CHAPTER 7

Quantum recursion

Recursion is one of the central ideas of computer science. Most programming languages support recursion or at least a special form of recursion such as the **while**-loop. A quantum extension of the **while**-loop was already introduced in Section 3.1. A more general notion of recursion in quantum programming was defined in Section 3.4. It was appropriately called *classical recursion of quantum programs* because its control flow is determined by the involved case statements of the form (3.3) and **while**-loops of the form (3.4) and thus is doomed to be classical, as discussed in Section 3.1.

In the last chapter, we studied quantum case statements and quantum choices of which the control flows are genuinely quantum because they are governed by quantum “coins.” This chapter further defines the notion of quantum recursion based on quantum case statement and quantum choice. The control flow of such a quantum recursive program is then quantum rather than classical. As we will see later, the treatment of quantum recursion with quantum control is much harder than classical recursion in quantum programming.

The chapter is organized as follows.

- The syntax of quantum recursive programs is defined in Section 7.1. Recursive quantum walks are introduced in Section 7.2 as examples for carefully motivating the notion of quantum recursion. Section 7.1 provides us with a language in which a precise formulation of recursive quantum walks is possible.
- It requires mathematical tools from second quantization – a theoretical framework in which we are able to depict quantum systems with a variable number of particles – to define the semantics of quantum recursive programs. Since it is used only in this chapter, second quantization was not included in the preliminaries chapter (Chapter 2). Instead, we introduce the basics of second quantization, in particular Fock spaces and operators in them, in Section 7.3.
- We define the semantics of quantum recursion in two steps. The first step is carried out in Section 7.4 where quantum recursive equations are solved in the free Fock space, which is mathematically convenient to manipulate but does not represent a realistic system in physics. The second step is completed in Section 7.5 where the solutions of recursive equations are symmetrized so that

Foundations of Quantum Programming. http://dx.doi.org/10.1016/B978-0-12-802306-8.00007-0

they can apply in the physically meaningful framework, namely the symmetric and antisymmetric Fock spaces of bosons and fermions. Furthermore, the principal system semantics of a quantum recursive program is defined in Section 7.6 by tracing out auxiliary "quantum coins" from its symmetrized semantics.

- Recursive quantum walks are reconsidered in Section 7.7 to illustrate various semantic notions introduced in this chapter. A special class of quantum recursions, namely quantum **while**-loops with quantum control flows are carefully examined in Section 7.8.

7.1 SYNTAX OF QUANTUM RECURSIVE PROGRAMS

In this section, we formally define the syntax of quantum recursive programs. To give the reader a clearer picture, we choose not to include quantum measurements in the declarations of quantum recursions. It is not difficult to add quantum measurements into the theory of quantum recursive programs by combining the ideas used in this chapter and the last one, but the presentation will be much more complicated.

Let us start by exhibiting the alphabet of the language of quantum recursive programs. In the last chapter, any quantum variable can serve as a "coin" in defining a quantum case statement. For convenience, in this chapter we explicitly separate quantum "coins" from other quantum variables; that is, we assume two sets of quantum variables:

- principal system variables, ranged over by $p, q, \ldots$;
- "coin" variables, ranged over by $c, d, \ldots$.

These two sets are required to be disjoint. We also assume a set of procedure identifiers, ranged over by $X, X_1, X_2, \ldots$. A modification of the quantum programming language QuGCL presented in the last chapter is defined by the next:

Definition 7.1.1. *Program schemes are defined by the following syntax:*

$$P ::= X \mid \mathbf{abort} \mid \mathbf{skip} \mid P_1; P_2 \mid U[\overline{c}, \overline{q}] \mid \mathbf{qif}\ [c](\square i \cdot |i\rangle \rightarrow P_i)\ \mathbf{fiq}$$

Obviously, this definition is obtained from Definition 6.2.1 by adding procedure identifiers and excluding measurements (and thus classical variables for recording the outcomes of measurements). More explicitly,

- X is a procedure identifier;
- **abort**, **skip**, and sequential composition $P_1; P_2$ are as in Definition 6.2.1.
- Unitary transformation $U[\overline{c}, \overline{q}]$ is the same as before except that "coins" and principal system variables are separated; that is, $\overline{c}$ is a sequence of "coin" variables, $\overline{q}$ is a sequence of principal system variables, and U is a unitary operator in the state Hilbert space of the system consisting of $\overline{c}$ and $\overline{q}$. We will always put "coin" variables before principal system variables. Both $\overline{c}$ and $\overline{q}$ are

allowed to be empty. When $\overline{c}$ is empty, we simply write $U[\overline{q}]$ for $U[\overline{c},\overline{q}]$ and it describes the evolution of the principal system $\overline{q}$; when $\overline{q}$ is empty, we simply write $U[\overline{c}]$ for $U[\overline{c},\overline{q}]$ and it describes the evolution of the "coins" $\overline{c}$. If both $\overline{c}$ and $\overline{q}$ are not empty, then $U[\overline{c},\overline{q}]$ describes the interaction between "coins" $\overline{c}$ and the principal system $\overline{q}$.

- Quantum case statement $\mathbf{qif}\ [c](\Box i \cdot |i\rangle \to P_i)\ \mathbf{fiq}$ is as in Definition 6.2.1. Here, for simplicity we only use a single "coin" c rather than a sequence of "coins." Once again, we emphasize that "coin" c is required not to occur in any subprogram P_i because, according to its physical interpretation, it is always external to the principal system.

Recall from Section 6.5, quantum choice can be defined in terms of quantum case statement and sequential composition:

$$[P(c)]\bigoplus_i (|i\rangle \to P_i) \stackrel{\triangle}{=} P;\mathbf{qif}\ [c]\ (\Box i \cdot |i\rangle \to P_i)\ \ \mathbf{fiq}$$

where P contains only quantum variable c. In particular, if the "coin" is a qubit, then a quantum choice can be abbreviated as

$$P_0 \oplus_P P_1 \text{ or } P_0\ {}_P\!\oplus P_1.$$

Quantum program schemes without procedure identifiers are actually a special class of quantum programs considered in the last chapter. So, their semantics can be directly derived from Definition 6.4.2. For the convenience of the reader, we explicitly display their semantics in the following definition. The principal system of a quantum program P is the composition of the systems denoted by principal system variables appearing in P. We write $\mathcal{H}$ for the state Hilbert space of the principal system.

Definition 7.1.2. *The semantics $[\![P]\!]$ of a program P (i.e., a program scheme without procedure identifiers) is inductively defined as follows:*

(i) *If $P = \mathbf{abort}$, then $[\![P]\!] = 0$ (the zero operator in $\mathcal{H}$), and if $P = \mathbf{skip}$, then $[\![P]\!] = I$ (the identity operator in $\mathcal{H}$);*
(ii) *If P is a unitary transformation $U[\overline{c},\overline{q}]$, then $[\![P]\!]$ is the unitary operator U (in the state Hilbert space of the system consisting of $\overline{c}$ and $\overline{q}$);*
(iii) *If $P = P_1;P_2$, then $[\![P]\!] = [\![P_2]\!] \cdot [\![P_1]\!]$;*
(iv) *If $P = \mathbf{qif}\ [c](\Box i \cdot |i\rangle \to P_i)\ \mathbf{fiq}$, then*

$$[\![P]\!] = \Box\left(c, |i\rangle \to [\![P_i]\!]\right) \stackrel{\triangle}{=} \sum_i \left(|i\rangle_c\langle i| \otimes [\![P_i]\!]\right). \tag{7.1}$$

This definition is a special case of Definition 6.4.2 where the semi-classical semantics of QuGCL programs were defined. However, the reader may have noticed that in the preceding definition we use $[\![P]\!]$ to denote the semantics of a program P, and in the last chapter $\|P\|$ is employed to denote the semi-classical semantics of

P and $[\![P]\!]$ denotes the purely quantum semantics of P. We choose to write $[\![P]\!]$ for the semantics of P in Definition 7.1.2 because in this chapter programs contain no measurements, so their semi-classical semantics and purely quantum semantics are essentially the same. Obviously, $[\![P]\!]$ in the preceding definition is an operator in the state Hilbert space of the system consisting of both the principal system of P and the "coins" in P; i.e., $\mathcal{H}_C \otimes \mathcal{H}$, where $\mathcal{H}_C$ is the state space of the "coins" in P. Since **abort** may appear in P, $[\![P]\!]$ is not necessarily a unitary. Using the terminology from the last chapter, $[\![P]\!]$ can be seen as an operator-valued function in $\mathcal{H}_C \otimes \mathcal{H}$ over a singleton $\Delta = \{\epsilon\}$.

Finally, we can define the syntax of quantum recursive programs. If a program scheme P contains at most the procedure identifiers $X_1, \dots, X_m$, then we write

$$P = P[X_1, \dots, X_m].$$

Definition 7.1.3

(i) *Let $X_1, \dots, X_m$ be different procedure identifiers. A declaration for $X_1, \dots, X_m$ is a system of equations:*

$$D : \begin{cases} X_1 \Leftarrow P_1, \\ \qquad \cdots\cdots \\ X_m \Leftarrow P_m, \end{cases}$$

where for every $1 \leq i \leq m$, $P_i = P_i[X_1, \dots, X_m]$ is a program scheme containing at most procedure identifiers $X_1, \dots, X_m$.

(ii) *A recursive program consists of a program scheme $P = P[X_1, \dots, X_m]$, called the main statement, and a declaration D for $X_1, \dots, X_m$ such that all "coin" variables in P do not appear in D; that is, they do not appear in the procedure bodies $P_1, \dots, P_m$.*

The requirement in the preceding definition that the "coins" in the main statement P and those in the declaration D are distinct is obviously necessary because a "coin" used to define a quantum case statement is always considered to be external to its principal system.

Perhaps, the reader already noticed that Definition 7.1.3 looks almost the same as Definitions 3.4.2 and 3.4.3. But there is actually an essential difference between them: in the preceding definition, program schemes $P_1, \dots, P_m$ in the declaration D and the main statement P can contain quantum case statements, whereas only case statements of the form (3.3) (and **while**-loops of the form (3.4)) are present in Definitions 3.4.2 and 3.4.3. Therefore, as said repeatedly, a recursive program defined here has quantum control flow, but a recursive program considered in Section 3.4 has only classical control flow. For this reason, the former is termed a *quantum recursive program* and the latter a *recursive quantum program*. As we saw in Section 3.4, the techniques in classical programming theory can be straightforwardly generalized to define the semantics of recursive quantum programs. On the other hand, if a quantum program containing quantum case statements is not recursively defined, its semantics can be defined using the techniques developed in the last chapter; in particular, Definition 7.1.2 is a simplified version of Definition 6.4.2. However, new techniques are required in order to define the semantics of quantum recursive programs, as will be clearly seen at the end of the next section.

7.2 MOTIVATING EXAMPLES: RECURSIVE QUANTUM WALKS

The syntax of quantum recursive programs was introduced in the last section. The aim of this section is two-fold:

(i) present a motivating example of quantum recursive program;
(ii) give a hint to answering the question: how to define the semantics of quantum recursive programs?

