a classical **while**-program is generated by the grammar:

$$\begin{aligned} S ::= \ & \textbf{skip} \mid u := t \mid S_1; S_2 \\ & \mid \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi} \\ & \mid \textbf{while } b \textbf{ do } S \textbf{ od}. \end{aligned}$$

Here, S, S_1, S_2 are programs, u is a variable, t is an expression, and b is a Boolean expression. Intuitively, **while**-programs are executed as follows:

- The statement "**skip**" does nothing but terminates.
- The assignment "$u := t$" assigns the value of expression t to variable u.
- The sequential composition "$S_1; S_2$" first executes S_1, and when S_1 terminates, it executes S_2.
- The conditional statement "**if** b **then** S_1 **else** S_2 **fi**" starts from evaluating the Boolean expression b: if b is true, S_1 is executed; otherwise, S_2 is executed. The conditional statement can be generalized to the case statement:

$$\begin{aligned} & \textbf{if } G_1 \to S_1 \\ & \square\ G_2 \to S_2 \\ & \quad \ldots\ldots \\ & \square\ G_n \to S_n \\ & \textbf{fi} \end{aligned} \tag{3.1}$$

  or more compactly

$$\textbf{if } (\square i \cdot G_i \to S_i) \textbf{ fi}$$

  where $G_1, G_2, \ldots, G_n$ are Boolean expressions, called guards, and $S_1, S_2, \ldots, S_n$ are programs. The case statement starts from evaluating guards: if G_i is true, then the corresponding subprogram S_i is executed.
- The **while**-loop "**while** b **do** S **od**" starts from evaluating the loop guard b: if b is false, the loop terminates immediately; otherwise, the loop body S is executed, and when S terminates, the process is repeated.

Now we expand the **while**-language so that it can be used for quantum programming. We first fix the alphabet of the quantum **while**-language:

- A countably infinite set *qVar* of quantum variables. The symbols $q, q', q_0, q_1, q_2, \ldots$ will be used as meta-variables ranging over quantum variables.

- Each quantum variable $q \in qVar$ has a type $\mathcal{H}_q$, which is a Hilbert space – the state space of the quantum system denoted by q. For simplicity, we only consider two basic types:

$$\textbf{Boolean} = \mathcal{H}_2, \qquad \textbf{integer} = \mathcal{H}_\infty.$$

  Note that the sets denoted by types **Boolean** and **integer** in classical computation are exactly the computational bases of $\mathcal{H}_2$ and $\mathcal{H}_\infty$, respectively (see Examples 2.1.1 and 2.1.2). The main results presented in this chapter can be easily generalized to the case with more data types.

A quantum register is a finite sequence of distinct quantum variables. (The notion of quantum register in Section 2.2 is slightly generalized here by allowing it to contain other quantum variables than qubit variables.) The state Hilbert space of a quantum register $\overline{q} = q_1, \ldots, q_n$ is the tensor product of the state spaces of the quantum variables occurring in $\overline{q}$:

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

When necessary, we write $|\psi\rangle_{q_i}$ to indicate that $|\psi\rangle$ is a state of quantum variable q_i; that is, $|\psi\rangle$ is in $\mathcal{H}_{q_i}$. Thus, $|\psi\rangle_{q_i}\langle\varphi_i|$ denotes the outer product of states $|\psi\rangle$ and $|\varphi\rangle$ of q_i, and $|\psi_1\rangle_{q_1} \ldots |\psi_n\rangle_{q_n}$ is a state in $\mathcal{H}_{\overline{q}}$ in which q_i is in state $|\psi_i\rangle$ for every $1 \leq i \leq n$.

With these ingredients, we can define programs in the quantum **while**-language.

Definition 3.1.1. *Quantum programs are generated by the syntax:*

$$\begin{aligned} S ::= \ & \textbf{skip} \mid q := |0\rangle \mid \overline{q} := U[\overline{q}] \mid S_1; S_2 \\ & \mid \textbf{if } (\square m \cdot M[\overline{q}] = m \rightarrow S_m) \textbf{ fi} \\ & \mid \textbf{while } M[\overline{q}] = 1 \textbf{ do } S \textbf{ od}. \end{aligned} \tag{3.2}$$

This definition deserves a careful explanation:

- As in the classical **while**-language, statement "**skip**" does nothing and terminates immediately.
- The initialization statement "$q := |0\rangle$" sets quantum variable q to the basis state $|0\rangle$. For any pure state $|\psi\rangle \in \mathcal{H}_q$, there is obviously a unitary operator U in $\mathcal{H}_q$ such that $|\psi\rangle = U|0\rangle$. So, the system q can be prepared in state $|\psi\rangle$ by this initialization and the unitary transformation $q := U[q]$.
- The statement "$\overline{q} := U[\overline{q}]$" means that unitary transformation U is performed on quantum register $\overline{q}$, leaving the states of the quantum variables not in $\overline{q}$ unchanged.
- Sequential composition is similar to its counterpart in a classical programming language.

- The program construct

$$\begin{aligned}\textbf{if } (\square m \cdot M[\overline{q}] = m \rightarrow S_m) \textbf{ fi} \equiv \textbf{if } & M[\overline{q}] = m_1 \rightarrow S_{m_1} \\ \square \quad & m_2 \rightarrow S_{m_2} \\ & \cdots\cdots \\ \square \quad & m_n \rightarrow S_{m_n} \\ \textbf{fi} & \end{aligned} \tag{3.3}$$

  is a quantum generalization of the classical case statement (3.1). Recall that the first step of execution of statement (3.1) is to see which guard G_i is satisfied. However, according to the Postulate of quantum mechanics 3 (see Subsection 2.1.4), the way to acquire information about a quantum system is to perform a measurement on it. So, in executing the statement (3.3), quantum measurement

$$M = \{M_m\} = \{M_{m_1}, M_{m_2}, \ldots, M_{m_n}\}$$

  will be performed on quantum register $\overline{q}$, and then a subprogram S_m will be selected to be executed next according to the outcome of the measurement. An essential difference between the measurement-based case statement (3.3) and a classical case statement is that the state of program variables is changed after performing the measurement in the former, whereas it is not changed after checking the guards in the latter.
- The statement

$$\textbf{while } M[\overline{q}] = 1 \textbf{ do } S \textbf{ od} \tag{3.4}$$

  is a quantum generalization of the classical loop "**while** b **do** S **od**". To acquire information about quantum register $\overline{q}$, a measurement M is performed on it. The measurement $M = \{M_0, M_1\}$ is a yes-no measurement with only two possible outcomes: 0 ("no"), 1 ("yes"). If the outcome 0 is observed, then the program terminates, and if the outcome 1 occurs, then the program executes the subprogram S and continues. The only difference between the quantum loop (3.4) and a classical loop is that checking the loop guard b in the classical loop does not change the state of program variables, but this is not the case in the quantum loop.

Classical Control Flow:

Now it is the right time to explain that the control flow of a program in the quantum **while**-language is *classical*, as indicated at the beginning of this chapter. Recall that the control flow of a program is the order of its execution. In the quantum **while**-language, there are only two statements – the case statement (3.3) and the loop (3.4) – whose execution is determined by a choice as to which of two or more paths should be followed. The case statement (3.3) selects a command to

execute according to the outcome of measurement M: if the outcome is m_i, then the corresponding command S_{m_i} will be executed. Since the outcome of a quantum measurement is classical information, the control flow in statement (3.3) is classical. The same argument illustrates that the control flow in the loop (3.4) is classical too.

As pointed out in Subsection 1.2.2, it is also possible to define programs with quantum control flow. Quantum control flow of programs is much harder to understand, which will be the theme of Chapters 6 and 7.

Program Variables:

Before concluding this section, we present the following technical definition, which will be needed in the sequel.

Definition 3.1.2. *The set $qvar(S)$ of quantum variables in quantum program S is recursively defined as follows:*

(i) *If $S \equiv$ **skip**, then $qvar(S) = \emptyset$;*
(ii) *If $S \equiv q := |0\rangle$, then $qvar(S) = \{q\}$;*
(iii) *If $S \equiv \overline{q} := U[\overline{q}]$, then $qvar(S) = \overline{q}$;*
(iv) *If $S \equiv S_1; S_2$, then $qvar(S) = qvar(S_1) \cup qvar(S_2)$;*
(v) *If $S \equiv$ **if** $(\square m \cdot M[\overline{q}] = m \rightarrow S_m)$ **fi**, then*

$$qvar(S) = \overline{q} \cup \bigcup_m qvar(S_m);$$

(vi) *If $S \equiv$ **while** $M[\overline{q}] = 1$ **do** S **od**, then $qvar(S) = \overline{q} \cup qvar(S)$.*

3.2 OPERATIONAL SEMANTICS

The syntax of quantum **while**-programs was defined in the last section. This section defines the operational semantics of the quantum **while**-language. We first introduce several notations:

- A positive operator ρ in a Hilbert space $\mathcal{H}$ is called a *partial density operator* if $tr(\rho) \leq 1$. So, a density operator ρ (see Definition 2.1.21) is a partial density operator with $tr(\rho) = 1$. We write $\mathcal{D}(\mathcal{H})$ for the set of partial density operators in $\mathcal{H}$. In quantum programming theory, a partial density operator is a very useful notion, because a program with loops (or more generally, recursions) may not terminate with a certain probability, and its output is a partial density operator but not necessarily a density operator.
- We write $\mathcal{H}_{all}$ for the tensor product of the state Hilbert spaces of all quantum variables:

$$\mathcal{H}_{all} = \bigotimes_{q \in qVar} \mathcal{H}_q.$$

- Let $\overline{q} = q_1, \ldots, q_n$ be a quantum register. An operator A in the state Hilbert space $\mathcal{H}_{\overline{q}}$ of $\overline{q}$ has a cylindrical extension $A \otimes I$ in $\mathcal{H}_{all}$, where I is the identity operator in the state Hilbert space

$$\bigotimes_{q \in qVar \setminus \overline{q}} \mathcal{H}_q$$

  of the quantum variables that are not in $\overline{q}$. In the sequel, we will simply write A for its cylindrical extension, and it can be easily recognized from the context, without any risk of confusion.
- We will use E to denote the empty program: i.e., termination.

As in the theory of classical programming, the execution of a quantum program can be properly described in terms of transition between configurations.

Definition 3.2.1. *A quantum configuration is a pair $\langle S, \rho \rangle$, where:*

(i) *S is a quantum program or the empty program E;*
(ii) *$\rho \in \mathcal{D}(\mathcal{H}_{all})$ is a partial density operator in $\mathcal{H}_{all}$, and it is used to indicate the (global) state of quantum variables.*

A transition between quantum configurations:

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

means that after executing quantum program S one step in state ρ, the state of quantum variables becomes ρ', and S' is the remainder of S still to be executed. In particular, if $S' = E$, then S terminates in state ρ'.

Definition 3.2.2. *The operational semantics of quantum programs is the transition relation $\to$ between quantum configurations defined by the transition rules in Figure 3.1.*

The operational semantics (i.e., the relation $\to$) defined previously should be understood as the smallest transition relation between quantum configurations that satisfies the rules in Figure 3.1. Obviously, the transition rules (IN), (UT), (IF), (L0) and (L1) are determined by the postulates of quantum mechanics. As you saw in Chapter 2, probabilities always arise from the measurements in a quantum computation. But it should be noticed that the operational semantics of quantum programs is an ordinary transition relation $\to$ rather than a probabilistic transition relation. Several remarks should help the reader to understand these transition rules:

- The symbol U in the target configuration of the rule (UT) stands indeed for the cylindrical extension of U in $\mathcal{H}_{all}$. A similar remark applies to the rules (IF), (L0) and (L1) for measurements and loops.
- In the rule (IF), the outcome m is observed with probability

$$p_m = tr(M_m \rho M_m^\dagger),$$

(SK) $$\frac{}{\langle \mathbf{skip}, \rho\rangle \to \langle E, \rho\rangle}$$

(IN) $$\frac{}{\langle q := |0\rangle, \rho\rangle \to \langle E, \rho_0^q\rangle}$$

where

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

(UT) $$\frac{}{\langle \overline{q} := U[\overline{q}], \rho\rangle \to \langle E, U\rho U^\dagger\rangle}$$

(SC) $$\frac{\langle S_1, \rho\rangle \to \langle S_1', \rho'\rangle}{\langle S_1; S_2, \rho\rangle \to \langle S_1'; S_2, \rho'\rangle}$$

where we make the convention that $E; S_2 = S_2$.

(IF) $$\frac{}{\langle \mathbf{if}\ (\square m \cdot M[\overline{q}] = m \to S_m)\ \mathbf{fi}, \rho\rangle \to \langle S_m, M_m\rho M_m^\dagger\rangle}$$

for each possible outcome m of measurement $M = \{M_m\}$.

(L0) $$\frac{}{\langle \mathbf{while}\ M[\overline{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}, \rho\rangle \to \langle E, M_0\rho M_0^\dagger\rangle}$$

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

FIGURE 3.1

Transition rules for quantum **while**-programs.

and in this case, after the measurement the state becomes

$$\rho_m = M_m\rho M_m^\dagger / p_m.$$

So, a natural presentation of the rule (IF) is the probabilistic transition:

$$\frac{}{\langle \mathbf{if}\ (\square m \cdot M[\overline{q}] = m \to S_m)\ \mathbf{fi}, \rho\rangle \xrightarrow{p_m} \langle S_m, \rho_m\rangle}$$

However, if we encode both probability p_m and density operator ρ_m into a partial density operator

$$M_m \rho M_m^\dagger = p_m \rho_m,$$

then the rule can be presented as an ordinary (a nonprobabilistic) transition.