This aim will be achieved by considering a class of examples, called recursive quantum walks, which are a variant of the quantum walks introduced in Subsection 2.3.4. Actually, recursive quantum walks can only be properly presented with the help of the syntax given in the last section.

7.2.1 SPECIFICATION OF RECURSIVE QUANTUM WALKS

A one-dimensional quantum walk, called a Hadamard walk, was defined in Example 2.3.1. For simplicity, in this section we focus on the recursive Hadamard walk, a modification of the Hadamard walk. Recursive quantum walks on a graph can be defined by modifying Example 2.3.2 in a similar way.

Recall from Examples 2.3.2 and 6.7.1 that the state Hilbert space of the Hadamard walk is $\mathcal{H}_d \otimes \mathcal{H}_p$, where

- $\mathcal{H}_d = \text{span}\{|L\rangle, |R\rangle\}$ is the "direction coin" space, and L, R are used to indicate the directions Left and Right, respectively;
- $\mathcal{H}_p = \text{span}\{|n\rangle : n \in \mathbb{Z}\}$ is the position space, and n indicates the position marked by integer n.

The single-step operator W of the Hadamard walk is a quantum choice, which is the sequential composition of a "coin-tossing" Hadamard operator H on the "direction coin" d and translation operator T on the position variable p. The translation T is a quantum case statement that selects left or right translations according to the basis states $|L\rangle, |R\rangle$ of the "coin" d:

- If d is in state $|L\rangle$ then the walker moves one position left;
- If d is in state $|R\rangle$ then it moves one position right.

Of course, d can also be in a superposition of $|L\rangle$ and $|R\rangle$, and thus a superposition of left and right translations happens, which produces a quantum control flow. Formally,

$$\begin{aligned} W = T_L[p] \oplus_{H[d]} T_R[p] = H[d]; \mathbf{qif}\ [d]\ & |L\rangle \to T_L[p] \\ \square\ & |R\rangle \to T_R[p] \\ \mathbf{fiq} & \end{aligned}$$

where T_L and T_R are the left and right translation operators, respectively, in the position space $\mathcal{H}_p$. The Hadamard walk is then defined in a simple way of recursion with the single-step operator W, namely *repeated applications* of W.

Now we slightly modify the Hadamard walk using a little bit more complicated form of recursion.

Example 7.2.1

(i) *The unidirectionally recursive Hadamard walk first runs the "coin-tossing" Hadamard operator $H[d]$ and then a quantum case statement:*
- *If the "direction coin" d is in state $|L\rangle$ then the walker moves one position left;*
- *If d is in state $|R\rangle$ then it moves one position right, followed by* ***a procedure behaving as the recursive walk itself****.*

Using the syntax presented in the last section, the unidirectionally recursive Hadamard walk can be precisely defined to be a recursive program X declared by the following equation:

$$X \Leftarrow T_L[p] \oplus_{H[d]} (T_R[p]; X) \tag{7.2}$$

where d, p are the direction and position variables, respectively.

(ii) *The bidirectionally recursive Hadamard walk first runs the "coin-tossing" Hadamard operator $H[d]$ and then a quantum case statement:*
- *If the "direction coin" d is in state $|L\rangle$ then the walker moves one position left, followed by* ***a procedure behaving as the recursive walk itself****;*
- *If d is in state $|R\rangle$ then it moves one position right, also followed by* ***a procedure behaving as the recursive walk itself****.*

More precisely, the walk can be defined to be the program X declared by the following recursive equation:

$$X \Leftarrow (T_L[p]; X) \oplus_{H[d]} (T_R[p]; X). \tag{7.3}$$

(iii) *A variant of the bidirectionally recursive Hadamard walk is the program X (or Y) declared by the following system of recursive equations:*

$$\begin{cases} X \Leftarrow T_L[p] \oplus_{H[d]} (T_R[p]; Y), \\ Y \Leftarrow (T_L[p]; X) \oplus_{H[d]} T_R[p]. \end{cases} \tag{7.4}$$

The main difference between recursive equations (7.3) and (7.4) is that in the former procedure identifier X is calling itself, but in the latter X is calling Y and at the same time Y is calling X.

(iv) *Note that we used the same "coin" d in the two equations of* (7.4). *If two different "coins" d and e are used, then we have another variant of the bidirectionally recursive Hadamard walk specified by*

$$\begin{cases} X \Leftarrow T_L[p] \oplus_{H[d]} (T_R[p]; Y), \\ Y \Leftarrow (T_L[p]; X) \oplus_{H[e]} T_R[p]. \end{cases} \tag{7.5}$$

(v) *We can define a recursive quantum walk in another way if a quantum case statement with three branches is employed:*

$$\begin{array}{rl} X \Leftarrow U[d]; \mathbf{qif}\ [d] & |L\rangle \rightarrow T_L[p] \\ \square & |R\rangle \rightarrow T_R[p] \\ \square & |I\rangle \rightarrow X \\ \mathbf{fiq} & \end{array}$$

where d is not a qubit but a qutrit, i.e., a quantum system with 3*-dimensional state Hilbert space* $\mathcal{H}_d = span\{|L\rangle, |R\rangle, |I\rangle\}$*, L, R stand for the directions Left and Right, respectively, and I for Iteration, and U is a* 3×3 *unitary matrix, e.g. the* 3*-dimensional Fourier transform:*

$$F_3 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & e^{\frac{2}{3}\pi i} & e^{\frac{4}{3}\pi i} \\ 1 & e^{\frac{4}{3}\pi i} & e^{\frac{2}{3}\pi i} \end{pmatrix}.$$

Now let us have a glimpse of the behaviors of recursive quantum walks. We employ an idea similar to that used in Section 3.2. We use E to denote the empty program or termination. A configuration is defined to be a pair

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

with S being a program or the empty program E, and $|\psi\rangle$ a pure state of the quantum system. Then the behavior of a program can be visualized by a sequence of transitions between superpositions of configurations. Note that in Section 3.2 the computation of a program is a sequence of transitions between configurations. Here, however, we have to consider transitions between superpositions of configurations. Naturally, these superpositions of configurations are generated by quantum control flow of the program.

We only consider the unidirectionally recursive quantum walk X declared by equation (7.2) as an example. The reader is encouraged to work out the first few transitions of other walks in the preceding example in order to better understand how quantum recursive calls happen. Assume that it is initialized in state $|L\rangle_d|0\rangle_p$; that is, the "coin" is in direction L and the walker is at position 0. Then we have:

$$\begin{aligned}
\langle X, |L\rangle_d|0\rangle_p\rangle &\overset{(a)}{\to} \frac{1}{\sqrt{2}}\langle E, |L\rangle_d|-1\rangle_p\rangle + \frac{1}{\sqrt{2}}\langle X, |R\rangle_d|1\rangle_p\rangle \\
&\overset{(b)}{\to} \frac{1}{\sqrt{2}}\langle E, |L\rangle_d|-1\rangle_p\rangle + \frac{1}{2}\langle E, |R\rangle_d|L\rangle_{d_1}|0\rangle_p\rangle + \frac{1}{2}\langle X, |R\rangle_d|R\rangle_{d_1}|2\rangle_p\rangle \\
&\to \dots\dots \\
&\to \sum_{i=0}^{n}\frac{1}{\sqrt{2^{i+1}}}\langle E, |R\rangle_{d_0}\dots|R\rangle_{d_{i-1}}|L\rangle_{d_i}|i-1\rangle_p\rangle \\
&\qquad + \frac{1}{\sqrt{2^{n+1}}}\langle X, |R\rangle_{d_0}\dots|R\rangle_{d_{n-1}}|R\rangle_{d_n}|n+1\rangle_p\rangle
\end{aligned} \tag{7.6}$$

Here, $d_0 = d$, and new quantum "coins" $d_1, d_2, \dots$ that are identical to the original "coin" d are introduced in order to avoid the conflict of variables for "coins." To see why these distinct "coins" $d_1, d_2, \dots$ have to be introduced, we recall from Sections 6.1 and 6.2 that the "coins" $\overline{q}$ in a quantum case statement **qif** $[\overline{q}]$ $(\square i \cdot |i\rangle \to S_i)$ **fiq** are required to be external to subprograms S_i. Therefore, in equation (7.2) "coin" d is external to procedure X. Now let us see what happens in equation (7.6). First, the term after the arrow $\overset{(a)}{\to}$ is obtained by replacing the symbol X in the term before $\overset{(a)}{\to}$ with $T_L[p] \oplus_{H[d]} (T_R[p]; X)$. So, in the term after $\overset{(a)}{\to}$, d is external to X. To obtain the term after $\overset{(b)}{\to}$, we replace the symbol X in the term after $\overset{(a)}{\to}$ by $T_L[p] \oplus_{H[d_1]} (T_R[p]; X)$. Here, d_1 must be different from d; otherwise in the term after $\overset{(a)}{\to}$, $d = d_1$ occurs (although implicitly) in X, and thus a contradiction occurs. Repeating this argument illustrates that $d_0 = d, d_1, d_2, \dots$ should be different from each other.

Exercise 7.2.1. *Show the first few steps of recursive quantum walks defined by equations (7.4) and (7.5) initialized in $|L\rangle_d|0\rangle_p$. Observe the difference between the behaviors of the two walks. Notice that such a difference is impossible for classical random walks, where it does not matter if two different "coins" with the same probability distribution are used.*

The preceding recursive quantum walks are good examples of quantum recursion, but their behaviors are not very interesting from the viewpoint of quantum physics. As pointed out in Subsection 2.3.4, the major difference between the behaviors of classical random walks and quantum walks is caused by quantum interference – two separate paths leading to the same point may be out of phase and cancel one another. It is clear from equation (7.6) that quantum interference does not happen in the unidirectionally recursive quantum walk. Similarly, no quantum interference occurs in the other recursive quantum walks defined in the preceding example. The following is a much more interesting recursive quantum walk that shows a new phenomenon of quantum interference. As can be seen in equation (2.19), the paths that are cancelled in a (nonrecursive) quantum walk are finite. However, it is possible that infinite paths are cancelled in a recursive quantum walk.

Example 7.2.2. *Let $n \geq 2$. A variant of a bidirectionally recursive quantum walk can be defined as the program X declared by the following recursive equation:*

$$X \Leftarrow (T_L[p] \oplus_{H[d]} T_R[p])^n; ((T_L[p]; X) \oplus_{H[d]} (T_R[p]; X)) \tag{7.7}$$

Here, we use S^n to denote the sequential composition of n copies of a program S.

Now let us look at the behavior of this walk. We assume that the walk is initialized in state $|L\rangle_d|0\rangle_p$. Then the first three steps of the walk are given as follows:

$$\begin{aligned}
\langle X, |L\rangle_d|0\rangle_p\rangle &\to \frac{1}{\sqrt{2}}[\langle X_1, |L\rangle_d|-1\rangle_p\rangle + \langle X_1, |R\rangle_d|1\rangle_p\rangle] \\
&\to \frac{1}{2}[\langle X_2, |L\rangle_d|-2\rangle_p\rangle + \langle X_2, |R\rangle_d|0\rangle_p\rangle + \langle X_2, |L\rangle_d|0\rangle_p\rangle - \langle X_2, |R\rangle_d|2\rangle_p\rangle] \\
&\to \frac{1}{2\sqrt{2}}[(X_3, |L\rangle_d|-3\rangle_p) + (X_3, |R\rangle_d|-1\rangle_p) + (X_3, |L\rangle_d|-1\rangle_p) - (X_3, |R\rangle_d|1\rangle_p) \\
&\qquad + \langle X_3, |L\rangle_d|-1\rangle_p\rangle + \langle X_3, |R\rangle_d|1\rangle_p\rangle - \langle X_3, |L\rangle_d|1\rangle_p\rangle + \langle X_3, |R\rangle_d|3\rangle_p\rangle] \\
&= \frac{1}{2\sqrt{2}}[\langle X_3, |L\rangle_d|-3\rangle_p\rangle + \langle X_3, |R\rangle_d|-1\rangle_p\rangle + 2\langle X_3, |L\rangle_d|-1\rangle_p\rangle \\
&\qquad - \langle X_3, |L\rangle_d|1\rangle_p\rangle + \langle X_3, |R\rangle_d|3\rangle_p\rangle]
\end{aligned} \tag{7.8}$$

where

$$X_i = (T_L[p] \oplus_{H[d]} T_R[p])^{n-i}; ((T_L[p]; X) \oplus_{H[d]} (T_R[p]; X))$$

for $i = 1, 2, 3$. We observe that in the last step of equation (7.8) two configurations

$$-\langle X_3, |R\rangle_d|1\rangle_p\rangle, \quad \langle X_3, |R\rangle_d|1\rangle_p\rangle$$

cancel one another. It is clear that both of them can generate infinite paths because they contain the recursive walk X itself. Comparing equation (7.8) with (7.6), the reader may wonder why no new "coins" were introduced in (7.8). Actually, only the part $(T_L[p] \oplus_{H[d]} T_R[p])^n$ in the right-hand side of equation (7.7) is executed and no recursive calls happen in the three steps given in equation (7.8). Of course, fresh "coins" will be needed in the later steps where X is recursively called in order to avoid variable conflicts.

The behavior of the recursive program specified by the following equation:

$$X \Leftarrow ((T_L[p]; X) \oplus_{H[d]} (T_R[p]; X)); (T_L[p] \oplus_{H[d]} T_R[p])^n \tag{7.9}$$

is even more puzzling. Note that equation (7.9) is obtained from equation (7.7) by changing the order of the two subprograms on the right-hand side.

Exercise 7.2.2. *Examine the behavior of the walk X declared by equation (7.9) starting at $|L\rangle_d|0\rangle_p$.*

7.2.2 HOW TO SOLVE RECURSIVE QUANTUM EQUATIONS

We have already seen from equations (7.6) and (7.8) the first steps of the recursive quantum walks. But a precise description of their behaviors amounts to solving recursive equations (7.2), (7.3), (7.4), (7.5) and (7.7). In the theory of classical programming languages, syntactic approximation is employed to define the semantics of recursive programs. It was also successfully used in Section 3.4 to define the semantics of recursive quantum programs. Naturally, we would like to see whether syntactic approximation can be applied to quantum recursive programs too. To begin with, let us recall this technique by considering a simple recursive program declared by a single equation

$$X \Leftarrow F(X).$$

Let

$$\begin{cases} X^{(0)} = \textbf{abort}, \\ X^{(n+1)} = F[X^{(n)}/X] \ \text{ for } n \geq 0. \end{cases}$$

where $F[X^{(n)}/X]$ is the result of substitution of X in $F(X)$ by $X^{(n)}$. The program $X^{(n)}$ is called the nth syntactic approximation of X. Roughly speaking, the syntactic approximations $X^{(n)}$ $(n = 0, 1, 2, \ldots)$ describe the initial fragments of the behavior of the recursive program X. Then the semantics $[\![X]\!]$ of X is defined to be the limit of the semantics $[\![X^{(n)}]\!]$ of its syntactic approximations $X^{(n)}$:

$$[\![X]\!] = \lim_{n\to\infty} [\![X^{(n)}]\!].$$

Now we try to apply this method to the unidirectionally recursive Hadamard walk and construct its syntactic approximations as follows:

$$\begin{aligned} X^{(0)} &= \textbf{abort}, \\ X^{(1)} &= T_L[p] \oplus_{H[d]} (T_R[p]; \textbf{abort}), \\ X^{(2)} &= T_L[p] \oplus_{H[d]} (T_R[p]; T_L[p] \oplus_{H[d_1]} (T_R[p]; \textbf{abort})), \\ X^{(3)} &= T_L[p] \oplus_{H[d]} (T_R[p]; T_L[p] \oplus_{H[d_1]} (T_R[p]; T_L[p] \oplus_{H[d_2]} (T_R[p]; \textbf{abort}))), \\ &\cdots\cdots\cdots\cdots \end{aligned} \tag{7.10}$$

However, a problem arises in constructing these approximations: we have to continuously introduce new "coin" variables in order to avoid variable conflict; that is, for every $n = 1, 2, \ldots$, we have to introduce a new "coin" variable d_n in the $(n+1)$th syntactic approximation because, as emphasized many times before, "coins" $d, d_1, \ldots, d_{n-1}$ should be considered external to the innermost system that contains d_n. Therefore, variables $d, d_1, d_2, \ldots, d_n, \ldots$ denote distinct "coins." On the other hand, they must be thought of as identical particles in the sense that their physical properties are the same. Moreover, the number of the "coin" particles that are needed in running the recursive Hadamard walk is usually unknown beforehand because we do not know when the walk terminates. Obviously, a solution to this problem requires a mathematical framework in which we can deal with quantum systems where the number of particles of the same type – the "coins" – may vary. It is worth noting that this problem appears only in the quantum case but not in the theory of classical programming languages, because it is caused by employing an external "coin" system in defining a quantum case statement.

7.3 SECOND QUANTIZATION

At the end of the last section, we observed that solving a quantum recursive equation requires a mathematical model of a quantum system consisting of a variable number of identical particles. It is clear that such a model is out of the scope of basic quantum mechanics described in Section 2.1, where we only considered a composite quantum system with a fixed number of subsystems that are not necessarily identical (see the Postulate of quantum mechanics 4 in Subsection 2.1.5). Fortunately, physicists had developed a formalism for describing quantum systems with variable particle number, namely second quantization, more than 80 years ago. For the convenience of the reader, we give a brief introduction to the second quantization method in this section. This introduction focuses on the mathematical formulation of second quantization needed in the sequent sections; for physical interpretations, the reader can consult reference [163].

7.3.1 MULTIPLE-PARTICLE STATES

We first consider a quantum system of a fixed number of particles. It is assumed that these particles have the same state Hilbert space, but they are not necessarily identical particles. Let $\mathcal{H}$ be the state Hilbert space of a single particle. For any $n \geq 1$, we can define the tensor product $\mathcal{H}^{\otimes n}$ of n copies of $\mathcal{H}$ by Definition 2.1.18. For any family $|\psi_1\rangle, \dots, |\psi_n\rangle$ of single-particle states in $\mathcal{H}$, the Postulate of quantum mechanics 4 asserts that we have a state

$$|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle = |\psi_1 \otimes \cdots \otimes \psi_n\rangle$$

of n independent particles, in which the ith particle is in state $|\psi_i\rangle$ for every $1 \leq i \leq n$. Then $\mathcal{H}^{\otimes n}$ consists of the linear combinations of vectors $|\psi_1 \otimes \cdots \otimes \psi_n\rangle$:

$$\begin{aligned}\mathcal{H}^{\otimes n} &= span\{|\psi_1 \otimes \cdots \otimes \psi_n\rangle : |\psi_1\rangle, \dots, |\psi_n\rangle \in \mathcal{H}\} \\ &= \left\{\sum_{i=1}^{m} \alpha_i |\psi_{i1} \otimes \cdots \otimes \psi_{in}\rangle : m \geq 0, \alpha_i \in \mathbb{C}, |\psi_{i1}\rangle, \dots, |\psi_{in}\rangle \in \mathcal{H}\right\}.\end{aligned}$$

Recall from Subsection 2.1.5 that $\mathcal{H}^{\otimes n}$ is a Hilbert space too. More explicitly, the basic operations of vectors in $\mathcal{H}^{\otimes n}$ are defined by the following equations together with linearity:

(i) Addition:

$$|\psi_1 \otimes \cdots \otimes \psi_i \otimes \cdots \otimes \psi_n\rangle + |\psi_1 \otimes \cdots \otimes \psi_i' \otimes \cdots \otimes \psi_n\rangle$$
$$= |\psi_1 \otimes \cdots \otimes (\psi_i + \psi_i') \otimes \cdots \otimes \psi_n\rangle;$$

(ii) Scalar product:

$$\lambda|\psi_1 \otimes \cdots \otimes \psi_i \otimes \cdots \otimes \psi_n\rangle = |\psi_1 \otimes \cdots \otimes (\lambda\psi_i) \otimes \cdots \otimes \psi_n\rangle;$$

(iii) Inner product:

$$\langle\psi_1 \otimes \cdots \otimes \psi_n|\varphi_1 \otimes \cdots \otimes \varphi_n\rangle = \prod_{i=1}^{n}\langle\psi_i|\varphi_i\rangle.$$

Exercise 7.3.1. *Show that if $\mathcal{B}$ is a basis of $\mathcal{H}$, then*

$$\{|\psi_1 \otimes \cdots \otimes \psi_n\rangle : |\psi_1\rangle, \ldots, |\psi_n\rangle \in \mathcal{B}\}$$

is a basis of $\mathcal{H}^{\otimes n}$.

Permutation Operators:

Now we turn to consider a quantum system of multiple identical particles that possess the same intrinsic properties. Let us start by introducing several operators for describing the symmetry of identical particles. For each permutation π of $1, \ldots, n$, i.e., a bijection from $\{1, \ldots, n\}$ onto itself that maps i to $\pi(i)$ for every $1 \le i \le n$, we can define the permutation operator P_π in the space $\mathcal{H}^{\otimes n}$ by

$$P_\pi|\psi_1 \otimes \cdots \otimes \psi_n\rangle = |\psi_{\pi(1)} \otimes \cdots \otimes \psi_{\pi(n)}\rangle$$

together with linearity. Several basic properties of permutation operators are given in the following:

Proposition 7.3.1

(i) *P_π is a unitary operator.*

(ii) *$P_{\pi_1}P_{\pi_2} = P_{\pi_1\pi_2}$, $P_\pi^\dagger = P_{\pi^{-1}}$, where $\pi_1\pi_2$ is the composition of π_1 and π_2, $P_\pi^\dagger$ stands for the conjugate transpose (i.e., inverse) of P_π, and π^{-1} is the inverse of π.*

Furthermore, symmetrization and antisymmetrization operators can be defined in terms of permutation operators:

Definition 7.3.1. *The symmetrization and antisymmetrization operators in $\mathcal{H}^{\otimes n}$ are defined by*

$$S_+ = \frac{1}{n!}\sum_\pi P_\pi,$$
$$S_- = \frac{1}{n!}\sum_\pi (-1)^\pi P_\pi,$$

where π traverses over all permutations of $1, \ldots, n$, and $(-1)^\pi$ is the signature of the permutation π; that is,

$$(-1)^\pi = \begin{cases} 1 & \text{if } \pi \text{ is even,} \\ -1 & \text{if } \pi \text{ is odd.} \end{cases}$$

We list several useful properties of symmetrization and antisymmetrization in the following:

Proposition 7.3.2

(i) $P_\pi S_+ = S_+ P_\pi = S_+$.
(ii) $P_\pi S_- = S_- P_\pi = (-1)^\pi S_-$.
(iii) $S_+^2 = S_+ = S_+^\dagger$.
(iv) $S_-^2 = S_- = S_-^\dagger$.
(v) $S_+ S_- = S_- S_+ = 0$.