- Likewise, in the rules (L0) and (L1) the measurement outcomes 0 and 1 occur with probabilities:

$$p_0 = tr(M_0 \rho M_0^\dagger), \qquad p_1 = tr(M_1 \rho M_1^\dagger),$$

respectively, and the state becomes $M_0 \rho M_0^\dagger / p_0$ from ρ when the outcome is 0, and it becomes $M_1 \rho M_1^\dagger / p_1$ when the outcome is 1. These probabilities and postmeasurement states are encoded into partial density operators so that the rules (L0) and (L1) can be stated as ordinary transitions instead of probabilistic transitions.

From this discussion, we see that, exactly, the convention of combining probabilities with postmeasurement states enables us to define the operational semantics $\rightarrow$ as a nonprobabilistic transition relation.

Transition Rules for Pure States:

The transition rules in Figure 3.1 were stated in the language of density operators. As we will see in the next section, this general setting provides us with an elegant formulation of denotational semantics. However, pure states are usually more convenient in applications. So, we display the pure state variants of these transition rules in Figure 3.2. In the rules for pure states, a configuration is a pair

$$\langle S, |\psi\rangle \rangle$$

with S being a quantum program or the empty program E and $|\psi\rangle$ a pure state in $\mathcal{H}_{all}$. As indicated previously, transitions in Figure 3.1 are all nonprobabilistic. However, the transitions in rules (IF$'$), (L0$'$) and (L1$'$) are probabilistic in the form of

$$\langle S, |\psi\rangle \rangle \xrightarrow{p} \langle S', |\psi'\rangle \rangle.$$

Whenever the probability $p = 1$, then this transition is abbreviated to

$$\langle S, |\psi\rangle \rangle \rightarrow \langle S', |\psi'\rangle \rangle.$$

Of course, the rules in Figure 3.2 are special cases of the corresponding rules in Figure 3.1. Conversely, using the correspondence between density operators and ensembles of pure states, the rules in Figure 3.1 can be derived from their counterparts in Figure 3.2.

The reader might have noticed that Figure 3.2 does not include a pure state version of the initialization rule (IN). Actually, the rule (IN) has no pure state version because it is possible that an initialization will transfer a pure state into a mixed state: although

(SK') $$\frac{}{\langle \mathbf{skip}, |\psi\rangle\rangle \rightarrow \langle E, |\psi\rangle\rangle}$$

(UT') $$\frac{}{\langle \overline{q} := U[\overline{q}], |\psi\rangle\rangle \rightarrow \langle E, U|\psi\rangle\rangle}$$

(SC') $$\frac{\langle S_1, |\psi\rangle\rangle \xrightarrow{p} \langle S_1', |\psi'\rangle\rangle}{\langle S_1; S_2, |\psi\rangle\rangle \xrightarrow{p} \langle S_1'; S_2, |\psi'\rangle\rangle}$$

where we make the convention that $E; S_2 = S_2$.

(IF') $$\frac{}{\langle \mathbf{if}\ (\square m \cdot M[\overline{q}] = m \rightarrow S_m)\ \mathbf{fi}, |\psi\rangle\rangle \xrightarrow{||M_m|\psi\rangle||^2} \langle S_m, \frac{M_m|\psi\rangle}{||M_m|\psi\rangle||}\rangle}$$

for each outcome m of measurement $M = \{M_m\}$.

(L0′) $$\frac{}{\langle \mathbf{while}\ M[\overline{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}, |\psi\rangle\rangle \xrightarrow{||M_0|\psi\rangle||^2} \langle E, \frac{M_0|\psi\rangle}{||M_0|\psi\rangle||}\rangle}$$

(L1′) $$\frac{}{\langle \mathbf{while}\ M[\overline{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}, |\psi\rangle\rangle \xrightarrow{||M_1|\psi\rangle||^2} \langle S; \mathbf{while}\ M[\overline{q}] = 1\ \mathbf{do}\ S\ \mathbf{od}, \frac{M_1|\psi\rangle}{||M_1|\psi\rangle||}\rangle}$$

FIGURE 3.2

Transition rules for quantum **while**-programs in pure states.

the initialization $q := |0\rangle$ changes the state of local variable q into a pure state $|0\rangle$, its side effect on other variables may cause a transition of a global state $|\psi\rangle \in \mathcal{H}_{all}$ of all variables *qVar* into a mixed state. To see the rule (IN) more clearly, let us look at the case of $type(q) =$ **integer** as an example.

Example 3.2.1

(i) *We first consider the case where ρ is a pure state; that is, $\rho = |\psi\rangle\langle\psi|$ for some $|\psi\rangle \in \mathcal{H}_{all}$. We can write $|\psi\rangle$ in the form:*

$$|\psi\rangle = \sum_k \alpha_k |\psi_k\rangle,$$

where all $|\psi_k\rangle$ *are product states, say*

$$|\psi_k\rangle = \bigotimes_{p\in qVar} |\psi_{kp}\rangle.$$

Then

$$\rho = \sum_{k,l} \alpha_k \alpha_l^* |\psi_k\rangle\langle\psi_l|.$$

After the initialization $q := |0\rangle$ *the state becomes:*

$$\begin{aligned}
\rho_0^q &= \sum_{n=-\infty}^{\infty} |0\rangle_q\langle n|\rho|n\rangle_q\langle 0| \\
&= \sum_{k,l} \alpha_k \alpha_l^* \left(\sum_{n=-\infty}^{\infty} |0\rangle_q\langle n|\psi_k\rangle\langle\psi_l|n\rangle_q\langle 0| \right) \\
&= \sum_{k,l} \alpha_k \alpha_l^* \left(\sum_{n=-\infty}^{\infty} \langle\psi_{lq}|n\rangle\langle n|\psi_{kq}\rangle \right) \left(|0\rangle_q\langle 0| \otimes \bigotimes_{p\neq q} |\psi_{kp}\rangle\langle\psi_{lp}| \right) \\
&= \sum_{k,l} \alpha_k \alpha_l^* \langle\psi_{lq}|\psi_{kq}\rangle \left(|0\rangle_q\langle 0| \otimes \bigotimes_{p\neq q} |\psi_{kp}\rangle\langle\psi_{lp}| \right) \\
&= |0\rangle_q\langle 0| \otimes \left(\sum_{k,l} \alpha_k \alpha_l^* \langle\psi_{lq}|\psi_{kq}\rangle \bigotimes_{p\neq q} |\psi_{kp}\rangle\langle\psi_{lp}| \right). \qquad (3.5)
\end{aligned}$$

It is obvious that ρ_0^q *is not necessary to be a pure state although* ρ *is a pure state.*

(ii) *In general, suppose that* ρ *is generated by an ensemble* $\{(p_i, |\psi_i\rangle)\}$ *of pure states; that is,*

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|.$$

For each i, we write $\rho_i = |\psi_i\rangle\langle\psi_i|$ *and assume that it becomes* ρ_{i0}^q *after the initialization. By the previous argument, we can write* ρ_{i0}^q *in the form:*

$$\rho_{i0}^q = |0\rangle_q\langle 0| \otimes \left(\sum_k \alpha_{ik} |\varphi_{ik}\rangle\langle\varphi_{ik}| \right),$$

where $|\varphi_{ik}\rangle \in \mathcal{H}_{qVar\setminus\{q\}}$ *for all k. Then the initialization causes* ρ *to become*

$$\begin{aligned}\rho_0^q &= \sum_{n=-\infty}^{\infty} |0\rangle_q\langle n|\rho|n\rangle_q\langle 0| \\ &= \sum_i p_i \left(\sum_{n=-\infty}^{\infty} |0\rangle_q\langle n|\rho_i|n\rangle_q\langle 0| \right) \\ &= |0\rangle_q\langle 0| \otimes \left(\sum_{i,k} p_i \alpha_{ik} |\varphi_{ik}\rangle\langle\varphi_{ik}| \right).\end{aligned} \tag{3.6}$$

From equations (3.5) and (3.6) we see that the state of quantum variable q is set to be $|0\rangle$ *and the states of the other quantum variables are unchanged.*

Exercise 3.2.1. *Find a necessary and sufficient condition under which* ρ_0^q *in equation (3.5) is a pure state. [Hint: A density operator* ρ *is a pure state if and only if* $tr(\rho^2) = 1$*; see [174], Exercise 2.71.]*

Computation of a Program:

Now the notion of computation of a quantum program can be naturally defined in terms of its transitions.

Definition 3.2.3. *Let S be a quantum program and* $\rho \in \mathcal{D}(\mathcal{H}_{all})$.

(i) *A transition sequence of S starting in* ρ *is a finite or infinite sequence of configurations in the following form:*

$$\langle S, \rho\rangle \to \langle S_1, \rho_1\rangle \to \ldots \to \langle S_n, \rho_n\rangle \to \langle S_{n+1}, \rho_{n+1}\rangle \to \ldots$$

such that $\rho_n \neq 0$ *for all n (except the last n in the case of a finite sequence).*

(ii) *If this sequence cannot be extended, then it is called a computation of S starting in* ρ.

- **(a)** *If a computation is finite and its last configuration is* $\langle E, \rho'\rangle$*, then we say that it terminates in* ρ'.
- **(b)** *If it is infinite, then we say that it diverges. Moreover, we say that S can diverge from* ρ *whenever it has a diverging computation starting in* ρ.

To illustrate this definition, let us see a simple example.

Example 3.2.2. *Suppose that* $type(q_1) =$ **Boolean** *and* $type(q_2) =$ **integer**. *Consider the program*

$$\begin{aligned} S \equiv q_1 := |0\rangle;\ q_2 := |0\rangle;\ q_1 := H[q_1];\ q_2 := q_2 + 7; \\ \textbf{if } M[q_1] = 0 \to S_1 \\ \square \qquad\quad 1 \to S_2 \\ \textbf{fi} \end{aligned}$$

where:

- *H is the Hadamard transformation, $q_2 := q_2 + 7$ is a rewriting of*

$$q_2 := T_7[q_2]$$

  with T_7 being the translation operator defined in Example 2.1.4;
- *M is the measurement in the computational basis $|0\rangle, |1\rangle$ of $\mathcal{H}_2$; that is, $M = \{M_0, M_1\}$, $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$;*
- *$S_1 \equiv$ **skip**;*
- *$S_2 \equiv$ **while** $N[q_2] = 1$ **do** $q_1 := X[q_1]$ **od**, where X is the Pauli matrix (i.e., the NOT gate), $N = \{N_0, N_1\}$, and*

$$N_0 = \sum_{n=-\infty}^{0} |n\rangle\langle n|, \qquad N_1 = \sum_{n=1}^{\infty} |n\rangle\langle n|.$$

Let $\rho = |1\rangle_{q_1}\langle 1| \otimes |-1\rangle_{q_2}\langle -1| \otimes \rho_0$ and

$$\rho_0 = \bigotimes_{q \neq q_1, q_2} |0\rangle_q\langle 0|.$$

Then the computations of S starting in ρ are:

$$\begin{aligned}
\langle S, \rho\rangle &\to \langle q_2 := |0\rangle;\ q_1 := H[q_1];\ q_2 := q_2 + 7;\ \textbf{if}\ldots\textbf{fi}, \rho_1\rangle \\
&\to \langle q_1 := H[q_1];\ q_2 := q_2 + 7;\ \textbf{if}\ldots\textbf{fi}, \rho_2\rangle \\
&\to \langle q_2 := q_2 + 7;\ \textbf{if}\ldots\textbf{fi}, \rho_3\rangle \\
&\to \langle \textbf{if}\ldots\textbf{fi}, \rho_4\rangle \\
&\to \begin{cases} \langle S_1, \rho_5\rangle \to \langle E, \rho_5\rangle, \\ \langle S_2, \rho_6\rangle, \end{cases}
\end{aligned}$$

$$\begin{aligned}
\langle S_2, \rho_6\rangle &\to \langle q_1 := X[q_1];\ S_2, \rho_6\rangle \\
&\to \langle S_2, \rho_5\rangle \\
&\to \ldots \\
&\to \langle q_1 := X[q_1]; S_2, \rho_6\rangle \quad (\textit{after } 2n-1 \textit{ transitions}) \\
&\to \langle S_2, \rho_5\rangle \\
&\to \ldots
\end{aligned}$$

where:

$$\begin{aligned}
\rho_1 &= |0\rangle_{q_1}\langle 0| \otimes |-1\rangle_{q_2}\langle -1| \otimes \rho_0, \\
\rho_2 &= |0\rangle_{q_1}\langle 0| \otimes |0\rangle_{q_2}\langle 0| \otimes \rho_0, \\
\rho_3 &= |+\rangle_{q_1}\langle +| \otimes |0\rangle_{q_2}\langle 0| \otimes \rho_0, \\
\rho_4 &= |+\rangle_{q_1}\langle +| \otimes |7\rangle_{q_2}\langle 7| \otimes \rho_0,
\end{aligned}$$

$$\rho_5 = \frac{1}{2}|0\rangle_{q_1}\langle 0| \otimes |7\rangle_{q_2}\langle 7| \otimes \rho_0,$$
$$\rho_6 = \frac{1}{2}|1\rangle_{q_1}\langle 1| \otimes |7\rangle_{q_2}\langle 7| \otimes \rho_0.$$

So, S can diverge from ρ. Note that S_2 also has the transition

$$\langle S_2, \rho_6\rangle \to \langle E, 0_{\mathcal{H}_{all}}\rangle,$$

but we always discard the transitions in which the partial density operator of the target configuration is a zero operator.