Exercise 7.3.2. *Prove Propositions 7.3.1 and 7.3.2.*

Symmetric and Antisymmetric States:

Of course, quantum mechanics described in Section 2.1 can be used to cope with a quantum system of multiple particles. However, it is not complete when these particles are identical, and it must be supplemented by the following:

- **The principle of symmetrization:** The states of n identical particles are either completely symmetric or completely antisymmetric with the permutation of the n particles.
  - The symmetric particles are called *bosons*;
  - The antisymmetric particles are called *fermions*.

At the beginning of this subsection, we saw that $\mathcal{H}^{\otimes n}$ is the state Hilbert space of n particles if all of them have the same state space $\mathcal{H}$. According to the preceding principle, it is not the case that every vector in $\mathcal{H}^{\otimes n}$ can be used to denote a state of n identical particles. However, for each state $|\Psi\rangle$ in $\mathcal{H}^{\otimes n}$, we can construct the following two states by symmetrization or antisymmetrization:

- a symmetric (bosonic) state: $S_+|\Psi\rangle$;
- an antisymmetric (a fermionic) state: $S_-|\Psi\rangle$.

In particular, the symmetric and antisymmetric states corresponding to the product of one-particle states $|\psi_1\rangle, \ldots, |\psi_n\rangle$ are written as

$$|\psi_1, \cdots, \psi_n\rangle_v = S_v|\psi_1 \otimes \cdots \otimes \psi_n\rangle_v$$

where v is $+$ or $-$. With this notation, the principle of symmetrization can be restated as follows:

- For the bosons, the order of states $|\psi_i\rangle$ in $|\psi_1, \cdots, \psi_n\rangle_+$ is insignificant.
- For the fermions, $|\psi_1, \cdots, \psi_n\rangle_-$ changes sign under permutations of two states:

$$|\psi_1, \cdots, \psi_i, \cdots, \psi_j, \cdots, \psi_n\rangle_- = -|\psi_1, \cdots, \psi_j, \cdots, \psi_i, \cdots, \psi_n\rangle_-. \tag{7.11}$$

An immediate corollary of equation (7.11) is the following:

- **Pauli's exclusion principle:** If two states $|\psi_i\rangle$ and $|\psi_j\rangle$ are identical, then $|\psi_1, \cdots, \psi_i, \cdots, \psi_j, \cdots, \psi_n\rangle_-$ vanishes – two fermions can never be found in the same individual quantum state.

In summary, the principle of symmetrization implies that the state space of a system of n identical particles is not the total of $\mathcal{H}^{\otimes n}$, but rather one of the following two subspaces:

Definition 7.3.2

(i) *The n-fold symmetric tensor product of $\mathcal{H}$:*

$$\begin{aligned}\mathcal{H}_+^{\otimes n} &= S_+\left(\mathcal{H}^{\otimes n}\right)\\ &= \textit{the closed subspace of } \mathcal{H}^{\otimes n} \textit{ generated by the symmetric}\\ &\quad\ \textit{tensor products } |\psi_1, \cdots, \psi_n\rangle_+ \textit{ with } |\psi_1\rangle, \ldots, |\psi_n\rangle \in \mathcal{H}\end{aligned}$$

(ii) *The n-fold antisymmetric tensor product of $\mathcal{H}$:*

$$\begin{aligned}\mathcal{H}_-^{\otimes n} &= S_-\left(\mathcal{H}^{\otimes n}\right)\\ &= \textit{the closed subspace of } \mathcal{H}^{\otimes n} \textit{ generated by the antisymmetric}\\ &\quad\ \textit{tensor products } |\psi_1, \cdots, \psi_n\rangle_- \textit{ with } |\psi_1\rangle, \ldots, |\psi_n\rangle \in \mathcal{H}\end{aligned}$$

The addition, scalar product and inner product in $\mathcal{H}_v^{\otimes n}$ $(v = +, -)$ are directly inherited from $\mathcal{H}^{\otimes n}$. In particular, the following proposition provides a convenient way to compute the inner products in $\mathcal{H}_\pm^{\otimes n}$:

Proposition 7.3.3. *The inner product of symmetric and antisymmetric tensor products:*

$$\begin{aligned}{}_+\langle\psi_1, \cdots, \psi_n|\varphi_1, \cdots, \varphi_n\rangle_+ &= \frac{1}{n!} per\left(\langle\psi_i|\varphi_j\rangle\right)_{ij},\\ {}_-\langle\psi_1, \cdots, \psi_n|\varphi_1, \cdots, \varphi_n\rangle_- &= \frac{1}{n!} det\left(\langle\psi_i|\varphi_j\rangle\right)_{ij},\end{aligned}$$

where det and per stand for the determinant of matrix and the permutation (i.e., the determinant without the minus signs), respectively.

Exercise 7.3.3

(i) *Compute the dimensions of the symmetric and antisymmetric tensor product spaces $\mathcal{H}_\pm^{\otimes n}$.*

(ii) *Prove Proposition 7.3.3.*

7.3.2 FOCK SPACES

Quantum systems of a fixed number of identical particles were studied in the last subsection. Now let us see how to describe a quantum system with a variable number of particles. A natural idea is that the state Hilbert space of such a system is the direct sum of the state spaces of different numbers of particles. To realize this idea, let us first introduce the notion of direct sum of Hilbert spaces.

Definition 7.3.3. *Let $\mathcal{H}_1, \mathcal{H}_2, \ldots$ be an infinite sequence of Hilbert spaces. Then their direct sum is defined to be the vector space:*

$$\bigoplus_{i=1}^{\infty}\mathcal{H}_i = \left\{(|\psi_1\rangle, |\psi_2\rangle, \ldots) : |\psi_i\rangle \in \mathcal{H}_i \ (i = 1, 2, \ldots) \ \textit{with} \ \sum_{i=1}^{\infty} ||\psi_i||^2 < \infty\right\}$$

in which we define:

- *Addition:*

$$(|\psi_1\rangle, |\psi_2\rangle, \ldots) + (|\varphi_1\rangle, |\varphi_2\rangle, \ldots) = (|\psi_1\rangle + |\varphi_1\rangle, |\psi_2\rangle + |\varphi_2\rangle, \ldots);$$

- *Scalar multiplication:*

$$\alpha(|\psi_1\rangle, |\psi_2\rangle, \ldots) = (\alpha|\psi_1\rangle, \alpha|\psi_2\rangle, \ldots);$$

- *Inner product:*

$$\langle(\psi_1, \psi_2, \ldots)|(\varphi_1, \varphi_2, \ldots)\rangle = \sum_{i=1}^{\infty}\langle\psi_i|\varphi_i\rangle.$$

Exercise 7.3.4. *Show that $\bigoplus_{i=1}^{\infty}\mathcal{H}_i$ is a Hilbert space.*

Let $\mathcal{H}$ be the state Hilbert space of one particle. If we introduce the vacuum state $|\mathbf{0}\rangle$, then the 0-fold tensor product of $\mathcal{H}$ can be defined as the one-dimensional space

$$\mathcal{H}^{\otimes 0} = \mathcal{H}_{\pm}^{\otimes 0} = span\{|\mathbf{0}\rangle\}.$$

Now we are ready to describe the state space of a quantum system with a variable number of identical particles.

Definition 7.3.4

(i) *The free Fock space over $\mathcal{H}$ is defined to be the direct sum of the n-fold tensor products of $\mathcal{H}$:*

$$\mathcal{F}(\mathcal{H}) = \bigoplus_{n=0}^{\infty}\mathcal{H}^{\otimes n}.$$

(ii) *The symmetric (bosonic) Fock space and the antisymmetric (fermionic) Fock space over $\mathcal{H}$ are defined by*

$$\mathcal{F}_v(\mathcal{H}) = \bigoplus_{n=0}^{\infty} \mathcal{H}_v^{\otimes n}$$

where $v = +$ for bosons or $v = -$ for fermions.

The principle of symmetrization tells us that only the symmetric or antisymmetric Fock space is meaningful in physics, but here we also introduce the free Fock space since it is a useful mathematical tool that is sometimes easier to deal with than the symmetric and antisymmetric Fock spaces, as we will see in the next section.

To understand the Fock spaces better, let us look more carefully at the states and operations in them:

(i) A state in $\mathcal{F}_v(\mathcal{H})$ is of the form:

$$|\Psi\rangle = \sum_{n=0}^{\infty} |\Psi(n)\rangle \overset{\triangle}{=} (|\Psi(0)\rangle, |\Psi(1)\rangle, \cdots, |\Psi(n)\rangle, \cdots)$$

where $|\Psi(n)\rangle \in \mathcal{H}_v^{\otimes n}$ is a state of n particles for all $n = 0, 1, 2, \ldots$, and

$$\sum_{n=0}^{\infty} \langle \Psi(n)|\Psi(n)\rangle < \infty.$$

(ii) Basic operations in $\mathcal{F}_v(\mathcal{H})$:

- Addition:

$$\left(\sum_{n=0}^{\infty} |\Psi(n)\rangle\right) + \left(\sum_{n=0}^{\infty} |\Phi(n)\rangle\right) = \sum_{n=0}^{\infty} (|\Psi(n)\rangle + |\Phi(n)\rangle);$$

- Scalar product:

$$\alpha \left(\sum_{n=0}^{\infty} |\Psi(n)\rangle\right) = \sum_{n=0}^{\infty} \alpha |\Psi(n)\rangle;$$

- Inner product:

$$\left\langle \sum_{n=0}^{\infty} \Psi(n) \middle| \sum_{n=0}^{\infty} \Phi(n) \right\rangle = \sum_{n=0}^{\infty} \langle \Psi(n)|\Phi(n)\rangle.$$

(iii) Basis of $\mathcal{F}_v(\mathcal{H})$: The symmetric or antisymmetric product states $|\psi_1, \ldots, \psi_n\rangle_v$ ($n \geq 0$ and $|\psi_1\rangle, \ldots, |\psi_n\rangle \in \mathcal{H}$) form a basis of Fock space $\mathcal{F}_v(\mathcal{H})$; that is,

$$\mathcal{F}_v(\mathcal{H}) = span\{|\psi_1, \cdots, \psi_n\rangle_v : n = 0, 1, 2, \ldots \text{ and } |\psi_1\rangle, \ldots, |\psi_n\rangle \in \mathcal{H}\}$$

where $|\psi_1, \cdots, \psi_n\rangle_v$ is the vacuum state $|\mathbf{0}\rangle$ if $n = 0$.

(iv) We identify state $(0,\ldots,0,|\Psi(n)\rangle,0,\ldots,0)$ in $\mathcal{F}_v(\mathcal{H})$ with the state $|\Psi(n)\rangle$ in $\mathcal{H}_v^{\otimes n}$. Then $\mathcal{H}_v^{\otimes n}$ can be seen as a subspace of $\mathcal{F}_v(\mathcal{H})$. Moreover, for different particle numbers $m \neq n$, $\mathcal{H}_v^{\otimes m}$ and $\mathcal{H}_v^{\otimes n}$ are orthogonal because $\langle\Psi(m)|\Psi(n)\rangle = 0$.

Operators in Fock Spaces:

We already learned that a state of a quantum system with a variable number of identical particles can be represented by a vector in the Fock spaces. Now we move forward to prepare the mathematical tools for the description of observables and evolution of such a quantum system. We first define operators in the direct sum of Hilbert spaces.

Definition 7.3.5. *For each $i \geq 1$, let A_i be a bounded operator on $\mathcal{H}_i$ such that the sequence $\|A_i\|$ $(i = 1,2,\ldots)$ of norms (see Definition 2.1.11) is bounded; that is, $\|A_i\| \leq C$ $(i = 1,2,\ldots)$ for some constant C. Then we define operator*

$$A = (A_1, A_2, \ldots)$$

in $\bigoplus_{i=1}^{\infty}\mathcal{H}_i$ as follows:

$$A(|\psi_1\rangle,|\psi_2\rangle,\ldots) = (A_1|\psi_1\rangle, A_2|\psi_2\rangle,\ldots) \tag{7.12}$$

for every $(|\psi_1\rangle,|\psi_2\rangle,\ldots) \in \bigoplus_{i=1}^{\infty}\mathcal{H}_i$.

Similarly, we can define operator $A = (A_1,\ldots,A_n)$ in the direct sum $\bigoplus_{i=1}^{n}\mathcal{H}_i$ for a finite number n.

We often write:

$$\sum_{i=1}^{\infty} A_i = (A_1, A_2, \ldots) \text{ and } \sum_{i=1}^{n} A_i = (A_1, \ldots, A_n).$$

Exercise 7.3.5. *Show that A defined by equation (7.12) is a bounded operator in $\bigoplus_{i=1}^{\infty}\mathcal{H}_i$ and*

$$\|A\| \leq \sup_{i=1}^{\infty}\|A_i\|.$$

Now we can use this idea to define operators in Fock spaces. For each $n \geq 1$, let $\mathbf{A}(n)$ be an operator in $\mathcal{H}^{\otimes n}$. Then operator

$$\mathbf{A} = \sum_{n=0}^{\infty}\mathbf{A}(n) \tag{7.13}$$

can be defined in the free Fock space $\mathcal{F}(\mathcal{H})$ by Definition 7.3.5:

$$\mathbf{A}|\Psi\rangle = \mathbf{A}\left(\sum_{n=0}^{\infty}|\Psi(n)\rangle\right) = \sum_{n=0}^{\infty}\mathbf{A}(n)|\Psi(n)\rangle$$

for any

$$|\Psi\rangle = \sum_{n=0}^{\infty} |\Psi(n)\rangle$$

in $\mathcal{F}(\mathcal{H})$, where $\mathbf{A}|\mathbf{0}\rangle = 0$; that is, the vacuum state is considered to be an eigenvector of operator $\mathbf{A}$ with eigenvalue 0. Obviously, for all $n \geq 0$, $\mathcal{H}_v^{\otimes n}$ is invariant under $\mathbf{A}$; that is, $\mathbf{A}(\mathcal{H}_v^{\otimes n}) \subseteq \mathcal{H}_v^{\otimes n}$.

A general operator in the form of (7.13) does not necessarily preserve symmetry (respectively, antisymmetry) of bosons (respectively, fermions). To define an operator in the bosonic or fermionic Fock space, we need to consider its symmetry.

Definition 7.3.6. *If for each $n \geq 0$ and for each permutation π of $1, \ldots, n$, P_π and $\mathbf{A}(n)$ commute; that is,*

$$P_\pi \mathbf{A}(n) = \mathbf{A}(n) P_\pi,$$

then operator $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$ is said to be symmetric.

It is easy to see that both the symmetric Fock space $\mathcal{F}_+(\mathcal{H})$ and the antisymmetric Fock space $\mathcal{F}_-(\mathcal{H})$ are closed under a symmetric operator $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$:

$$\mathbf{A}(\mathcal{F}_v(\mathcal{H})) \subseteq \mathcal{F}_v(\mathcal{H})$$

for $v = +, -$. In other words, a symmetric operator $\mathbf{A}$ maps a bosonic (or fermionic) state $|\Psi\rangle$ to a bosonic (respectively, fermionic) state $\mathbf{A}|\Psi\rangle$.

Exercise 7.3.6. *Is the following statement correct: if an operator $\mathbf{A}$ in $\mathcal{F}(\mathcal{H})$ satisfies that $\mathbf{A}(\mathcal{F}_+(\mathcal{H})) \subseteq \mathcal{F}_+(\mathcal{H})$ (or $\mathbf{A}(\mathcal{F}_-(\mathcal{H})) \subseteq \mathcal{F}_-(\mathcal{H})$), then $\mathbf{A}$ is symmetric? Prove your conclusion.*

We can further introduce the symmetrization functional $\mathbb{S}$ that maps every operator $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$ to a symmetric operator:

$$\mathbb{S}(\mathbf{A}) = \sum_{n=0}^{\infty} \mathbb{S}(\mathbf{A}(n)) \tag{7.14}$$

where for each $n \geq 0$,

$$\mathbb{S}(\mathbf{A}(n)) = \frac{1}{n!} \sum_{\pi} P_\pi \mathbf{A}(n) P_\pi^{-1} \tag{7.15}$$

with π traversing over all permutations of $1, \ldots, n$, where P_π^{-1} is the inverse of P_π. Thus, each operator $\mathbf{A}$ in the free Fock space can be transformed by the symmetrization functional $\mathbb{S}$ to an operator $\mathbf{S}(\mathbf{A})$ that can be properly applied in the bosonic or fermionic Fock spaces.

7.3.3 OBSERVABLES IN FOCK SPACES

In the last subsection, Fock spaces were introduced as the state Hilbert spaces of quantum systems with variable particle number. We also studied various operators in the Fock spaces. Now we see how to describe the observables of these systems.

Many-Body Observables:

To warm up, let us start with the observables of a fixed number $n \geq 1$ of particles. By the basic postulates of quantum mechanics, in general, an observable of a quantum system with n particles can be expressed by a Hermitian operator in $\mathcal{H}^{\otimes n}$. Here, we like to carefully look at a very special class of observables in $\mathcal{H}^{\otimes n}$.

First, let us consider the simplest case where only one of the n particles is observed. Assume that O is a single-particle observable in $\mathcal{H}$. Then for each $1 \leq i \leq n$, the action $O^{[i]}$ of O on the ith factor of $\mathcal{H}^{\otimes n}$ can be defined by

$$O^{[i]}|\psi_1 \otimes \cdots \psi_i \otimes \cdots \otimes \psi_n\rangle = |\psi_1 \otimes \cdots \otimes (O\psi_i) \otimes \cdots \otimes \psi_n\rangle$$

together with linearity; that is,

$$O^{[i]} = I^{\otimes(i-1)} \otimes O \otimes I^{\otimes(n-i)},$$

where I is the identity operator in $\mathcal{H}$. Obviously, for a fixed i, the operator $O^{[i]}$ is not symmetric. Combining these actions $O^{[i]}$ on different particles i, we have:

Definition 7.3.7. *The one-body observable corresponding to O is*

$$O_1(n) = \sum_{i=1}^{n} O^{[i]}.$$

Secondly, we consider an observable on two of the n particles. Assume that O is an observable in the two-particle space $\mathcal{H} \otimes \mathcal{H}$. Then for any $1 \leq i < j \leq n$, we can define $O^{[ij]}$ in $\mathcal{H}^{\otimes n}$ as the operator that acts as O on the ith and jth factors of $\mathcal{H}^{\otimes n}$ and trivially on others; that is, the cylindrical extension of O in $\mathcal{H}^{\otimes n}$:

$$O^{[ij]} = O[i,j] \otimes \left(I^{\otimes(i-1)} \otimes I^{\otimes(j-i-1)} \otimes I^{\otimes(n-j)}\right).$$

If observable O is allowed to apply to any two of these n particles, we have:

Definition 7.3.8. *The two-body observable for the system of n particles corresponding to O is defined as*

$$O_2(n) = \sum_{1 \leq i < j \leq n} O^{[ij]}.$$

Exercise 7.3.7. *Show that if O is invariant under exchange of two particles:* $O^{[ij]} = O^{[ji]}$ *for all* $1 \leq i, j \leq n$*, then*

$$O_2(n) = \frac{1}{2} \sum_{i \neq j} O^{[ij]}.$$

Furthermore, we can define k-body observables $O_k(n)$ for $2 < k \leq n$ in a way similar to Definitions 7.3.7 and 7.3.8.

Observables in Fock Spaces:

The previous preparation enables us to study observables with a variable number of particles. Actually, equation (7.13) provides us with a natural way to define observables in Fock spaces; more precisely, if for each $n \geq 1$, the operator $\mathbf{A}(n)$ in equation (7.13) is an observable of n particles, then

$$\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$$

is called an extensive observable in the free Fock space $\mathcal{F}(\mathcal{H})$. In particular, if $\mathbf{A}$ is symmetric, then it is also an observable in both the symmetric and antisymmetric Fock spaces.

The following proposition gives a convenient method for computing the mean value of an extensive observable.

Proposition 7.3.4. *The mean value of* $\mathbf{A} = \sum_{n=0}^{\infty} \mathbf{A}(n)$ *in state* $|\Psi\rangle = \sum_{n=0}^{\infty} |\Psi(n)\rangle$ *is*

$$\begin{aligned}\langle\Psi|\mathbf{A}|\Psi\rangle &= \sum_{n=0}^{\infty} \langle\Psi(n)|\mathbf{A}(n)|\Psi(n)\rangle \\ &= \sum_{n=0}^{\infty} \langle\Psi(n)|\Psi(n)\rangle \cdot \frac{\langle\Psi(n)|\mathbf{A}(n)|\Psi(n)\rangle}{\langle\Psi(n)|\Psi(n)\rangle}\end{aligned}$$

where:

(i) $\langle\Psi(n)|\Psi(n)\rangle$ *is the probability to find n particles in the state* $|\Psi\rangle$*;*

(ii)

$$\frac{\langle\Psi(n)|\mathbf{A}(n)|\Psi(n)\rangle}{\langle\Psi(n)|\Psi(n)\rangle}$$

is the mean value of $\mathbf{A}(n)$ *for the system of n particles.*

To conclude this subsection, let us consider two special classes of observables in the Fock spaces. As an application of the previous procedure, for a given $k \geq 1$, the k-body observables in the free Fock space can be defined as follows:

$$\mathbf{O}_k = \sum_{n \geq k} O_k(n),$$

where for all $n \geq k$, $O_k(n)$ is the k-body observable in $\mathcal{H}^{\otimes n}$ (see Definitions 7.3.7 and 7.3.8). Furthermore, note that $\mathbf{O}_k$ is symmetric; for example, one-body observables $O_1(n)$ commute with the permutations:

$$O_1(n)|\psi_1,\ldots,\psi_n\rangle_\pm = \sum_{j=1}^{n}|\psi_1,\ldots,\psi_{j-1},O\psi_j,\psi_{j+1},\ldots,\psi_n\rangle_\pm.$$

A similar equation holds for $k \geq 2$. Therefore, $\mathbf{O}_k$ can be directly used in the symmetric and antisymmetric Fock spaces.