Nondeterminism:

To conclude this section, let us observe an interesting difference between the operational semantics of classical and quantum **while**-programs. Classical **while**-programs are a typical class of deterministic programs that have exactly one computation starting in a given state. (Here, if not only the conditional statement "**if** ... **then** ... **else**" but also the case statement (3.1) is included, then it is assumed that the guards $G_1, G_2, \ldots, G_n$ do not overlap each other.) However, this example showed that quantum **while**-programs no longer possess such a determinism because probabilism is introduced by the measurements in the statements "**if** $(\square m \cdot M[\overline{q}] = m \to S_m)$ **fi**" and "**while** $M[\overline{q}] = 1$ **do** S **od**". Essentially, the operational semantics $\to$ of quantum programs given in Definition 3.2.2 is a probabilistic transition relation. However, the after encoding probabilities into partial density operators, probabilism manifests as nondeterminism in the transition rules (IF), (L0) and (L2). Therefore, the semantics $\to$ should be understood as a nondeterministic transition relation.

3.3 DENOTATIONAL SEMANTICS

We defined the operational semantics of quantum programs in the previous section. Then the denotational semantics can be defined based on it, or more precisely on the notion of computation introduced in Definition 3.2.3. The denotational semantics of a quantum program is a semantic function that maps partial density operators to themselves. Intuitively, for any quantum program S, the semantic function of S sums up the computed results of all terminating computations of S.

If configuration $\langle S', \rho'\rangle$ can be reached from $\langle S, \rho\rangle$ in n steps through the transition relation $\to$, which means there are configurations $\langle S_1, \rho_1\rangle, \ldots, \langle S_{n-1}, \rho_{n-1}\rangle$ such that

$$\langle S, \rho\rangle \to \langle S_1, \rho_1\rangle \to \ldots \to \langle S_{n-1}, \rho_{n-1}\rangle \to \langle S', \rho'\rangle,$$

then we write:

$$\langle S, \rho\rangle \to^n \langle S', \rho'\rangle.$$

Furthermore, we write $\to^*$ for the reflexive and transitive closures of $\to$; that is,

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

if and only if $\langle S, \rho\rangle \to^n \langle S', \rho'\rangle$ for some $n \geq 0$.

Definition 3.3.1. *Let S be a quantum program. Then its semantic function*

$$[\![S]\!] : \mathcal{D}(\mathcal{H}_{all}) \to \mathcal{D}(\mathcal{H}_{all})$$

is defined by

$$[\![S]\!](\rho) = \sum \left\{ |\rho' : \langle S, \rho\rangle \to^* \langle E, \rho'\rangle| \right\} \tag{3.7}$$

for all $\rho \in \mathcal{D}(\mathcal{H}_{all})$, where $\{|\cdot|\}$ stands for multi-set, which means a (generalized) set that allows multiple instances of an element.

The reason for using a multi-set rather than an ordinary set in equation (3.7) is that the same partial density operator may be obtained through different computational paths, as we can see from the rules (IF), (L0) and (L1) in the last section. The following simple example illustrates the case more explicitly.

Example 3.3.1. *Assume that type(q) = **Boolean**. Consider the program:*

$$\begin{aligned} S \equiv q := |0\rangle;\ q := H[q];\ &\textbf{if}\ M[q] = 0 \to S_0 \\ &\square \qquad\quad 1 \to S_1 \\ &\textbf{fi} \end{aligned}$$

where:

- *M is the measurement in the computational basis $|0\rangle, |1\rangle$ of the state space $\mathcal{H}_2$ of a qubit;*
- *$S_0 \equiv q := I[q]$ and $S_1 \equiv q := X[q]$ with I and X being the identity operator and the NOT gate, respectively.*

Let $\rho = |0\rangle_{all}\langle 0|$, where

$$|0\rangle_{all} = \bigotimes_{q \in qVar} |0\rangle_q.$$

Then the computations of S starting in ρ are:

$$\begin{aligned} \langle S, \rho\rangle &\to \langle q := H[q]; \textbf{if} \ldots \textbf{fi}, \rho\rangle \\ &\to \langle \textbf{if} \ldots \textbf{fi}, |+\rangle_q\langle +| \otimes \bigotimes_{p \neq q} |0\rangle_p\langle 0|\rangle \\ &\to \begin{cases} \langle S_0, \frac{1}{2}|0\rangle_q\langle 0| \otimes \bigotimes_{p\neq q} |0\rangle_p\langle 0|\rangle \to \langle E, \frac{1}{2}\rho\rangle, \\ \langle S_1, \frac{1}{2}|1\rangle_q\langle 1| \otimes \bigotimes_{p\neq q} |0\rangle_p\langle 0|\rangle \to \langle E, \frac{1}{2}\rho\rangle. \end{cases} \end{aligned}$$

So, we have:

$$[\![S]\!](\rho) = \frac{1}{2}\rho + \frac{1}{2}\rho = \rho.$$

3.3.1 BASIC PROPERTIES OF SEMANTIC FUNCTIONS

As in classical programming theory, operational semantics is convenient for describing the execution of quantum programs. On the other hand, denotational semantics is suitable for studying mathematical properties of quantum programs. Now we establish several basic properties of semantic functions that are useful for reasoning about quantum programs.

First of all, we observe that the semantic function of any quantum program is linear.

Lemma 3.3.1 (Linearity). *Let $\rho_1, \rho_2 \in \mathcal{D}(\mathcal{H}_{all})$ and $\lambda_1, \lambda_2 \geq 0$. If $\lambda_1\rho_1 + \lambda_2\rho_2 \in \mathcal{D}(\mathcal{H}_{all})$, then for any quantum program S, we have:*

$$[\![S]\!](\lambda_1\rho_1 + \lambda_2\rho_2) = \lambda_1[\![S]\!](\rho_1) + \lambda_2[\![S]\!](\rho_2).$$

Proof. We can prove the following fact by induction on the structure of S:

- Claim: If $\langle S, \rho_1\rangle \to \langle S', \rho_1'\rangle$ and $\langle S, \rho_2\rangle \to \langle S', \rho_2'\rangle$, then

$$\langle S, \lambda_1\rho_1 + \lambda_2\rho_2\rangle \to \langle S', \lambda_1\rho_1' + \lambda_2\rho_2'\rangle.$$

Then the conclusion immediately follows. □

Exercise 3.3.1. *Prove the claim in the proof of Lemma 3.3.1.*

Secondly, we present a structural representation for the semantic functions of quantum programs except **while**-loops. The representation of the semantic function of a quantum loop requires some mathematical tools from lattice theory. So, it is postponed to Subsection 3.3.3 after we prepare the necessary tools in the next subsection.

Proposition 3.3.1 (Structural Representation).

(i) $[\![\mathbf{skip}]\!](\rho) = \rho$.

(ii) **(a)** *If type*$(q) =$ **Boolean**, *then*

$$[\![q := |0\rangle]\!](\rho) = |0\rangle_q\langle 0|\rho|0\rangle_q\langle 0| + |0\rangle_q\langle 1|\rho|1\rangle_q\langle 0|.$$

(b) *If type*$(q) =$ **integer**, *then*

$$[\![q := |0\rangle]\!](\rho) = \sum_{n=-\infty}^{\infty} |0\rangle_q\langle n|\rho|n\rangle_q\langle 0|.$$

(iii) $[\![\overline{q} := U[\overline{q}]]\!](\rho) = U\rho U^\dagger$.

(iv) $[\![S_1; S_2]\!](\rho) = [\![S_2]\!]([\![S_1]\!](\rho))$.

(v) $[\![\mathbf{if}\ (\square m \cdot M[\overline{q}] = m \to S_m)\ \mathbf{fi}]\!](\rho) = \sum_m [\![S_m]\!](M_m\rho M_m^\dagger)$.

Proof. (i), (ii) and (iii) are obvious.
(iv) By Lemma 3.3.1 and the rule (SC) we obtain:

$$\begin{aligned}
[\![S_2]\!]([\![S_1]\!](\rho)) &= [\![S_2]\!]\left(\sum\{|\rho_1 : \langle S_1,\rho\rangle \to^* \langle E,\rho_1\rangle|\}\right)\\
&= \sum\{|[\![S_2]\!](\rho_1) : \langle S_1,\rho\rangle \to^* \langle E,\rho_1\rangle|\}\\
&= \sum\left\{|\sum\{|\rho' : \langle S_2,\rho_1\rangle \to^* \langle E,\rho'\rangle|\} : \langle S_1,\rho\rangle \to^* \langle E,\rho_1\rangle|\right\}\\
&= \sum\{|\rho' : \langle S_1,\rho\rangle \to^* \langle E,\rho_1\rangle \text{ and } \langle S_2,\rho_1\rangle \to^* \langle E,\rho'\rangle|\}\\
&= \sum\{|\rho' : \langle S_1;S_2,\rho\rangle \to^* \langle E,\rho'\rangle|\}\\
&= [\![S_1;S_2]\!](\rho).
\end{aligned}$$

(v) follows immediately from the rule (IF). □

3.3.2 QUANTUM DOMAINS

Before presenting a representation of the semantic function of a quantum **while**-loop, we first have to pave the road leading toward it. In this subsection, we examine the domains of partial density operators and quantum operations. The notions and lemmas presented in this subsection will also be used in Section 3.4 and Chapter 7.

Basic Lattice Theory:

We first review the requisite concepts from lattice theory.

Definition 3.3.2. *A partial order is a pair $(L,\sqsubseteq)$ where L is a nonempty set and $\sqsubseteq$ is a binary relation on L satisfying the following conditions:*

- **(i)** *Reflexivity: $x \sqsubseteq x$ for all $x \in L$;*
- **(ii)** *Antisymmetry: $x \sqsubseteq y$ and $y \sqsubseteq x$ imply $x = y$ for all $x, y \in L$;*
- **(iii)** *Transitivity: $x \sqsubseteq y$ and $y \sqsubseteq z$ imply $x \sqsubseteq z$ for all $x, y, z \in L$.*

Definition 3.3.3. *Let $(L,\sqsubseteq)$ be a partial order.*

- **(i)** *An element $x \in L$ is called the least element of L when $x \sqsubseteq y$ for all $y \in L$. The least element is usually denoted by 0.*
- **(ii)** *An element $x \in L$ is called an upper bound of a subset $X \subseteq L$ if $y \sqsubseteq x$ for all $x \in X$.*
- **(iii)** *x is called the least upper bound of X, written $x = \bigsqcup X$, if*
  - **(a)** *x is an upper bound of X;*
  - **(b)** *for any upper bound y of X, it holds that $x \sqsubseteq y$.*

We often write $\bigsqcup_{n=0}^{\infty} x_n$ or $\bigsqcup_n x_n$ for $\bigsqcup X$ when X is a sequence $\{x_n\}_{n=0}^{\infty}$.

Definition 3.3.4. *A complete partial order (CPO for short) is a partial order $(L,\sqsubseteq)$ satisfying the following conditions:*

(i) *it has the least element* 0*;*
(ii) $\bigsqcup_{n=0}^{\infty} x_n$ *exists for any increasing sequence* $\{x_n\}$ *in* L*; i.e.*

$$x_0 \sqsubseteq \ldots \sqsubseteq x_n \sqsubseteq x_{n+1} \sqsubseteq \ldots.$$

Definition 3.3.5. *Let* $(L, \sqsubseteq)$ *be a CPO. Then a function* f *from* L *into itself is said to be continuous if*

$$f\left(\bigsqcup_n x_n\right) = \bigsqcup_n f(x_n)$$

for any increasing sequence $\{x_n\}$ *in* L*.*

The following theorem has been widely used in programming theory for the description of semantics of loops and recursive programs.

Theorem 3.3.1. *(Knaster-Tarski) Let* $(L, \sqsubseteq)$ *be a CPO and function* $f : L \to L$ *is continuous. Then* f *has the least fixed point*

$$\mu f = \bigsqcup_{n=0}^{\infty} f^{(n)}(0)$$

(i.e., $f(\mu f) = \mu f$*, and if* $f(x) = x$ *then* $\mu f \sqsubseteq x$*), where*

$$\begin{cases} f^{(0)}(0) & = 0, \\ f^{(n+1)}(0) & = f(f^{(n)}(0)) \text{ for } n \geq 0. \end{cases}$$

Exercise 3.3.2. *Prove Theorem 3.3.1.*

Domain of Partial Density Operators:

We now consider the lattice-theoretic structures of quantum objects required in the representation of quantum **while**-loops. Actually, we need to deal with two levels of quantum objects. At the lower level are partial density operators. Let $\mathcal{H}$ be an arbitrary Hilbert space. A partial order in the set $\mathcal{D}(\mathcal{H})$ of partial density operators was already introduced in Definition 2.1.13. Recall that the Löwner order is defined as follows: for any operators $A, B \in \mathcal{L}(\mathcal{H})$, $A \sqsubseteq B$ if $B - A$ is a positive operator. The lattice-theoretic property of $\mathcal{D}(\mathcal{H})$ equipped with the Löwner order $\sqsubseteq$ is revealed in the following:

Lemma 3.3.2. $(\mathcal{D}(\mathcal{H}), \sqsubseteq)$ *is a CPO with the zero operator* $0_{\mathcal{H}}$ *as its least element.*

Domain of Quantum Operations:

We further consider the lattice-theoretic structure of quantum operations (see Definition 2.1.25).

Lemma 3.3.3. *Each quantum operation in a Hilbert space* $\mathcal{H}$ *is a continuous function from* $(\mathcal{D}(\mathcal{H}), \sqsubseteq)$ *into itself.*

We write $\mathcal{QO}(\mathcal{H})$ for the set of quantum operations in Hilbert space $\mathcal{H}$. Quantum operations should be considered as a class of quantum objects at a higher lever than partial density operators because $\mathcal{D}(\mathcal{H}) \subseteq \mathcal{L}(\mathcal{H})$, whereas $\mathcal{QO}(\mathcal{H}) \subseteq \mathcal{L}(\mathcal{L}(\mathcal{H}))$.

The Löwner order between operators induces a partial order between quantum operations in a natural way: for any $\mathcal{E}, \mathcal{F} \in \mathcal{QO}(\mathcal{H})$,

- $\mathcal{E} \sqsubseteq \mathcal{F} \Leftrightarrow \mathcal{E}(\rho) \sqsubseteq \mathcal{F}(\rho)$ for all $\rho \in \mathcal{D}(\mathcal{H})$.