Another important observable in Fock spaces is defined in the following:

Definition 7.3.9. *The particle number operator N in $\mathcal{F}_\pm(\mathcal{H})$ is defined by*

$$\mathbf{N}\left(\sum_{n=0}^{\infty}|\Psi(n)\rangle\right) = \sum_{n=0}^{\infty} n|\Psi(n)\rangle.$$

As suggested by its name, for each $n \geq 0$, the particle number operator $\mathbf{N}$ gives the number n explicitly in front of the n-particle component $|\Psi(n)\rangle$ of $|\Psi\rangle$. Several basic properties of the particle number operator are given in the following two propositions:

Proposition 7.3.5

(i) *For each $n = 0, 1, 2\ldots$, $\mathcal{H}_\pm^{\otimes n}$ is the eigen-subspace of $\mathbf{N}$ with eigenvalue n.*
(ii) *The mean value of $\mathbf{N}$ in state $|\Psi\rangle = \sum_{n=0}^{\infty}|\Psi(n)\rangle$ is*

$$\langle\Psi|\mathbf{N}|\Psi\rangle = \sum_{n=0}^{\infty} n\langle\Psi(n)|\Psi(n)\rangle.$$

Proposition 7.3.6. *Extensive observables* $\mathbf{A}$ *and particle number operator* $\mathbf{N}$ *commute:* $\mathbf{AN} = \mathbf{NA}$.

Exercise 7.3.8. *Prove Propositions 7.3.5 and 7.3.6.*

7.3.4 EVOLUTION IN FOCK SPACES

We now turn to consider the dynamics of a quantum system with a variable particle number. Such a quantum system can evolve in two different ways:

(i) the evolution does not change the number of particles; and
(ii) the evolution changes the number of particles.

In this subsection, we focus on the first kind of evolution, and the second will be discussed in the next subsection. Obviously, equation (7.13) also gives us a method to define an evolution of the first kind. More precisely, if for every $n \geq 0$, the dynamics of n particles is modelled by operator $\mathbf{A}(n)$, then $\mathbf{A} = \sum_{n=0}^{\infty}\mathbf{A}(n)$ describes an evolution in the Fock spaces that does not change the number of particles. Let us more carefully examine a special evolution of this kind. Assume that the (discrete-time) evolution of one particle is represented by unitary operator U. Then the evolution of n particles without mutual interactions can be described by operator

$$\mathbf{U}(n) = U^{\otimes n}$$

in $\mathcal{H}^{\otimes n}$:

$$\mathbf{U}(n)|\psi_1 \otimes \ldots \otimes \psi_n\rangle = |U\psi_1 \otimes \ldots \otimes U\psi_n\rangle \tag{7.16}$$

for all $|\psi_1\rangle, \ldots, |\psi_n\rangle$ in $\mathcal{H}$. Here, the same unitary U applies to all of the n particles simultaneously. It is easy to verify that $\mathbf{U}(n)$ commutes with the permutations:

$$\mathbf{U}(n)|\psi_1, \ldots, \psi_n\rangle_\pm = |U\psi_1, \ldots, U\psi_n\rangle_\pm.$$

Then using equation (7.13), we can define a symmetric operator in the Fock spaces:

$$\mathbf{U} = \sum_{n=0}^{\infty} \mathbf{U}(n). \tag{7.17}$$

Clearly, it depicts the evolution of a quantum system with variable number of particles but without mutual interaction between the particles.

7.3.5 CREATION AND ANNIHILATION OF PARTICLES

In the previous subsection, we learned how to describe the first kind of evolution in the Fock spaces, where the number of particles is not changed; for example, the operator $\mathbf{U}$ defined by equation (7.17) maps the states of n particles to states of particles of the same number. In this subsection, we study the second kind of evolution in the Fock spaces that can change the number of particles. Obviously, this kind of evolution cannot be treated within the basic framework of quantum mechanics presented in Section 2.1. Nevertheless, the transitions between states of different particle numbers can be described by two basic operators: creation and annihilation.

Definition 7.3.10. *For each one-particle state $|\psi\rangle$ in $\mathcal{H}$, the creation operator $a^\dagger(\psi)$ associated with $|\psi\rangle$ in $\mathcal{F}_\pm(\mathcal{H})$ is defined by*

$$a^\dagger(\psi)|\psi_1, \ldots, \psi_n\rangle_v = \sqrt{n+1}|\psi, \psi_1, \ldots, \psi_n\rangle_v \tag{7.18}$$

for any $n \geq 0$ and all $|\psi_1\rangle, \ldots, |\psi_n\rangle$ in $\mathcal{H}$, together with linearity.

From its defining equation (7.18), we see that the operator $a^\dagger(\psi)$ adds a particle in the individual state $|\psi\rangle$ to the system of n particles without modifying their respective states; in particular, the symmetry or antisymmetry of the state is preserved in this transition. The coefficient $\sqrt{n+1}$ is added in the right-hand side of equation (7.18) mainly for the sake of coefficient normalization.

Definition 7.3.11. *For each one-particle state $|\psi\rangle$ in $\mathcal{H}$, the annihilation operator $a(\psi)$ in $\mathcal{F}_\pm(\mathcal{H})$ is defined to be the Hermitian conjugate of $a^\dagger(\psi)$:*

$$a(\psi) = (a^\dagger(\psi))^\dagger$$

that is,

$$\left(a^\dagger(\psi)|\varphi_1, \ldots, \varphi_n\rangle_v, |\psi_1, \ldots, \psi_n\rangle_v\right) = (|\varphi_1, \ldots, \varphi_n\rangle_v, a(\psi)|\psi_1, \ldots, \psi_n\rangle_v) \tag{7.19}$$

for all $|\varphi_1\rangle, \ldots, |\varphi_n\rangle, |\psi_1\rangle, \ldots, |\psi_n\rangle \in \mathcal{H}$ and for every $n \geq 0$.

The following proposition gives a representation of annihilation operators.

Proposition 7.3.7

$$a(\psi)|\mathbf{0}\rangle = 0,$$
$$a(\psi)|\psi_1,\cdots,\psi_n\rangle_\pm = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}(v)^i\langle\psi|\psi_i\rangle|\psi_1,\cdots,\psi_{i-1},\psi_{i+1},\cdots,\psi_n\rangle_\pm.$$

It can be clearly seen from this proposition that annihilation operator $a(\psi)$ decreases the number of particles by one unit, while preserving the symmetry of the state.

Exercise 7.3.9. *Prove Proposition 7.3.7. Hint: use the defining equation (7.19) of* $a(\psi)$.

7.4 SOLVING RECURSIVE EQUATIONS IN THE FREE FOCK SPACE

As convinced by an example of a recursive quantum walk at the end of Section 7.2, we have to deal with a quantum system with a variable number of identical particles in order to model the behavior of quantum "coins" employed in the execution of a quantum recursive program. So, a mathematical framework for this purpose, namely second quantization, was introduced in the last section. Indeed, second quantization provides us with all of the necessary tools for defining the semantics of quantum recursion. The purpose of this and the next sections is to carefully define the semantics of quantum recursive programs. This will be done in two steps. In this section, we show how to solve recursive equations in the free Fock spaces without considering symmetry or antisymmetry of the particles that are used to implement the quantum "coins."

7.4.1 A DOMAIN OF OPERATORS IN THE FREE FOCK SPACE

Similar to the cases of recursive classical programs and classical recursion of quantum programs studied in Section 3.4, we first need to set the stage – a domain in which we can accommodate the solutions of quantum recursive equations. In this subsection, we study such a domain as an abstract mathematical object, leaving its applications to semantics of recursive quantum programs aside. This focus can give us a better understanding of the structure of the domain.

Let C be a set of quantum "coins." For each $c \in C$, let $\mathcal{H}_c$ be the state Hilbert space of "coin" c and $\mathcal{F}(\mathcal{H}_c)$ the free Fock space over $\mathcal{H}_c$. We write

$$\mathcal{G}(\mathcal{H}_C) \triangleq \bigotimes_{c\in C}\mathcal{F}(\mathcal{H}_c)$$

for the tensor product of the free Fock spaces of all "coins." We also assume that $\mathcal{H}$ is the state Hilbert space of the principal system. Then the state space of the quantum system combining the principal system with variable numbers of "coins" is $\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}$.

Let ω be the set of nonnegative integers. Then ω^C is the set of C-indexed tuples of nonnegative integers: $\overline{n} = \{n_c\}_{c \in C}$ with $n_c \in \omega$ for all $c \in C$. It is clear that

$$\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H} = \bigoplus_{\overline{n} \in \omega^C} \left[\left(\bigotimes_{c \in C} \mathcal{H}_c^{\otimes n_c} \right) \otimes \mathcal{H} \right]$$

Furthermore, let $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$ be the set of all operators of the form

$$\mathbf{A} = \sum_{\overline{n} \in \omega^C} \mathbf{A}(\overline{n}),$$

where $\mathbf{A}(\overline{n})$ is an operator in $\left(\bigotimes_{c \in C} \mathcal{H}_c^{\otimes n_c} \right) \otimes \mathcal{H}$ for each $\overline{n} \in \omega^C$. Then $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$ will serve as the space of the solutions of quantum recursive equations. To present the partial order in $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$ needed for solving quantum recursive equations, we first define a partial order $\leq$ in ω^C as follows:

- $\overline{n} \leq \overline{m}$ if and only if $n_c \leq m_c$ for all $c \in C$.

A subset $\Omega \subseteq \omega^C$ is said to be *below-closed* if $\overline{n} \in \Omega$ and $\overline{m} \leq \overline{n}$ imply $\overline{m} \in \Omega$.

Definition 7.4.1. *The flat order $\sqsubseteq$ on $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$ is defined as follows: for any $\mathbf{A} = \sum_{\overline{n} \in \omega^C} \mathbf{A}(\overline{n})$ and $\mathbf{B} = \sum_{\overline{n} \in \omega^C} \mathbf{B}(\overline{n})$ in $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$,*

- $\mathbf{A} \sqsubseteq \mathbf{B}$ *if and only if there exists a below-closed subset $\Omega \subseteq \omega^C$ such that*
  - $\mathbf{A}(\overline{n}) = \mathbf{B}(\overline{n})$ *for all $\overline{n} \in \Omega$; and*
  - $\mathbf{A}(\overline{n}) = 0$ *for all $\overline{n} \in \omega^C \setminus \Omega$.*

As said previously, the flat order can be simply understood as an abstract mathematical object. But our intention behind it has to be explained along with its application to the semantics of quantum recursive programs. A quantum recursive program P is executed by repeated substitutions, in each of which new "coins" should be added in order to avoid variable conflict, as shown in the example at the end of Section 7.2. For each $\overline{n} \in \omega^C$, n_c is used to record the number of copies of "coin" c employed in the computation of P. So, $\overline{n} \leq \overline{m}$ indicates that the (copies of) "coins" denoted by $\overline{m}$ are more than that denoted by $\overline{n}$. Obviously, the more "coins" are used in the execution of P, the more "content" is computed. Thus, if $\mathbf{A}$ and $\mathbf{B}$ are the partial computational results of P at two different stages of execution, then $\mathbf{A} \sqsubseteq \mathbf{B}$ means that the computed "content" at the stage of $\mathbf{B}$ is more than that at the stage of $\mathbf{A}$. This explanation will become clearer after reading Proposition 7.4.1.

The following is a key lemma in this chapter that unveils the lattice-theoretic structure of operators in the free Fock space.

Lemma 7.4.1. $(\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$ *is a complete partial order (CPO) (see Definition 3.3.4).*

Proof. First, $\sqsubseteq$ is reflexive because ω^C itself is below-closed. To show that $\sqsubseteq$ is transitive, we assume that $\mathbf{A} \sqsubseteq \mathbf{B}$ and $\mathbf{B} \sqsubseteq \mathbf{C}$. Then there exist below-closed subsets $\Omega, \Gamma \subseteq \omega^C$ such that

(i) $\mathbf{A}(\overline{n}) = \mathbf{B}(\overline{n})$ for all $\overline{n} \in \Omega$ and $\mathbf{A}(\overline{n}) = 0$ for all $\overline{n} \in \omega^C \setminus \Omega$;
(ii) $\mathbf{B}(\overline{n}) = \mathbf{C}(\overline{n})$ for all $\overline{n} \in \Gamma$ and $\mathbf{B}(\overline{n}) = 0$ for all $\overline{n} \in \omega^C \setminus \Gamma$.

Clearly, $\Omega \cap \Gamma$ is below-closed too, and $\mathbf{A}(\overline{n}) = \mathbf{B}(\overline{n}) = \mathbf{C}(\overline{n})$ for all $\overline{n} \in \Omega \cap \Gamma$. On the other hand, if

$$\overline{n} \in \omega^C \setminus (\Omega \cap \Gamma) = (\omega^C \setminus \Omega) \cup [\Omega \cap (\omega^C \setminus \Gamma)],$$

then:

- either $\overline{n} \in \omega^C \setminus \Omega$ and it follows from clause (i) that $\mathbf{A}(\overline{n}) = 0$;
- or $\overline{n} \in \Omega \cap (\omega^C \setminus \Gamma)$ and by combining clauses (i) and (ii) we obtain $\mathbf{A}(\overline{n}) = \mathbf{B}(\overline{n}) = 0$.

Therefore, $\mathbf{A} \sqsubseteq \mathbf{C}$. Similarly, we can prove that $\sqsubseteq$ is antisymmetric. So, $(\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$ is a partial order.

Obviously, the operator $\mathbf{A} = \sum_{\overline{n} \in \omega^C} \mathbf{A}(\overline{n})$ with $\mathbf{A}(\overline{n}) = 0$ (the zero operator in $\left(\bigotimes_{c \in C} \mathcal{H}_c^{\otimes n_c}\right) \otimes \mathcal{H}$) for all $\overline{n} \in \omega^C$ is the least element of $(\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$. Now it suffices to show that any chain $\{\mathbf{A}_i\}$ in $(\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$ has the least upper bound. For each i, we put

$$\Delta_i = \{\overline{n} \in \omega^C : \mathbf{A}_i(\overline{n}) \neq 0\},$$
$$\Delta_i \downarrow = \{\overline{m} \in \omega^C : \overline{m} \leq \overline{n} \text{ for some } \overline{n} \in \Delta_i\}.$$

Here $\Delta_i \downarrow$ is the below-completion of Δ_i. Furthermore, we define operator $\mathbf{A} = \sum_{\overline{n} \in \omega^C} \mathbf{A}(\overline{n})$ as follows:

$$\mathbf{A}(\overline{n}) = \begin{cases} \mathbf{A}_i(\overline{n}) & \text{if } \overline{n} \in \Delta_i \downarrow \text{ for some } i, \\ 0 & \text{if } \overline{n} \notin \bigcup_i (\Delta_i \downarrow). \end{cases}$$

- *Claim* 1: $\mathbf{A}$ is well-defined; that is, if $\overline{n} \in \Delta_i \downarrow$ and $\overline{n} \in \Delta_j \downarrow$, then $\mathbf{A}_i(\overline{n}) = \mathbf{A}_j(\overline{n})$. In fact, since $\{\mathbf{A}_i\}$ is a chain, we have $\mathbf{A}_i \sqsubseteq \mathbf{A}_j$ or $\mathbf{A}_j \sqsubseteq \mathbf{A}_i$. We only consider the case of $\mathbf{A}_i \sqsubseteq \mathbf{A}_j$ (the case of $\mathbf{A}_j \sqsubseteq \mathbf{A}_i$ can be proved by duality). Then there exists a below-closed subset $\Omega \subseteq \omega^C$ such that $\mathbf{A}_i(\overline{n}) = \mathbf{A}_j(\overline{n})$ for all $\overline{n} \in \Omega$ and $\mathbf{A}_i(\overline{n}) = 0$ for all $\overline{n} \in \omega^C \setminus \Omega$. It follows from $\overline{n} \in \Delta_i \downarrow$ that $\overline{n} \leq \overline{m}$ for some $\overline{m}$ with $\mathbf{A}_i(\overline{m}) \neq 0$. Since $\overline{m} \notin \omega^C \setminus \Omega$, i.e. $\overline{m} \in \Omega$, we have $\overline{n} \in \Omega$ because Ω is below-closed. So, $\mathbf{A}_i(\overline{n}) = \mathbf{A}_j(\overline{n})$.
- *Claim* 2: $\mathbf{A} = \bigsqcup_i \mathbf{A}_i$. In fact, for each i, $\Delta_i \downarrow$ is below-closed, and $\mathbf{A}_i(\overline{n}) = \mathbf{A}(\overline{n})$ for all $\overline{n} \in \Delta_i \downarrow$ and $\mathbf{A}_i(\overline{n}) = 0$ for all $\overline{n} \in \omega^C \setminus (\Delta_i \downarrow)$. So, $\mathbf{A}_i \sqsubseteq \mathbf{A}$, and $\mathbf{A}$ is an upper bound of $\{\mathbf{A}_i\}$. Now assume that $\mathbf{B}$ is an upper bound of $\{\mathbf{A}_i\}$: for all i,

$\mathbf{A}_i \sqsubseteq \mathbf{B}$; that is, there exists below-closed $\Omega_i \subseteq \omega^C$ such that $\mathbf{A}_i(\overline{n}) = \mathbf{B}(\overline{n})$ for all $\overline{n} \in \Omega_i$ and $\mathbf{A}_i(\overline{n}) = 0$ for all $\overline{n} \in \omega^C \setminus \Omega_i$. By the definition of Δ_i and below-closeness of Ω_i, we know that $\Delta_i \downarrow \subseteq \Omega_i$. We take

$$\Omega = \bigcup_i (\Delta_i \downarrow).$$

Clearly, Ω is below-closed, and if $\overline{n} \in \omega^C \setminus \Omega$, then $\mathbf{A}(\overline{n}) = 0$. On the other hand, if $\overline{n} \in \Omega$, then for some i, we have $\overline{n} \in \Delta_i \downarrow$, and it follows that $\overline{n} \in \Omega_i$ and $\mathbf{A}(\overline{n}) = \mathbf{A}_i(\overline{n}) = \mathbf{B}(\overline{n})$. Therefore, $\mathbf{A} \sqsubseteq \mathbf{B}$. □

The previous lemma presented the lattice-theoretical structure of $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$. Now we further define two algebraic operations in $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$, namely product and guarded composition. These two operations will be used to define the semantics of sequential compositions and quantum case statements in quantum program schemes.

Definition 7.4.2. *For any operators $\mathbf{A} = \sum_{\overline{n} \in \omega^C} \mathbf{A}(\overline{n})$ and $\mathbf{B} = \sum_{\overline{n} \in \omega^C} \mathbf{B}(\overline{n})$ in $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$, their product is defined as*

$$\mathbf{A} \cdot \mathbf{B} = \sum_{\overline{n} \in \omega^C} (\mathbf{A}(\overline{n}) \cdot \mathbf{B}(\overline{n})), \tag{7.20}$$

which is also in $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$.

This definition is a component-wise extension of ordinary operator product: for each $\overline{n} \in \omega^C$, $\mathbf{A}(\overline{n}) \cdot \mathbf{B}(\overline{n})$ is the product of operators $\mathbf{A}(\overline{n})$ and $\mathbf{B}(\overline{n})$ in $\left(\bigotimes_{c \in C} \mathcal{H}_c^{n_c}\right) \otimes \mathcal{H}$. The guarded composition of operators in the free Fock space can be defined by simply extending equation (7.1) in the same way.

Definition 7.4.3. *Let $c \in C$ and $\{|i\rangle\}$ be an orthonormal basis of $\mathcal{H}_c$, and let $\mathbf{A}_i = \sum_{\overline{n} \in \omega^C} \mathbf{A}_i(\overline{n})$ be an operator in $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$ for each i. Then the guarded composition of $\mathbf{A}_i$'s along with the basis $\{|i\rangle\}$ is*

$$\square\,(c, |i\rangle \to \mathbf{A}_i) = \sum_{\overline{n} \in \omega^C} \left(\sum_i (|i\rangle_c \langle i| \otimes \mathbf{A}_i(\overline{n})) \right). \tag{7.21}$$

Note that for each $\overline{n} \in \omega^C$, $\sum_i (|i\rangle_c \langle i| \otimes \mathbf{A}_i(n))$ is an operator in

$$\mathcal{H}_c^{\otimes (n_c+1)} \otimes \left(\bigotimes_{d \in C \setminus \{c\}} \mathcal{H}_d^{n_d} \right) \otimes \mathcal{H},$$

and thus $\square\,(c, |i\rangle \to \mathbf{A}_i) \in \mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$.

The following lemma shows that both product and guarded composition of operators in the free Fock space are continuous with respect to the flat order (see Definition 3.3.5).

Lemma 7.4.2. *Let $\{\mathbf{A}_j\}$ and $\{\mathbf{B}_j\}$ be chains in $(\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$, and so are $\{\mathbf{A}_{ij}\}$ for each i. Then*

(i) $\bigsqcup_j (\mathbf{A}_j \cdot \mathbf{B}_j) = \left(\bigsqcup_j \mathbf{A}_j\right) \cdot \left(\bigsqcup_j \mathbf{B}_j\right)$.