In a sense, the Löwner order is lifted from lower-level objects $\mathcal{D}(\mathcal{H})$ to higher-level objects $\mathcal{QO}(\mathcal{H})$.

Lemma 3.3.4. *$(\mathcal{QO}(\mathcal{H}), \sqsubseteq)$ is a CPO.*

The proofs of Lemmas 3.3.2, 3.3.3 and 3.3.4 are quite involved. For readability, they are all postponed to Section 3.6.

3.3.3 SEMANTIC FUNCTION OF LOOP

Now we are ready to show that the semantic function of a quantum **while**-loop can be represented as the limit of the semantic functions of its finite syntactic approximations. To do this, we need an auxiliary notation: **abort** denotes a quantum program such that

$$[\![\textbf{abort}]\!](\rho) = 0_{\mathcal{H}_{\text{all}}}$$

for all $\rho \in \mathcal{D}(\mathcal{H})$. Intuitively, program **abort** is never guaranteed to terminate; for example, we can choose

$$\textbf{abort} \equiv \textbf{while}\ M_{\text{trivial}}[q] = 1\ \textbf{do skip od},$$

where q is a quantum variable, and $M_{\text{trivial}} = \{M_0 = 0_{\mathcal{H}_q}, M_1 = I_{\mathcal{H}_q}\}$ is a trivial measurement in the state space $\mathcal{H}_q$. The program **abort** will serve as the basis for inductively defining the syntactic approximations of a quantum loop.

Definition 3.3.6. *Consider a quantum loop*

$$\textbf{while} \equiv \textbf{while}\ M[\overline{q}] = 1\ \textbf{do}\ S\ \textbf{od}. \tag{3.8}$$

For any integer $k \geq 0$, the kth syntactic approximation $\textbf{while}^{(k)}$ *of* **while** *is inductively defined by*

$$\begin{cases} \textbf{while}^{(0)} & \equiv \textbf{abort}, \\ \textbf{while}^{(k+1)} & \equiv \textbf{if}\ M[\overline{q}] = 0 \rightarrow \textbf{skip} \\ & \quad \square \quad\ 1 \rightarrow S; \textbf{while}^{(k)} \\ & \quad \textbf{fi} \end{cases}$$

A representation of the semantic function of a quantum **while**-loop is then presented in the following:

Proposition 3.3.2. *Let* **while** *be the loop* (3.8)*. Then*

$$[\![\textbf{while}]\!] = \bigsqcup_{k=0}^{\infty} \left[\!\left[\textbf{while}^{(k)}\right]\!\right],$$

where $\mathbf{while}^{(k)}$ *is the kth syntactic approximation of* **while** *for every* $k \geq 0$*, and the symbol* $\bigsqcup$ *stands for the supremum of quantum operations: i.e., the least upper bound in CPO* $(\mathcal{QO}(\mathcal{H}_{all}), \sqsubseteq)$.

Proof. For $i = 0, 1$, we introduce auxiliary operators

$$\mathcal{E}_i : \mathcal{D}(\mathcal{H}_{all}) \to \mathcal{D}(\mathcal{H}_{all})$$

defined by $\mathcal{E}_i(\rho) = M_i \rho M_i^\dagger$ for all $\rho \in \mathcal{D}(\mathcal{H})$.

First, we prove:

$$\left[\!\left[\mathbf{while}^{(k)}\right]\!\right](\rho) = \sum_{n=0}^{k-1} \left[\mathcal{E}_0 \circ \left([\![S]\!] \circ \mathcal{E}_1\right)^n\right](\rho)$$

for all $k \geq 1$ by induction on k. The symbol $\circ$ in the preceding equality stands for composition of quantum operations; that is, the composition $\mathcal{F} \circ \mathcal{E}$ of quantum operations $\mathcal{E}$ and $\mathcal{F}$ is defined by $(\mathcal{F} \circ \mathcal{E})(\rho) = \mathcal{F}(\mathcal{E}(\rho))$ for every $\rho \in \mathcal{D}(\mathcal{H})$. The case of $k = 1$ is obvious. Then by Proposition 3.3.1 (i), (iv) and (v) and the induction hypothesis on $k - 1$ we obtain:

$$\begin{aligned}
\left[\!\left[\mathbf{while}^{(k)}\right]\!\right](\rho) &= [\![\mathbf{skip}]\!](\mathcal{E}_0(\rho)) + \left[\!\left[S; \mathbf{while}^{(k-1)}\right]\!\right](\mathcal{E}_1(\rho)) \\
&= \mathcal{E}_0(\rho) + \left[\!\left[\mathbf{while}^{(k-1)}\right]\!\right]((\![S]\!] \circ \mathcal{E}_1)(\rho)) \\
&= \mathcal{E}_0(\rho) + \sum_{n=0}^{k-2} \left[\mathcal{E}_0 \circ ([\![S]\!] \circ \mathcal{E}_1)^n\right]((\![S]\!] \circ \mathcal{E}_1)(\rho)) \qquad (3.9) \\
&= \sum_{n=0}^{k-1} \left[\mathcal{E}_0 \circ ([\![S]\!] \circ \mathcal{E}_1)^n\right](\rho).
\end{aligned}$$

Secondly, we have:

$$\begin{aligned}
[\![\mathbf{while}]\!](\rho) &= \sum \left\{ |\rho' : \langle \mathbf{while}, \rho\rangle \to^* \langle E, \rho'\rangle| \right\} \\
&= \sum_{k=1}^{\infty} \sum \left\{ |\rho' : \langle \mathbf{while}, \rho\rangle \to^k \langle E, \rho'\rangle| \right\}.
\end{aligned}$$

So, it suffices to show that

$$\sum \left\{ |\rho' : \langle \mathbf{while}, \rho\rangle \to^k \langle E, \rho'\rangle| \right\} = \left[\mathcal{E}_0 \circ ([\![S]\!] \circ \mathcal{E}_1)^{k-1}\right](\rho)$$

for all $k \geq 1$. By the previous points, it is not hard to prove this equality by induction on k. □

A fixed point characterization of the semantic function of a quantum loop can be derived from the preceding proposition.

Corollary 3.3.1. *Let* **while** *be the loop* (3.8)*. Then for any* $\rho \in \mathcal{D}(\mathcal{H}_{all})$*, it holds that*

$$[\![\textbf{while}]\!](\rho) = M_0 \rho M_0^\dagger + [\![\textbf{while}]\!]\left([\![S]\!]\left(M_1 \rho M_1^\dagger\right)\right).$$

Proof. Immediate from Proposition 3.3.2 and equation (3.9). □

3.3.4 CHANGE AND ACCESS OF QUANTUM VARIABLES

One key issue in understanding the behavior of a program is to observe how it changes the states of program variables and how it accesses program variables during its execution. As the first application of the semantic function just studied, we now address this issue for quantum programs.

To simplify the presentation, we introduce an abbreviation. Let $X \subseteq qVar$ be a set of quantum variables. For any operator $A \in \mathcal{L}(\mathcal{H}_{all})$, we write:

$$tr_X(A) = tr_{\bigotimes_{q\in X} \mathcal{H}_q}(A)$$

where $tr_{\bigotimes_{q\in X} \mathcal{H}_q}$ is the partial trace over system $\bigotimes_{q\in X} \mathcal{H}_q$ (see Definition 2.1.22). Then we have:

Proposition 3.3.3

(i) $tr_{qvar(S)}([\![S]\!](\rho)) = tr_{qvar(S)}(\rho)$ *whenever* $tr([\![S]\!](\rho)) = tr(\rho)$.
(ii) *If it holds that*

$$tr_{qVar\setminus qvar(S)}(\rho_1) = tr_{qVar\setminus qvar(S)}(\rho_2),$$

then we have:

$$tr_{qVar\setminus qvar(S)}([\![S]\!](\rho_1)) = tr_{qVar\setminus qvar(S)}([\![S]\!](\rho_2)).$$

Recall from Definition 2.1.22 that $tr_X(\rho)$ describes the state of the quantum variables not in X when the global state of all quantum variables is ρ. So, the preceding proposition can be intuitively explained as follows:

- Proposition 3.3.3(i) indicates that the state of the quantum variables not in $qvar(S)$ after executing program S is the same as that before executing S. This means that program S can only change the state of quantum variables in $qvar(S)$.
- Proposition 3.3.3(ii) shows that if two input states ρ_1 and ρ_2 coincide on the quantum variables in $qvar(S)$, then the computed outcomes of S, starting in ρ_1 and ρ_2, respectively, will also coincide on these quantum variables. In other words, if the output of program S with input ρ_1 is different from that with input ρ_2, then ρ_1 and ρ_2 must be different when restricted to $qvar(S)$. This means that program S can access at most the quantum variables in $qvar(S)$.

Exercise 3.3.3. *Prove Proposition 3.3.3. [Hint: use the representation of semantic functions presented in Propositions 3.3.1 and 3.3.2.]*

3.3.5 TERMINATION AND DIVERGENCE PROBABILITIES

Another key issue with the behavior of a program is its termination. The first consideration about this problem for quantum programs is based on the following proposition showing that a semantic function does not increase the trace of partial density operator of quantum variables.

Proposition 3.3.4. *For any quantum program S and for all partial density operators $\rho \in \mathcal{D}(\mathcal{H}_{all})$, it holds that*

$$tr([\![S]\!](\rho)) \leq tr(\rho).$$

Proof. We proceed by induction on the structure of S.

- Case 1. $S \equiv$ **skip**. Obvious.
- Case 2. $S \equiv q := |0\rangle$. If $type(q) =$ **integer**, then using the equality $tr(AB) = tr(BA)$ we obtain:

$$\begin{aligned} tr([\![S]\!](\rho)) &= \sum_{n=-\infty}^{\infty} tr(|0\rangle_q\langle n|\rho|n\rangle_q\langle 0|) \\ &= \sum_{n=-\infty}^{\infty} tr({}_q\langle 0|0\rangle_q\langle n|\rho|n\rangle_q) \\ &= tr\left[\left(\sum_{n=-\infty}^{\infty} |n\rangle_q\langle n|\right)\rho\right] = tr(\rho). \end{aligned}$$

  It can be proved in a similar way when $type(q) =$ **Boolean**.
- Case 3. $S \equiv \overline{q} := U[\overline{q}]$. Then

$$tr([\![S]\!](\rho)) = tr\left(U\rho U^\dagger\right) = tr\left(U^\dagger U\rho\right) = tr(\rho).$$

- Case 4. $S \equiv S_1; S_2$. It follows from the induction hypothesis on S_1 and S_2 that

$$\begin{aligned} tr([\![S]\!](\rho)) &= tr([\![S_2]\!]([\![S_1]\!](\rho))) \\ &\leq tr([\![S_1]\!](\rho)) \\ &\leq tr(\rho). \end{aligned}$$

- Case 5. $S \equiv$ **if** $(\square m \cdot M[\overline{q}] = m \rightarrow S_m)$ **fi**. Then by the induction hypothesis we obtain:

$$\begin{aligned} tr([\![S]\!](\rho)) &= \sum_m tr\left([\![S_m]\!]\left(M_m\rho M_m^\dagger\right)\right) \\ &\leq \sum_m tr\left(M_m\rho M_m^\dagger\right) \\ &= tr\left[\left(\sum_m M_m^\dagger M_m\right)\rho\right] = tr(\rho). \end{aligned}$$

- Case 6. $S \equiv$ **while** $M[\overline{q}] = 1$ **do** S' **od**. We write $(\textbf{while}')^n$ for the statement obtained through replacing S by S' in $(\textbf{while})^n$ given in Definition 3.3.6. With Proposition 3.3.2, it suffices to show that

$$tr\left(\left[\!\left[(\textbf{while}')^n\right]\!\right](\rho)\right) \leq tr(\rho)$$

for all $n \geq 0$. This can be carried out by induction on n. The case of $n = 0$ is obvious. By the induction hypothesis on n and S', we have:

$$\begin{aligned} tr\left(\left[\!\left[(\textbf{while}')^{n+1}\right]\!\right](\rho)\right) &= tr\left(M_0\rho M_0^\dagger\right) + tr\left(\left[\!\left[(\textbf{while}')^n\right]\!\right]\left(\left[\!\left[S'\right]\!\right]\left(M_1\rho M_1^\dagger\right)\right)\right) \\ &\leq tr\left(M_0\rho M_0^\dagger\right) + tr\left(\left[\!\left[S'\right]\!\right]\left(M_1\rho M_1^\dagger\right)\right) \\ &\leq tr\left(M_0\rho M_0^\dagger\right) + tr\left(M_1\rho M_1^\dagger\right) \\ &= tr\left[\left(M_0^\dagger M_0 + M_1^\dagger M_1\right)\rho\right] \\ &= tr(\rho). \end{aligned}$$

□

Intuitively, $tr([\![S]\!](\rho))$ is the probability that program S terminates when starting in state ρ. From the proof of the preceding proposition, we can see that the only program constructs that can cause $tr([\![S]\!](\rho)) < tr(\rho)$ are the loops occurring in S. Thus,

$$tr(\rho) - tr([\![S]\!](\rho))$$

is the probability that program S diverges from input state ρ. This can be further illustrated by the following example.

Example 3.3.2. *Let* $type(q) =$ **integer**, *and let*

$$M_0 = \sum_{n=1}^{\infty} \sqrt{\frac{n-1}{2n}}(|n\rangle\langle n| + |-n\rangle\langle -n|),$$

$$M_1 = |0\rangle\langle 0| + \sum_{n=1}^{\infty} \sqrt{\frac{n+1}{2n}}(|n\rangle\langle n| + |-n\rangle\langle -n|).$$

Then $M = \{M_0, M_1\}$ *is a yes-no measurement in the state Hilbert space* $\mathcal{H}_q$ *(note that* M *is not a projective measurement). Consider the program:*

$$\textbf{while} \equiv \textbf{while}\ M[q] = 1\ \textbf{do}\ q := q + 1\ \textbf{od}$$

Let $\rho = |0\rangle_q\langle 0| \otimes \rho_0$ *with*

$$\rho_0 = \bigotimes_{p \neq q} |0\rangle_p\langle 0|.$$

Then after some calculations we have:

$$[\![(\textbf{while})^n]\!](\rho) = \begin{cases} 0_{\mathcal{H}_{all}} & \text{if } n = 0, 1, 2, \\ \frac{1}{2}\left(\sum_{k=2}^{n-1} \frac{k-1}{k!}|k\rangle_q\langle k|\right) \otimes \rho_0 & \text{if } n \geq 3, \end{cases}$$

$$[\![\textbf{while}]\!](\rho) = \frac{1}{2}\left(\sum_{n=2}^{\infty} \frac{n-1}{n!}|n\rangle_q\langle n|\right) \otimes \rho_0$$

and

$$tr([\![\textbf{while}]\!](\rho)) = \frac{1}{2}\sum_{n=2}^{\infty} \frac{n-1}{n!} = \frac{1}{2}.$$

This means that program **while** *terminates on input* ρ *with probability* $\frac{1}{2}$*, and it diverges from input* ρ *with probability* $\frac{1}{2}$.

A more systematic study of the termination of quantum programs will be presented in Chapter 5.

3.3.6 SEMANTIC FUNCTIONS AS QUANTUM OPERATIONS

To conclude this section, we establish a connection between quantum programs and quantum operations (see Subsection 2.1.7).

The semantic function of a quantum program is defined to be a mapping from partial density operators in $\mathcal{H}_{all}$ to themselves. Let V be a subset of $qVar$. Whenever a quantum operation $\mathcal{E}$ in $\mathcal{H}_{all}$ is the cylindric extension of a quantum operation $\mathcal{F}$ in $\mathcal{H}_V = \bigotimes_{q\in V} \mathcal{H}_q$, meaning that

$$\mathcal{E} = \mathcal{F} \otimes \mathcal{I}$$

where $\mathcal{I}$ is the identity quantum operation in $\mathcal{H}_{qVar\setminus V}$, we always identify $\mathcal{E}$ with $\mathcal{F}$, and $\mathcal{E}$ can be seen as a quantum operation in $\mathcal{H}_V$. With this convention, we have:

Proposition 3.3.5. *For any quantum program* S*, its semantic function* $[\![S]\!]$ *is a quantum operation in* $\mathcal{H}_{qvar(S)}$.

Proof. It can be proved by induction on the structure of S. For the case that S is not a loop, it follows from Theorem 2.1.1 (iii) and Proposition 3.3.1. For the case that S is a loop, it follows from Proposition 3.3.2 and Lemma 3.3.4. □

Conversely, one may ask: can every quantum operation be modelled by a quantum program? To answer this question, let us first introduce the notion of local quantum variable.

Definition 3.3.7. *Let* S *be a quantum program and* $\overline{q}$ *a sequence of quantum variables. Then:*

(i) *The block command defined by S with local variables $\overline{q}$ is:*

$$\textbf{begin local } \overline{q} : S \textbf{ end}. \tag{3.10}$$

(ii) *The quantum variables of the block command are:*

$$qvar\,(\textbf{begin local } \overline{q} : S \textbf{ end}) = qvar(S) \setminus \overline{q}.$$

(iii) *The denotational semantics of the block command is the quantum operation from $\mathcal{H}_{qvar(S)}$ to $\mathcal{H}_{qvar(S)\setminus\overline{q}}$ defined by*

$$[\![\textbf{begin local } \overline{q} : S \textbf{ end}]\!]\,(\rho) = tr_{\mathcal{H}_{\overline{q}}}([\![S]\!](\rho)) \tag{3.11}$$

for any density operator $\rho \in \mathcal{D}(\mathcal{H}_{qvar(S)})$, where $tr_{\mathcal{H}_{\overline{q}}}$ stands for the partial trace over $\mathcal{H}_{\overline{q}}$ (see Definition 2.1.22).

The intuitive meaning of block command (3.10) is that program S is running in the environment where $\overline{q}$ are local variables that will be initialized in S. After executing S, the auxiliary system denoted by the local variables $\overline{q}$ is discarded. This is why the trace over $\mathcal{H}_{\overline{q}}$ is taken in the defining equation (3.11) of the semantics of the block command. Note that (3.11) is a partial density operator in $\mathcal{H}_{qvar(S)\setminus\overline{q}}$.

A block command will be seen as a quantum program in the sequel. Then we are able to provide a positive answer to the question raised earlier. The following proposition is essentially a restatement of Theorem 2.1.1 (ii) in terms of quantum programs.

Proposition 3.3.6. *For any finite subset V of qVar, and for any quantum operation $\mathcal{E}$ in $\mathcal{H}_V$, there exists a quantum program (more precisely, a block command) S such that $[\![S]\!] = \mathcal{E}$.*

Proof. By Theorem 2.1.1 (ii), there exist:

(i) quantum variables $\overline{p} \subseteq qVar \setminus \overline{q}$,
(ii) a unitary transformation U in $\mathcal{H}_{\overline{p}\cup\overline{q}}$,
(iii) a projection P onto a closed subspace of $\mathcal{H}_{\overline{p}\cup\overline{q}}$, and
(iv) a state $|e_0\rangle$ in $\mathcal{H}_{\overline{p}}$

such that

$$\mathcal{E}(\rho) = tr_{\mathcal{H}_{\overline{p}}}\Big[PU(|e_0\rangle\langle e_0| \otimes \rho)U^\dagger P\Big]$$

for all $\rho \in \mathcal{D}(\mathcal{H}_{\overline{q}})$. Obviously, we can find a unitary operator U_0 in $\mathcal{H}_{\overline{p}}$ such that

$$|e_0\rangle = U_0|0\rangle_{\overline{p}}$$

where $|0\rangle_{\overline{p}} = |0\rangle \ldots |0\rangle$ (all quantum variables in $\overline{p}$ are initialized in state $|0\rangle$). On the other hand,

$$M = \{M_0 = P, M_1 = I - P\}$$

is a yes-no measurement in $\mathcal{H}_{\overline{p}\cup\overline{q}}$, where I is the identity operator in $\mathcal{H}_{\overline{p}\cup\overline{q}}$. We set

$$\begin{aligned} S \equiv \textbf{begin local } \overline{p} : \overline{p} := |0\rangle_{\overline{p}};\ \overline{p} := U_0[\overline{p}];\ \overline{p}\cup\overline{q} := U[\overline{p}\cup\overline{q}]; \\ \textbf{if } M[\overline{p}\cup\overline{q}] = 0 \rightarrow \textbf{skip} \\ \square \qquad\quad 1 \rightarrow \textbf{abort} \\ \textbf{fi} \\ \textbf{end} \end{aligned}$$

Then it is easy to check that $[\![S]\!] = \mathcal{E}$. □

3.4 CLASSICAL RECURSION IN QUANTUM PROGRAMMING

The notion of recursion allows the programming of repetitive tasks without a large number of similar steps to be specified individually. In the previous sections, we have studied the quantum extension of the **while**-language, which provides a program construct, namely the quantum **while**-loop, to implement a special kind of recursion – iteration – in quantum computation. The general form of recursive procedure has been widely used in classical programming. It renders a more powerful technique than iteration, in which a function can be defined, directly or indirectly, in terms of itself. In this section, we add the general notion of recursion into the quantum **while**-language in order to specify procedures in quantum computation that can call themselves.

The notion of recursion considered in this section should be properly termed as *classical recursion in quantum programming* because the control flow within it is still classical, or more precisely, the control is determined by the outcomes of quantum measurements. The notion of recursion with quantum control flow will be introduced in Chapter 7. To avoid confusion, a quantum program containing recursion with classical control will be called a *recursive quantum program*, whereas a quantum program containing recursion with quantum control will be referred to as a *quantum recursive program*. With the mathematical tools prepared in Subsection 3.3.2, the theory of recursive quantum programs presented in this section is more or less a straightforward generalization of the theory of classical recursive programs. However, as you will see in Chapter 7, the treatment of quantum recursive programs is much more difficult, and it requires some ideas that are radically different from those used in this section.

3.4.1 SYNTAX

We first define the syntax of recursive quantum programs. The alphabet of recursive quantum programs is the alphabet of quantum **while**-programs expanded by adding a set of procedure identifiers, ranged over by $X, X_1, X_2, \ldots$.

Quantum program schemes are defined as generalized quantum **while**-programs that may contain procedure identifiers. Formally, we have:

Definition 3.4.1. *Quantum program schemes are generated by the syntax:*

$$\begin{aligned} S ::= \; & X \mid \textbf{skip} \mid q := |0\rangle \mid \overline{q} := U[\overline{q}] \mid S_1; S_2 \\ & \mid \textbf{if } (\square m \cdot M[\overline{q}] = m \to S_m) \textbf{ fi} \\ & \mid \textbf{while } M[\overline{q}] = 1 \textbf{ do } S \textbf{ od}. \end{aligned} \tag{3.12}$$

The only difference between the syntax (3.2) and (3.12) is that a clause for procedure identifiers X is added in the latter. If a program scheme S contains at most the procedure identifiers $X_1, \ldots, X_n$, then we write

$$S \equiv S[X_1, \ldots, X_n].$$

As in classical programming, procedure identifiers in a quantum program scheme are used as subprograms, which are usually called *procedure calls*. They are specified by declarations defined in the following:

Definition 3.4.2. *Let $X_1, \ldots, X_n$ be different procedure identifiers. A declaration for $X_1, \ldots, X_n$ is a system of equations:*

$$D : \begin{cases} X_1 \Leftarrow S_1, \\ \quad \ldots\ldots \\ X_n \Leftarrow S_n, \end{cases} \tag{3.13}$$

where for every $1 \leq i \leq n$, $S_i \equiv S_i[X_1, \ldots, X_n]$ is a quantum program scheme.

Now we are ready to introduce the key notion of this section.

Definition 3.4.3. *A recursive quantum program consists of:*

(i) *a quantum program scheme $S \equiv S[X_1, \ldots, X_n]$, called the main statement; and*
(ii) *a declaration D for $X_1, \ldots, X_n$.*

3.4.2 OPERATIONAL SEMANTICS

A recursive quantum program is a quantum program scheme together with a declaration of procedure identifiers within it. So, we first define the operational semantics of quantum program schemes with respect to a given declaration. To this end, we need to generalize the notion of configuration defined in Section 3.2.

Definition 3.4.4. *A quantum configuration is a pair $\langle S, \rho \rangle$, where:*

(i) *S is a quantum program scheme or the empty program E;*
(ii) *$\rho \in \mathcal{D}(\mathcal{H}_{all})$ is a partial density operator in $\mathcal{H}_{all}$.*

This definition is the same as Definition 3.2.1 except that S is allowed to be not only a program but also a program scheme.

Now the operational semantics of quantum programs given in Definition 3.2.2 can be easily generalized to the case of quantum program schemes.

Definition 3.4.5. *Let D be a given declaration. The operational semantics of quantum program schemes with respect to D is the transition relation $\to_D$ between*

quantum configurations defined by the transition rules in Figure 3.1 together with the following rule for recursion:

$$\text{(REC)} \qquad \frac{}{\langle X_i, \rho\rangle \to_D \langle S_i, \rho\rangle} \text{ if } X_i \Leftarrow S_i \text{ is in the declaration } D.$$

FIGURE 3.3

Transition rule for recursive quantum programs.

Of course, when used in this definition, the rules in Figure 3.1 are extended by allowing program schemes to appear in configurations, and the transition symbol $\to$ is replaced by $\to_D$. As in classical programming, the rule (REC) in Figure 3.3 can be referred to as the *copy rule*, meaning that at runtime a procedure call is treated like the procedure body inserted at the place of call.

3.4.3 DENOTATIONAL SEMANTICS

Based on the operational semantics described in the last subsection, the denotational semantics of quantum program schemes can be easily defined by straightforward extending of Definitions 3.2.3 and 3.3.1.

Definition 3.4.6. *Let D be a given declaration. For any quantum program scheme S, its semantic function with respect to D is the mapping:*

$$[\![S|D]\!] : \mathcal{D}(\mathcal{H}_{all}) \to \mathcal{D}(\mathcal{H}_{all})$$

defined by

$$[\![S|D]\!](\rho) = \sum \{|\rho' : \langle S, \rho\rangle \to_D^* \langle E, \rho'\rangle|\}$$

for every $\rho \in \mathcal{D}(\mathcal{H}_{all})$, where $\to_D^*$ is the reflexive and transitive closure of $\to_D$.

Suppose that a recursive quantum program consists of the main statement S and declaration D. Then its denotational semantics is defined to be $[\![S|D]\!]$. Obviously, if S is a program (i.e., a program scheme that contains no procedure identifiers), then $[\![S|D]\!]$ does not depend on D and it coincides with Definition 3.3.1, and thus we can simply write $[\![S]\!]$ for $[\![S|D]\!]$.

Example 3.4.1. *Consider the declaration*

$$D : \begin{cases} X_1 \Leftarrow S_1, \\ X_2 \Leftarrow S_2 \end{cases}$$

where:

$$S_1 \equiv \textbf{if } M[q] = 0 \rightarrow q := H[q]; X_2 \;\square\; 1 \rightarrow \textbf{skip} \;\textbf{fi}$$

$$S_2 \equiv \textbf{if } N[q] = 0 \rightarrow q := Z[q]; X_1 \;\square\; 1 \rightarrow \textbf{skip} \;\textbf{fi}$$

q is a qubit variable, M is the measurement in the computational basis $|0\rangle, |1\rangle$ and N the measurement in the basis $|+\rangle, |-\rangle$; that is,

$$M = \{M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|\},$$

$$N = \{N_0 = |+\rangle\langle +|, N_1 = |-\rangle\langle -|\}.$$

Then the computations of recursive quantum program X_1 with declaration D starting in $\rho = |+\rangle\langle +|$ are:

$$\langle X_1, \rho\rangle \rightarrow_D \langle S_1, \rho\rangle \rightarrow_D \begin{cases} \langle q := H[q]; X_2, \frac{1}{2}|0\rangle\langle 0|\rangle \rightarrow_D \langle X_2, \frac{1}{2}\rho\rangle, \\ \langle \textbf{skip}, \frac{1}{2}|1\rangle\langle 1|\rangle \rightarrow_D \langle E, \frac{1}{2}|1\rangle\langle 1|\rangle \end{cases}$$

where:

$$\langle X_2, \frac{1}{2}\rho\rangle \rightarrow_D \langle S_2, \frac{1}{2}\rho\rangle \rightarrow_D \langle q := Z[q]; X_1, \frac{1}{2}\rho\rangle \rightarrow_D \langle X_1, \frac{1}{2}|-\rangle\langle -|\rangle \rightarrow_D \cdots.$$

and we have:

$$[\![X_1|D]\!](\rho) = \sum_{n=1}^{\infty} \frac{1}{2^n}|1\rangle\langle 1| = |1\rangle\langle 1|.$$

Before moving on to study various properties of general recursive programs, let us see how a quantum **while**-loop discussed in the previous sections can be treated as a special recursive quantum program. We consider the loop:

$$\textbf{while} \equiv \textbf{while } M[\overline{q}] = 1 \textbf{ do } S \textbf{ od}.$$

Here S is a quantum program (containing no procedure identifiers). Let X be a procedure identifier with the declaration D:

$$X \equiv \textbf{if } M[\overline{q}] = 0 \rightarrow \textbf{skip} \;\square\; 1 \rightarrow S; X \;\textbf{fi}$$

Then the quantum loop **while** is actually equivalent to the recursive quantum program with X as its main statement.

Exercise 3.4.1. *Show that* $[\![\textbf{while}]\!] = [\![X|D]\!]$.

Basic Properties of Semantic Functions of Recursive Quantum Programs:

We now establish some basic properties of semantic functions of recursive quantum programs. The next proposition is a generalization of Propositions 3.3.1 and 3.3.2 to quantum program schemes with respect to a declaration.

Proposition 3.4.1. *Let D be the declaration given by equation (3.13). Then for any $\rho \in \mathcal{D}(\mathcal{H}_{all})$, we have:*

(i) $[\![X|D]\!](\rho) = \begin{cases} 0_{\mathcal{H}_{all}} & \text{if } X \notin \{X_1, \ldots, X_n\}, \\ [\![S_i|D]\!](\rho) & \text{if } X = X_i \ (1 \le i \le n); \end{cases}$

(ii) *if S is* **skip**, *initialization or unitary transformation, then* $[\![S|D]\!](\rho) = [\![S]\!](\rho)$;

(iii) $[\![T_1; T_2|D]\!](\rho) = [\![T_2|D]\!]([\![T_1|D]\!](\rho))$;

(iv) $[\![\textbf{if } (\square m \cdot M[\overline{q}] = m \to T_m) \textbf{ fi}|D]\!](\rho) = \sum_m [\![T_m|D]\!] \left(M_m \rho M_m^\dagger\right)$;

(v) $[\![\textbf{while } M[\overline{q}] = 1 \textbf{ do } S \textbf{ od}|D]\!](\rho) = \bigsqcup_{k=0}^{\infty} \left[\!\left[\textbf{while}^{(k)}|D\right]\!\right](\rho)$, *where* $\textbf{while}^{(k)}$ *is the kth syntactic approximation of the loop (see Definition 3.3.6) for every integer $k \ge 0$.*

Proof. Similar to the proofs of Propositions 3.3.1 and 3.3.2. □

Proposition 3.4.1(v) can be further generalized so that the denotational semantics of a recursive quantum program can be expressed in terms of that of its syntactic approximations.

Definition 3.4.7. *Consider the recursive quantum program with the main statement $S \equiv S[X_1, \ldots, X_m]$ and the declaration D given by equation (3.13). For any integer $k \ge 0$, the kth syntactic approximation $S_D^{(k)}$ of S with respect to D is inductively defined as follows:*

$$\begin{cases} S_D^{(0)} & \equiv \textbf{abort}, \\ S_D^{(k+1)} & \equiv S\left[S_{1D}^{(k)}/X_1, \ldots, S_{nD}^{(k)}/X_n\right], \end{cases} \tag{3.14}$$

where **abort** *is as in Subsection 3.3.3, and*

$$S[P_1/X_1, \ldots, P_n/X_n]$$

stands for the result of simultaneous substitution of $X_1, \ldots, X_n$ by $P_1, \ldots, P_n$, respectively, in S.

It should be noticed that the preceding definition is given by induction on k with S being an arbitrary program scheme. Thus, $S_{1D}^{(k)}, \ldots, S_{nD}^{(k)}$ in equation (3.14) are assumed to be already defined in the induction hypothesis for k. It is clear that for all $k \ge 0$, $S_D^{(k)}$ is a program (containing no procedure identifiers). The following lemma clarifies the relationship of a substitution and a declaration when used to define the semantics of a program.

Lemma 3.4.1. *Let D be a declaration given by equation (3.13). Then for any program scheme S, we have:*

(i) $[\![S|D]\!] = [\![S\,[S_1/X_1, \ldots, S_n/X_n]\,|D]\!]$.

(ii) $\left[\!\left[S_D^{(k+1)}\right]\!\right] = [\![S|D^{(k)}]\!]$ *for every integer $k \geq 0$, where declaration*

$$D^{(k)} = \begin{cases} X_1 \Leftarrow S_{1D}^{(k)}, \\ \quad \ldots\ldots \\ X_n \Leftarrow S_{nD}^{(k)}. \end{cases}$$

Proof.

(i) can be proved by induction on the structure of S together with Proposition 3.4.1.

(ii) It follows from (i) that

$$\begin{aligned} \left[\!\left[S|D^{(k)}\right]\!\right] &= \left[\!\left[S\left[S_{1D}^{(k)}/X_1, \ldots, S_{nD}^{(k)}/X_n\right]|D^{(k)}\right]\!\right] \\ &= \left[\!\left[S\left[S_{1D}^{(k)}/X_1, \ldots, S_{nD}^{(k)}/X_n\right]\right]\!\right] \\ &= \left[\!\left[S_D^{(k+1)}\right]\!\right]. \end{aligned}$$

□

Based on the previous lemma, we obtain a representation of the semantic function of a recursive program by its syntactic approximations.

Proposition 3.4.2. *For any recursive program S with declaration D, we have:*

$$[\![S|D]\!] = \bigsqcup_{k=0}^{\infty} \left[\!\left[S_D^{(k)}\right]\!\right].$$

Proof. For any $\rho \in \mathcal{D}(\mathcal{H}_{all})$, we want to show that

$$[\![S|D]\!](\rho) = \bigsqcup_{k=0}^{\infty} \left[\!\left[S_D^{(k)}\right]\!\right](\rho).$$

It suffices to prove that for any integers $r, k \geq 0$, the following claims hold:

Claim 1: $\langle S, \rho\rangle \to_D^r \langle E, \rho'\rangle \Rightarrow \exists l \geq 0$ s.t. $\langle S_D^{(l)}, \rho\rangle \to^* \langle E, \rho'\rangle$.

Claim 2: $\langle S_D^{(k)}, \rho\rangle \to^r \langle E, \rho'\rangle \Rightarrow \langle S, \rho\rangle \to_D^* \langle E, \rho'\rangle$.

This can be done by induction on r, k and the depth of inference using the transition rules. □

Exercise 3.4.2. *Complete the proof of Proposition 3.4.2.*

3.4.4 FIXED POINT CHARACTERIZATION

Proposition 3.4.2 can be seen as a generalization of Proposition 3.3.2 via Proposition 3.4.1(v). A fixed point characterization of quantum **while**-loop was given in Subsection 3.3.3 as a corollary of Proposition 3.3.2. In this section, we give a

fixed point characterization of recursive quantum programs and thus obtain a generalization of Corollary 3.3.1. In classical programming theory, recursive equations are solved in a certain domain of functions. Here, we are going to solve recursive quantum equations in the domain of quantum operations defined in Subsection 3.3.2. To this end, we first introduce the following:

Definition 3.4.8. *Let $S \equiv S[X_1, \ldots, X_n]$ be a quantum program scheme, and let $\mathcal{QO}(\mathcal{H}_{all})$ be the set of quantum operations in $\mathcal{H}_{all}$. Then its semantic functional is the mapping:*

$$[\![S]\!] : \mathcal{QO}(\mathcal{H}_{all})^n \to \mathcal{QO}(\mathcal{H}_{all})$$

defined as follows: for any $\mathcal{E}_1, \ldots, \mathcal{E}_n \in \mathcal{QO}(\mathcal{H}_{all})$,

$$[\![S]\!](\mathcal{E}_1, \ldots, \mathcal{E}_n) = [\![S|E]\!]$$

where

$$E : \begin{cases} X_1 \Leftarrow T_1, \\ \quad \ldots\ldots \\ X_n \Leftarrow T_n \end{cases}$$

is a declaration such that for each $1 \leq i \leq n$, T_i is a program (containing no procedure identifiers) with $[\![T_i]\!] = \mathcal{E}_i$.

We argue that the semantic functional $[\![S]\!]$ is well-defined. It follows from Proposition 3.3.6 that the programs T_i always exist. On the other hand, if

$$E' : \begin{cases} X_1 \Leftarrow T'_1, \\ \quad \ldots\ldots \\ X_n \Leftarrow T'_n \end{cases}$$

is another declaration with each program T'_i satisfying $[\![T'_i]\!] = \mathcal{E}_i$, then we can show that

$$[\![S|E]\!] = [\![S|E']\!].$$

Now we define the domain in which we are going to find a fixed point of the semantic functional defined by a declaration for procedure identifiers $X_1, \ldots, X_n$. Let us consider the Cartesian power $\mathcal{QO}(\mathcal{H}_{all})^n$. An order $\sqsubseteq$ in $\mathcal{QO}(\mathcal{H}_{all})^n$ is naturally induced by the order $\sqsubseteq$ in $\mathcal{QO}(\mathcal{H}_{all})$: for any $\mathcal{E}_1, \ldots, \mathcal{E}_n, \mathcal{F}_1, \ldots, \mathcal{F}_n \in \mathcal{QO}(\mathcal{H}_{all})$,

- $(\mathcal{E}_1, \ldots, \mathcal{E}_n) \sqsubseteq (\mathcal{F}_1, \ldots, \mathcal{F}_n) \Leftrightarrow$ for every $1 \leq i \leq n$, $\mathcal{E}_i \sqsubseteq \mathcal{F}_i$.

It follows from Lemma 3.3.4 that $(\mathcal{QO}(\mathcal{H}_{all})^n, \sqsubseteq)$ is a CPO. Furthermore, we have:

Proposition 3.4.3. *For any quantum program scheme $S \equiv S[X_1, \ldots, X_n]$, its semantic functional*

$$[\![S]\!] : \left(\mathcal{QO}(\mathcal{H}_{all})^n, \sqsubseteq\right) \to \left(\mathcal{QO}(\mathcal{H}_{all}), \sqsubseteq\right)$$

is continuous.

Proof. For each $1 \leq i \leq n$, let $\{\mathcal{E}_{ij}\}_j$ be an increasing sequence in $(\mathcal{QO}(\mathcal{H}_{all}), \sqsubseteq)$. What we need to prove is:

$$[\![S]\!]\left(\bigsqcup_j \mathcal{E}_{1j}, \ldots, \bigsqcup_j \mathcal{E}_{nj}\right) = \bigsqcup_j [\![S]\!]\left(\mathcal{E}_{1j}, \ldots, \mathcal{E}_{nj}\right).$$

Suppose that

$$D : \begin{cases} X_1 \Leftarrow P_1, \\ \quad \ldots\ldots \\ X_n \Leftarrow P_n, \end{cases} \qquad D_j : \begin{cases} X_1 \Leftarrow P_{1j}, \\ \quad \ldots\ldots \\ X_n \Leftarrow P_{nj} \end{cases}$$

be declarations such that

$$[\![P_i]\!] = \bigsqcup_j \mathcal{E}_{ij} \text{ and } [\![P_{ij}]\!] = \mathcal{E}_{ij}$$

for any $1 \leq i \leq n$ and for any j. Then it suffices to show that

$$[\![S|D]\!] = \bigsqcup_j [\![S|D_j]\!]. \tag{3.15}$$

Using Proposition 3.4.1, this can be done by induction on the structure of S. □

Exercise 3.4.3. *Prove equation (3.15).*

Let D be the declaration given by equation (3.13). Then D naturally induces a semantic functional:

$$[\![D]\!] : \mathcal{QO}(\mathcal{H}_{all})^n \to \mathcal{QO}(\mathcal{H}_{all})^n,$$

$$[\![D]\!](\mathcal{E}_1, \ldots, \mathcal{E}_n) = ([\![S_1]\!](\mathcal{E}_1, \ldots, \mathcal{E}_n), \ldots, [\![S_n]\!](\mathcal{E}_1, \ldots, \mathcal{E}_n))$$

for any $\mathcal{E}_1, \ldots, \mathcal{E}_n \in \mathcal{QO}(\mathcal{H}_{all})$. It follows from Proposition 3.4.3 that

$$[\![D]\!] : \left(\mathcal{QO}(\mathcal{H}_{all})^n, \sqsubseteq\right) \to \left(\mathcal{QO}(\mathcal{H}_{all})^n, \sqsubseteq\right)$$

is continuous. Then by the Knaster-Tarski Theorem (Theorem 3.3.1) we assert that $[\![D]\!]$ has a fixed point:

$$\mu[\![D]\!] = \left(\mathcal{E}_1^*, \ldots, \mathcal{E}_n^*\right) \in \mathcal{QO}(\mathcal{H}_{all})^n.$$

We now are able to present the fixed-point characterization of recursive quantum programs:

Theorem 3.4.1. *For the recursive quantum program with the main statement S and the declaration D, we have:*

$$[\![S|D]\!] = [\![S]\!](\mu[\![D]\!]) = [\![S]\!](\mathcal{E}_1^*, \ldots, \mathcal{E}_n^*).$$

Proof. First of all, we claim that for any program scheme $T \equiv T[X_1, \ldots, X_n]$ and for any programs $T_1, \ldots, T_n$,

$$[\![T\,[T_1/X_1, \ldots, T_n/X_n]]\!] = [\![T]\!]([\![T_1]\!], \ldots, [\![T_n]\!]). \tag{3.16}$$

In fact, let us consider declaration

$$E : \begin{cases} X_1 \Leftarrow T_1, \\ \quad \ldots\ldots \\ X_n \Leftarrow T_n. \end{cases}$$

Then by Definition 3.4.8 and Lemma 3.4.1 (i) we obtain:

$$\begin{aligned} [\![T]\!]([\![T_1]\!], \ldots, [\![T_n]\!]) = [\![T|E]\!] &= [\![T[T_1/X_1, \ldots, T_n/X_n]|E]\!] \\ &= [\![T[T_1/X_1, \ldots, T_n/X_n]]\!] \end{aligned}$$

because $T_1, \ldots, T_n$ are all programs (without procedure identifiers).

Secondly, we define the iteration of $[\![D]\!]$, starting from the least element $\overline{\mathbf{0}} = (\mathbf{0}, \ldots, \mathbf{0})$ in $\mathcal{QC}(\mathcal{H}_{all})^n$, as follows:

$$\begin{cases} [\![D]\!]^{(0)}(\overline{\mathbf{0}}) = (\mathbf{0}, \ldots, \mathbf{0}), \\ [\![D]\!]^{(k+1)}(\overline{\mathbf{0}}) = [\![D]\!]\left([\![D]\!]^{(k)}(\overline{\mathbf{0}})\right) \end{cases}$$

where $\mathbf{0}$ is the zero quantum operation in $\mathcal{H}_{all}$. Then it holds that

$$[\![D]\!]^{(k)}\left(\overline{\mathbf{0}}\right) = \left(\left[\!\left[S_{1D}^{(k)}\right]\!\right], \ldots, \left[\!\left[S_{nD}^{(k)}\right]\!\right]\right) \tag{3.17}$$

for every integer $k \geq 0$. Equation (3.17) can by proved by induction on k. Indeed, the case of $k = 0$ is obvious. The induction hypothesis for k together with equation (3.16) yields:

$$\begin{aligned} &[\![D]\!]^{(k+1)}\left(\overline{\mathbf{0}}\right) = [\![D]\!]\left(\left[\!\left[S_{1D}^{(k)}\right]\!\right], \ldots, \left[\!\left[S_{nD}^{(k)}\right]\!\right]\right) \\ &= \left([\![S_1]\!]\left(\left[\!\left[S_{1D}^{(k)}\right]\!\right], \ldots, \left[\!\left[S_{nD}^{(k)}\right]\!\right]\right), \ldots, [\![S_n]\!]\left(\left[\!\left[S_{1D}^{(k)}\right]\!\right], \ldots, \left[\!\left[S_{nD}^{(k)}\right]\!\right]\right)\right) \\ &= \left(\left[\!\left[S_1\left[S_{1D}^{(k)}/X_1, \ldots, S_{nD}^{(k)}/X_n\right]\right]\!\right], \ldots, \left[\!\left[S_n\left[S_{1D}^{(k)}/X_1, \ldots, S_{nD}^{(k)}/X_n\right]\right]\!\right]\right) \\ &= \left(\left[\!\left[S_{1D}^{(k+1)}\right]\!\right], \ldots, \left[\!\left[S_{nD}^{(k+1)}\right]\!\right]\right). \end{aligned}$$

Finally, using equation (3.16), Proposition 3.4.3, the Knaster-Tarski Theorem and Proposition 3.4.1 (iii), we obtain:

$$\begin{aligned}
[\![S]\!]\,(\mu\,[\![D]\!]) &= [\![S]\!]\left(\bigsqcup_{k=0}^{\infty}[\![D]\!]^{(k)}\left(\overline{\mathbf{0}}\right)\right)\\
&= [\![S]\!]\left(\bigsqcup_{k=0}^{\infty}\left(\left[\!\left[S_{1D}^{(k)}\right]\!\right],\ldots,\left[\!\left[S_{nD}^{(k)}\right]\!\right]\right)\right)\\
&= \bigsqcup_{k=0}^{\infty}[\![S]\!]\left(\left[\!\left[S_{1D}^{(k)}\right]\!\right],\ldots,\left[\!\left[S_{nD}^{(k)}\right]\!\right]\right)\\
&= \bigsqcup_{k=0}^{\infty}\left[\!\left[S\left[S_{1D}^{(k)},\ldots,S_{nD}^{(k)}\right]\right]\!\right]\\
&= \bigsqcup_{k=0}^{\infty}\left[\!\left[S_{D}^{(k+1)}\right]\!\right]\\
&= [\![S|D]\!].
\end{aligned}$$

□

To conclude this section, we leave the following two problems for the reader. As pointed out at the beginning of this section, the materials presented in this section are similar to the theory of classical recursive programs, but I believe that research on these two problems will reveal some interesting and subtle differences between recursive quantum programs and classical recursive programs.

Problem 3.4.1

(i) *Can a general measurement in a quantum program be implemented by a projective measurement together with a unitary transformation? If the program contains no recursions (and no loops), then this question was already answered by Proposition 2.1.1.*

(ii) *How can a measurement in a quantum program be deferred? For the case that the program contains no recursions (and no loops), this question was answered by the principle of deferred measurement in Subsection 2.2.6. The interesting case is a program with recursions or loops.*

Problem 3.4.2. *Only recursive quantum programs without parameters are considered in this section. How do we define recursive quantum programs with parameters? We will have to deal with two different kinds of parameters:*

(i) *classical parameters;*

(ii) *quantum parameters.*

Bernstein-Varzirani recursive Fourier sampling [1,37,38] and Grover fixed point quantum search [109] are two examples of recursive quantum programs with parameters.

3.5 ILLUSTRATIVE EXAMPLE: GROVER QUANTUM SEARCH

The quantum **while**-language and its extension with recursive quantum programs have been studied in the previous sections. We now use the quantum **while**-language to program the Grover search algorithm in order to illustrate its utility. For the convenience of the reader, let us first briefly recall the algorithm from Subsection 2.3.3. The database searched by the algorithm consists of $N = 2^n$ elements, indexed by numbers $0, 1, \ldots, N-1$. It is assumed that the search problem has exactly L solutions with $1 \leq L \leq \frac{N}{2}$, and we are supplied with an oracle – a black box with the ability to recognize solutions to the search problem. We identify an integer $x \in \{0, 1, \ldots, N-1\}$ with its binary representation $x \in \{0,1\}^n$. The oracle is represented by the unitary operator O on $n+1$ qubits:

$$|x\rangle|q\rangle \xrightarrow{O} |x\rangle|q \oplus f(x)\rangle$$

for all $x \in \{0,1\}^n$ and $q \in \{0,1\}$, where $f : \{0,1\}^n \to \{0,1\}$ defined by

$$f(x) = \begin{cases} 1 & \text{if } x \text{ is a solution,} \\ 0 & \text{otherwise} \end{cases}$$

is the characteristic function of solutions. The Grover operator G consists of the following steps:

(i) Apply the oracle O;
(ii) Apply the Hadamard transform $H^{\otimes n}$;
(iii) Perform a conditional phase shift Ph:

$$|0\rangle \to |0\rangle, \quad |x\rangle \to -|x\rangle \text{ for all } x \neq 0;$$

that is, $Ph = 2|0\rangle\langle 0| - I$.
(iv) Apply the Hadamard transform $H^{\otimes n}$ again.

A geometric intuition of operator G as a rotation was carefully described in Subsection 2.3.3. Employing the Grover operator, the search algorithm is described in Figure 3.4, where the number k of iterations of the Grover operator is taken to be the positive integer in the interval $[\frac{\pi}{2\theta} - 1, \frac{\pi}{2\theta}]$, and θ is the angle rotated by the Grover operator and defined by the equation:

$$\cos\frac{\theta}{2} = \sqrt{\frac{N-L}{2}} \; (0 \leq \frac{\theta}{2} \leq \frac{\pi}{2}).$$

Now we program the Grover algorithm in the quantum **while**-language. We use $n+2$ quantum variables: $q_0, q_1, \ldots, q_{n-1}, q, r$.

- Their types are as follows:

$$type(q_i) = type(q) = \textbf{Boolean} \; (0 \leq i < n),$$
$$type(r) = \textbf{integer}$$

- **Procedure:**

1. $|0\rangle^{\otimes n}|1\rangle$
2. $\overset{H^{\otimes(n+1)}}{\rightarrow} \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle|-\rangle = \left(\cos\frac{\theta}{2}|\alpha\rangle + \sin\frac{\theta}{2}|\beta\rangle\right)|-\rangle,$
3. $\overset{G^k}{\rightarrow} \left[\cos\left(\frac{2k+1}{2}\theta\right)|\alpha\rangle + \sin\left(\frac{2k+1}{2}\theta\right)|\beta\rangle\right]|-\rangle$
4. $\overset{\text{measure the first } n \text{ qubits in the computational basis}}{\rightarrow} |x\rangle|-\rangle$

FIGURE 3.4

Grover search algorithm.

- The variable r is introduced to count the number of iterations of the Grover operator. We use quantum variable r instead of a classical variable for this purpose since, for the reason of simplicity, classical variables are not included in the quantum **while**-language.

Then the Grover algorithm can be written as the program *Grover* in Figure 3.5. Note that the size of the searched database is $N = 2^n$, so n in the program *Grover* should be understood as a metavariable. Several ingredients in *Grover* are specified as follows:

- The measurement $M = \{M_0, M_1\}$ in the loop guard (line 7) is given as follows:

$$M_0 = \sum_{l \geq k} |l\rangle_r\langle l|, \quad M_1 = \sum_{l<k} |l\rangle_r\langle l|$$

with k being a positive integer in the interval $\left[\frac{\pi}{2\theta} - 1, \frac{\pi}{2\theta}\right]$;

- **Program:**

1. $q_0 := |0\rangle;\ q_1 := |0\rangle;\ \ldots\ldots;\ q_{n-1} := |0\rangle;$
2. $q := |0\rangle;$
3. $r := |0\rangle;$
4. $q := X[q];$
5. $q_0 := H[q_0];\ q_1 := H[q_1];\ \ldots\ldots;\ q_{n-1} := H[q_{n-1}];$
6. $q := H[q];$
7. **while** $M[r] = 1$ **do** D **od**;
8. **if** $(\square x \cdot M'[q_0, q_1, \ldots, q_{n-1}] = x \rightarrow$ **skip**$)$ **fi**

FIGURE 3.5

Quantum search program *Grover*.

- The loop body D (line 7) is given in Figure 3.6;
- In the **if** . . . **fi** statement (line 8), N is the measurement in the computational basis of n qubits; that is,

$$M' = \{M'_x : x \in \{0,1\}^n\}$$

with $M'_x = |x\rangle\langle x|$ for every x.

- **Loop Body:**

1. $q_0, q_1, \ldots, q_{n-1}, q := O[q_0, q_1, \ldots, q_{n-1}, q]$;
2. $q_0 := H[q_0]$; $q_1 := H[q_1]$; $\ldots\ldots$; $q_{n-1} := H[q_{n-1}]$;
3. $q_0, q_1, \ldots, q_{n-1} := Ph[q_0, q_1, \ldots, q_{n-1}]$;
4. $q_0 := H[q_0]$; $q_1 := H[q_1]$; $\ldots\ldots$; $q_{n-1} := H[q_{n-1}]$;
5. $r := r + 1$

FIGURE 3.6

Loop body D.

The correctness of this program will be proved in the next chapter using the program logic developed there.

3.6 PROOFS OF LEMMAS

Several lemmas about the domains of partial density operators and quantum operations were presented without proofs in Subsection 3.3.2. For completeness, we give their proofs in this section.

The proof of Lemma 3.3.2 requires the notion of square root of a positive operator, which in turn requires the spectral decomposition theorem for Hermitian operators in an infinite-dimensional Hilbert space $\mathcal{H}$. Recall from Definition 2.1.16 that an operator $M \in \mathcal{L}(\mathcal{H})$ is Hermitian if $M^\dagger = M$. As defined in Subsection 2.1.2, a projector P_X is associated with each closed subspace X of $\mathcal{H}$. A spectral family in $\mathcal{H}$ is a family

$$\{E_\lambda\}_{-\infty<\lambda<+\infty}$$

of projectors indexed by real numbers λ satisfying the following conditions:

(i) $E_{\lambda_1} \sqsubseteq E_{\lambda_2}$ whenever $\lambda_1 \leq \lambda_2$;
(ii) $E_\lambda = \lim_{\mu\to\lambda+} E_\mu$ for each λ; and
(iii) $\lim_{\lambda\to-\infty} E_\lambda = 0_\mathcal{H}$ and $\lim_{\lambda\to+\infty} E_\lambda = I_\mathcal{H}$.

Theorem 3.6.1 ([182], Theorem III.6.3). *(Spectral decomposition) If M is a Hermitian operator with $spec(M) \subseteq [a, b]$, then there is a spectral family $\{E_\lambda\}$ such that*

$$M = \int_a^b \lambda dE_\lambda,$$

where the integral in the right-hand side is defined to be an operator satisfying the following condition: for any $\epsilon > 0$, there exists $\delta > 0$ such that for any $n \geq 1$ and $x_0, x_1, \ldots, x_{n-1}, x_n, y_1, \ldots, y_{n-1}, y_n$ with

$$a = x_0 \leq y_1 \leq x_1 \leq \ldots \leq y_{n-1} \leq x_{n-1} \leq y_n \leq x_n = b,$$

it holds that

$$d\left(\int_a^b \lambda dE_\lambda, \sum_{i=1}^n y_i (E_{x_i} - E_{x_{i-1}})\right) < \epsilon$$

whenever $\max_{i=1}^n (x_i - x_{i-1}) < \delta$. Here, $d(\cdot, \cdot)$ stands for the distance between operators (see Definition 2.1.14).

Now we are able to define the square root of a positive operator A. Since A is a Hermitian operator, it enjoys a spectral decomposition:

$$A = \int \lambda dE_\lambda.$$

Then its square root is defined to be

$$\sqrt{A} = \int \sqrt{\lambda} dE_\lambda.$$

With these preliminaries, we can give:

Proof of lemma 3.3.2. First, for any positive operator A, we get:

$$|\langle \varphi | A | \psi \rangle|^2 = \left| \left(\sqrt{A} |\varphi\rangle, \sqrt{A} |\psi\rangle \right) \right|^2 \leq \langle \varphi | A | \varphi \rangle \langle \psi | A | \psi \rangle \tag{3.18}$$

by the Cauchy-Schwarz inequality (see [174], page 68).

Now let $\{\rho_n\}$ be an increasing sequence in $(\mathcal{D}(\mathcal{H}), \sqsubseteq)$. For any $|\psi\rangle \in \mathcal{H}$, put $A = \rho_n - \rho_m$ and $|\varphi\rangle = A|\psi\rangle$. Then

$$\langle \psi | A | \psi \rangle \leq \langle \psi | \rho_n | \psi \rangle \leq \| \psi \|^2 \cdot tr(\rho_n) \leq \| \psi \|^2,$$

and similarly we have $\langle \varphi | A | \varphi \rangle \leq \| \varphi \|^2$. Thus, it follows from equation (3.18) that

$$|\langle \varphi | A | \psi \rangle|^2 \leq \| \psi \|^2 \cdot \| \varphi \|^2.$$

Furthermore, we obtain:

$$\begin{aligned}\|A\|^4 &= \sup_{|\psi\rangle\neq 0} \frac{\|A|\psi\rangle\|^4}{\|\psi\|^4} \\ &= \sup_{|\psi\rangle\neq 0} \frac{\langle\varphi|A|\psi\rangle^2}{\|\psi\|^4} \\ &\leq \sup_{|\psi\rangle\neq 0} \frac{\|\varphi\|^2}{\|\psi\|^2} \\ &= \sup_{|\psi\rangle\neq 0} \frac{\|A|\psi\rangle\|^2}{\|\psi\|^2} = \|A\|^2\end{aligned}$$

and $\|A\| \leq 1$. This leads to

$$\begin{aligned}\langle\varphi|A|\varphi\rangle &= \left(A\sqrt{A}|\psi\rangle, A\sqrt{A}|\psi\rangle\right) \\ &= \left\|A\sqrt{A}|\psi\rangle\right\|^2 \\ &\leq \|A\|^2 \cdot \left\|\sqrt{A}|\psi\rangle\right\|^2 \\ &= \left(\sqrt{A}|\psi\rangle, \sqrt{A}|\psi\rangle\right) \\ &= \langle\psi|A|\psi\rangle.\end{aligned}$$

Using equation (3.18) once again we get:

$$\begin{aligned}\|\rho_n|\psi\rangle - \rho_m|\psi\rangle\|^4 &= |\langle\varphi|A|\psi\rangle|^2 \\ &\leq \langle\psi|A|\psi\rangle^2 = |\langle\psi|\rho_n|\psi\rangle - \langle\psi|\rho_m|\psi\rangle|^2.\end{aligned} \tag{3.19}$$

Note that $\{\langle\psi|\rho_n|\psi\rangle\}$ is an increasing sequence of real numbers bounded by $\|\psi\|^2$, and thus it is a Cauchy sequence. This together with equation (3.19) implies that $\{\rho_n|\psi\rangle\}$ is a Cauchy sequence in $\mathcal{H}$. So, we can define:

$$\left(\lim_{n\to\infty} \rho_n\right)|\psi\rangle = \lim_{n\to\infty} \rho_n|\psi\rangle.$$

Furthermore, for any complex numbers $\lambda_1, \lambda_2 \in \mathbb{C}$ and $|\psi_1\rangle, |\psi_2\rangle \in \mathcal{H}$, it holds that

$$\begin{aligned}\left(\lim_{n\to\infty} \rho_n\right)(\lambda_1|\psi_1\rangle + \lambda_2|\psi_2\rangle) &= \lim_{n\to\infty} \rho_n\,(\lambda_1|\psi_1\rangle + \lambda_2|\psi_2\rangle) \\ &= \lim_{n\to\infty} (\lambda_1\rho_n|\psi_1\rangle + \lambda_2\rho_n|\psi_2\rangle) \\ &= \lambda_1 \lim_{n\to\infty} \rho_n|\psi_1\rangle + \lambda_2 \lim_{n\to\infty} \rho_n|\psi_2\rangle \\ &= \lambda_1 \left(\lim_{n\to\infty} \rho_n\right)|\psi_1\rangle + \lambda_2 \left(\lim_{n\to\infty} \rho_n\right)|\psi_2\rangle,\end{aligned}$$

and $\lim_{n\to\infty}\rho_n$ is a linear operator. For any $|\psi\rangle \in \mathcal{H}$, we have:

$$\langle\psi|\lim_{n\to\infty}\rho_n|\psi\rangle = \left(|\psi\rangle, \lim_{n\to\infty}\rho_n|\psi\rangle\right) = \lim_{n\to\infty}\langle\psi|\rho_n|\psi\rangle \geq 0.$$

Thus, $\lim_{n\to\infty}\rho_n$ is positive. Let $\{|\psi_i\rangle\}$ be an orthonormal basis of $\mathcal{H}$. Then

$$\begin{aligned} tr\left(\lim_{n\to\infty}\rho_n\right) &= \sum_i \langle\psi_i|\lim_{n\to\infty}\rho_n|\psi_i\rangle \\ &= \sum_i \left(|\psi_i\rangle, \lim_{n\to\infty}\rho_n|\psi_i\rangle\right) \\ &= \lim_{n\to\infty}\sum_i \langle\psi_i|\rho_n|\psi_i\rangle \\ &= \lim_{n\to\infty} tr(\rho_n) \leq 1, \end{aligned}$$

and $\lim_{n\to\infty}\rho_n \in \mathcal{D}(\mathcal{H})$. So, it suffices to show that

$$\lim_{n\to\infty}\rho_n = \bigsqcup_{n=0}^{\infty}\rho_n;$$

that is,

(i) $\rho_m \sqsubseteq \lim_{n\to\infty}\rho_n$ for all $m \geq 0$; and
(ii) if $\rho_m \sqsubseteq \rho$ for all $m \geq 0$, then $\lim_{n\to\infty}\rho_n \sqsubseteq \rho$.

Note that for any positive operators B and C, $B \sqsubseteq C$ if and only if $\langle\psi|B|\psi\rangle \leq \langle\psi|C|\psi\rangle$ for all $|\psi\rangle \in \mathcal{H}$. Then both (i) and (ii) follow immediately from

$$\langle\psi|\lim_{n\to\infty}\rho_n|\psi\rangle = \lim_{n\to\infty}\langle\psi|\rho_n|\psi\rangle.$$

This completes the proof of Lemma 3.3.2.

The proof of Lemma 3.3.3 can be easily done based on Lemma 3.3.2.

Proof of Lemma 3.3.3. Suppose that $\mathcal{E}$ is a quantum operation with Kraus representation $\mathcal{E} = \sum_i E_i \circ E_i^\dagger$ (see Theorem 2.1.1), and $\{\rho_n\}$ is an increasing sequence in $\mathcal{D}(\mathcal{H})$. Then by Lemma 3.3.2 we obtain:

$$\begin{aligned} \mathcal{E}\left(\bigsqcup_n \rho_n\right) &= \mathcal{E}\left(\lim_{n\to\infty}\rho_n\right) \\ &= \sum_i E_i\left(\lim_{n\to\infty}\rho_n\right)E_i^\dagger \\ &= \lim_{n\to\infty}\sum_i E_i\rho_n E_i^\dagger \\ &= \lim_{n\to\infty}\mathcal{E}(\rho_n) \\ &= \bigsqcup_n \mathcal{E}(\rho_n). \end{aligned}$$

Finally, we can prove Lemma 3.3.4.

Proof of Lemma 3.3.4. Let $\{\mathcal{E}_n\}$ be an increasing sequence in $(\mathcal{QO}(\mathcal{H}), \sqsubseteq)$. Then for any $\rho \in \mathcal{D}(\mathcal{H})$, $\{\mathcal{E}_n(\rho)\}$ is an increasing sequence in $(\mathcal{D}(\mathcal{H}), \sqsubseteq)$. With Lemma 3.3.2 we can define:

$$\left(\bigsqcup_n \mathcal{E}_n\right)(\rho) = \bigsqcup_n \mathcal{E}_n(\rho) = \lim_{n\to\infty} \mathcal{E}_n(\rho),$$

and it holds that

$$tr\left(\left(\bigsqcup_n \mathcal{E}_n\right)(\rho)\right) = tr\left(\lim_{n\to\infty} \mathcal{E}_n(\rho)\right) = \lim_{n\to\infty} tr(\mathcal{E}_n(\rho)) \leq 1$$

because $tr(\cdot)$ is continuous. Furthermore, $\bigsqcup_n \mathcal{E}_n$ can be defined on the whole of $\mathcal{L}(\mathcal{H})$ by linearity. The defining equation of $\bigsqcup_n \mathcal{E}_n$ implies:

(i) $\mathcal{E}_m \sqsubseteq \bigsqcup_n \mathcal{E}_n$ for all $m \geq 0$;
(ii) if $\mathcal{E}_m \sqsubseteq \mathcal{F}$ for all $m \geq 0$ then $\bigsqcup_n \mathcal{E}_n \sqsubseteq \mathcal{F}$.

So, it suffices to show that $\bigsqcup_n \mathcal{E}_n$ is completely positive. Suppose that $\mathcal{H}_R$ is an extra Hilbert space. For any $C \in \mathcal{L}(\mathcal{H}_R)$ and $D \in \mathcal{L}(\mathcal{H})$, we have:

$$\begin{aligned}\left(\mathcal{I}_R \otimes \bigsqcup_n \mathcal{E}_n\right)(C \otimes D) &= C \otimes \left(\bigsqcup_n \mathcal{E}_n\right)(D) \\ &= C \otimes \lim_{n\to\infty} \mathcal{E}_n(D) \\ &= \lim_{n\to\infty} (C \otimes \mathcal{E}_n(D)) \\ &= \lim_{n\to\infty} (\mathcal{I}_R \otimes \mathcal{E}_n)(C \otimes D).\end{aligned}$$

Then for any $A \in \mathcal{L}(\mathcal{H}_R \otimes \mathcal{H})$ we get:

$$\left(\mathcal{I}_R \otimes \bigsqcup_n \mathcal{E}_n\right)(A) = \lim_{n\to\infty} (\mathcal{I}_R \otimes \mathcal{E}_n)(A)$$

by linearity. Thus, if A is positive, then $(\mathcal{I}_R \otimes \mathcal{E}_n)(A)$ is positive for all n, and $\left(\mathcal{I}_R \otimes \bigsqcup_n \mathcal{E}_n\right)(A)$ is positive.

3.7 BIBLIOGRAPHIC REMARKS

The quantum **while**-language presented in Section 3.1 was defined in [221], but various quantum program constructs in it were introduced in the previous works by Sanders and Zuliani [191,241] and Selinger [194], among others. A general form of quantum **while**-loops was introduced and their properties were thoroughly investigated in [227]. A discussion about the existing quantum programming languages was

already given in Subsection 1.1.1; it is worth comparing the quantum programming language described in this chapter with the languages mentioned there.

The presentation of operational and denotational semantics in Sections 3.2 and 3.3 is mainly based on [221]. The denotational semantics was actually first given by Feng et al. in [82], but the treatments of denotational semantics in [82] and [221] are different: in [82], the denotational semantics is directly defined, whereas in [221], the operational semantics comes first and then the denotational semantics is derived from the operational semantics. The idea of encoding a probability and a density operator into a partial density operator in the transition rules was suggested by Selinger [194]. A domain theory for quantum computation was first considered by Kashefi [133]. Lemmas 3.3.2 and 3.3.4 were obtained by Selinger [194] in the case of finite-dimensional Hilbert spaces. The proof of Lemma 3.3.2 for the general case was given in [225], and it is essentially a modification of the proof of Theorem III.6.2 in [182]. A form of Proposition 3.3.6 was first presented by Selinger in [194]. The current statement of Proposition 3.3.6 is given based on the notion of local quantum variable, which was introduced in [233].

Recursion in quantum programming was first considered by Selinger in [194]. But the materials presented in Section 3.4 are slightly different from those in [194] and unpublished elsewhere.

Finally, it should be pointed out that this chapter is essentially the quantum generalization of the semantics of the classical **while**-programs and recursive programs as presented by Apt, de Boer and Olderog in [21].