(ii) $\bigsqcup_j \square\, (c, |i\rangle \rightarrow \mathbf{A}_{ij}) = \square\left(c, |i\rangle \rightarrow \left(\bigsqcup_j \mathbf{A}_{ij}\right)\right)$.

Proof. We only prove part (ii). The proof of part (i) is similar. For each i, we assume that

$$\bigsqcup_j \mathbf{A}_{ij} = \mathbf{A}_i = \sum_{\overline{n}\in\omega^C} \mathbf{A}_i(\overline{n}).$$

By the construction of least upper bound in $(\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$ given in the proof of Lemma 7.4.1, we can write

$$\mathbf{A}_{ij} = \sum_{\overline{n}\in\Omega_{ij}} \mathbf{A}_i(\overline{n})$$

for some $\Omega_{ij} \subseteq \omega^C$ with $\bigcup_j \Omega_{ij} = \omega^C$ for every i. By appending zero operators to the end of shorter summations, we may further ensure that index sets Ω_{ij}'s for all i are the same, say Ω_j. Then by the defining equation (7.21) we obtain:

$$\begin{aligned}\bigsqcup_j \square\,(c, |i\rangle \rightarrow \mathbf{A}_{ij}) &= \bigsqcup_j \sum_{\overline{n}\in\Omega_j} \left(\sum_i (|i\rangle_c\langle i| \otimes \mathbf{A}_i(\overline{n}))\right) \\ &= \sum_{\overline{n}\in\omega^C} \left(\sum_i (|i\rangle_c\langle i| \otimes \mathbf{A}_i(\overline{n}))\right) = \square\,(c, |i\rangle \rightarrow \mathbf{A}_i)\,.\end{aligned}$$

□

7.4.2 SEMANTIC FUNCTIONALS OF PROGRAM SCHEMES

The semantics of quantum programs (i.e., program schemes without procedure identifiers) was already defined in Definition 7.1.2. With the preparation in the last subsection, now we are able to define the semantics of general quantum program schemes.

Let $P = P[X_1, \ldots, X_m]$ be a program scheme with procedure identifiers $X_1, \ldots, X_m$. We write C for the set of "coins" occurring in P. For each $c \in C$, let $\mathcal{H}_c$ be the state Hilbert space of quantum "coin" c. By the principal system of P we mean the composition of the systems denoted by principal variables appearing in P. Let $\mathcal{H}$ be the state Hilbert space of the principal system. Then the semantics of P can be defined as a functional in the domain $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$ described in the last subsection.

Definition 7.4.4. *The semantic functional of program scheme $P = P[X_1, \ldots, X_m]$ is a mapping*

$$[\![P]\!] : \mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})^m \rightarrow \mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}).$$

For any operators $\mathbf{A}_1, \ldots, \mathbf{A}_m \in \mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$, $[\![P]\!](\mathbf{A}_1, \ldots, \mathcal{A}_m)$ is inductively defined as follows:

(i) *If $P = \mathbf{abort}$, then $[\![P]\!](\mathbf{A}_1, \dots, \mathbf{A}_m)$ is the zero operator*

$$\mathbf{A} = \sum_{\overline{n} \in \omega^C} \mathbf{A}(\overline{n})$$

with $\mathbf{A}(\overline{n}) = 0$ (the zero operator in $\left(\bigotimes_{c \in C} \mathcal{H}_c^{\otimes n_c}\right) \otimes \mathcal{H}$) for all $\overline{n} \in \omega^C$;

(ii) *If $P = \mathbf{skip}$, then $[\![P]\!](\mathbf{A}_1, \dots, \mathbf{A}_m)$ is the identity operator*

$$\mathbf{A} = \sum_{\overline{n} \in \omega^C} \mathbf{A}(\overline{n})$$

with $\mathbf{A}(\overline{n}) = I$ (the identity operator in $\left(\bigotimes_{c \in C} \mathcal{H}_c^{\otimes n_c}\right) \otimes \mathcal{H}$) for all $\overline{n} \in \omega^C$ with $n_c \neq 0$ for every $c \in C$;

(iii) *If $P = U[\overline{c}, \overline{q}]$, then $[\![P]\!](\mathbf{A}_1, \dots, \mathbf{A}_m)$ is the cylindrical extension of U:*

$$\mathbf{A} = \sum_{\overline{n} \in \omega^C} \mathbf{A}(\overline{n})$$

with

$$\mathbf{A}(\overline{n}) = I_1 \otimes I_2(\overline{n}) \otimes U \otimes I_3$$

where:

(a) *I_1 is the identity operator in the state Hilbert space of those "coins" that are not in $\overline{c}$;*

(b) *$I_2(\overline{n})$ is the identity operator in $\bigotimes_{c \in \overline{c}} \mathcal{H}_c^{\otimes(n_c - 1)}$; and*

(c) *I_3 is the identity operator in the state Hilbert space of those principal variables that are not in $\overline{q}$ for all $n \geq 1$;*

(iv) *If $P = X_j$ $(1 \leq j \leq m)$, then $[\![P]\!](\mathbf{A}_1, \dots, \mathbf{A}_m) = \mathbf{A}_j$;*

(v) *If $P = P_1; P_2$, then*

$$[\![P]\!](\mathbf{A}_1, \dots, \mathbf{A}_m) = [\![P_2]\!](\mathbf{A}_1, \dots, \mathbf{A}_m) \cdot [\![P_1]\!](\mathbf{A}_1, \dots, \mathbf{A}_m)$$

(see the defining equation (7.20) of the product of operators in the free Fock space);

(vi) *If $P = \mathbf{qif}\ [c](\Box i \cdot |i\rangle \to P_i)\ \mathbf{fiq}$, then*

$$[\![P]\!](\mathbf{A}_1, \dots, \mathbf{A}_m) = \Box\left(c, |i\rangle \to [\![P_i]\!](\mathbf{A}_1, \dots, \mathbf{A}_m)\right)$$

(see the defining equation (7.21) of guarded composition of operators in the free Fock space).

It is easy to see that whenever $m = 0$, that is, P contains no procedure identifiers, then the previous definition degenerates to Definition 7.1.2.

As we learned from classical programming theory and in Section 3.4, continuity of the functions involved in a recursive equation is usually crucial for the existence of solutions of the equation. So, we now examine continuity of the semantic functionals defined previously. To this end, the Cartesian power $\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})^m$ is naturally equipped with the order $\sqsubseteq$ defined component-wise from the flat order in the CPO $\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})$: for any $\mathbf{A}_1,\ldots,\mathbf{A}_m,\mathbf{B}_1,\ldots,\mathbf{B}_m\in\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})$,

- $(\mathbf{A}_1,\ldots,\mathbf{A}_m)\sqsubseteq(\mathbf{B}_1,\ldots,\mathbf{B}_m)$ if and only if for every $1\le i\le m$, $\mathbf{A}_i\sqsubseteq\mathbf{B}_i$.

The second occurrence of symbol "$\sqsubseteq$" in the preceding statement stands for the flat order in $\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})$. Then $(\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})^m,\sqsubseteq)$ is a CPO too. Furthermore, we have:

Theorem 7.4.1 (Continuity of Semantic Functionals). *For any program scheme $P=P[X_1,\ldots,X_m]$, the semantic functional*

$$[\![P]\!]:(\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})^m,\sqsubseteq)\rightarrow(\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H}),\sqsubseteq)$$

is continuous.

Proof. It can be easily proved by induction on the structure of P using Lemma 7.4.2. □

There is an essential difference between quantum recursive programs defined in this chapter and recursive quantum programs studied in Section 3.4. The semantics of recursive quantum programs can be defined in a way that is very similar to the way in which we deal with recursion in classical programming theory. More explicitly, it can be properly characterized as a fixed point of semantic functionals. However, semantic functionals are not complete for describing the behavior of quantum recursive programs considered here. They must be combined with the notion of *creation functional* defined in the following:

Definition 7.4.5. *For each "coin" $c\in C$, the creation functional*

$$\mathbb{K}_c:\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})\rightarrow\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})$$

is defined as follows: for any $\mathbf{A}=\sum_{\overline{n}\in\omega^C}\mathbf{A}(\overline{n})\in\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})$,

$$\mathbb{K}_c(\mathbf{A})=\sum_{\overline{n}\in\omega^C}(I_c\otimes\mathbf{A}(\overline{n}))$$

where I_c is the identity operator in $\mathcal{H}_c$.

We observe that $\mathbf{A}(\overline{n})$ is an operator in $\left(\bigotimes_{d\in C}\mathcal{H}_d^{\otimes n_d}\right)\otimes\mathcal{H}$, whereas $I_c\otimes\mathbf{A}(\overline{n})$ is an operator in

$$\mathcal{H}_c^{\otimes(n_c+1)}\otimes\left(\bigotimes_{d\in C\setminus\{c\}}\mathcal{H}_d^{\otimes n_d}\right)\otimes\mathcal{H}.$$

In a sense, the creation functional can be seen as a counterpart of creation operator (Definition 7.3.10) in the domain $\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})$. Intuitively, the creation functional $\mathbb{K}_c$ moves all copies of $\mathcal{H}_c$ one position to the right so that the ith copy becomes the

$(i+1)$th copy for all $i = 0, 1, 2, \ldots$. Thus, a new position is created at the left end for a new copy of $\mathcal{H}_c$. For other "coins" d, $\mathbb{K}_c$ does not move any copy of $\mathcal{H}_d$.

It is clear that for any two "coins" c, d, the corresponding creation functionals $\mathbb{K}_c$ and $\mathbb{K}_d$ commute; that is,

$$\mathbb{K}_a \circ \mathbb{K}_d = \mathbb{K}_d \circ \mathbb{K}_c.$$

Note that the set C of "coins" in a program scheme P is finite. Suppose that $C = \{c_1, c_2, \ldots, c_k\}$. Then we can define the creation functional

$$\mathbb{K}_C = \mathbb{K}_{c_1} \circ \mathbb{K}_{c_2} \circ \ldots \circ \mathbb{K}_{c_k}.$$

For the special case where the set C of "coins" is empty, $\mathbb{K}_C$ is the identity functional; that is, $\mathbb{K}_C(\mathbf{A}) = \mathbf{A}$ for all $\mathbf{A}$.

Continuity of the creation functionals with respect to the flat order between operators in the free Fock space is shown in the following:

Lemma 7.4.3 (Continuity of Creation Functionals). *For each $c \in C$, the creation functionals*

$$\mathbb{K}_c \text{ and } \mathbb{K}_C : (\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq) \to (\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$$

are continuous.

Proof. Straightforward by definition. □

Combining continuity of semantic functionals and the creation functionals (Theorem 7.4.1 and Lemma 7.4.3), we obtain:

Corollary 7.4.1. *Let $P = P[X_1, \ldots, X_m]$ be a program scheme and C the set of "coins" occurring in P. We define the functional*

$$\mathbb{K}_C^m \circ [\![P]\!] : (\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})^m, \sqsubseteq) \to (\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$$

by

$$(\mathbb{K}_C^m \circ [\![P]\!])(\mathbf{A}_1, \ldots, \mathbf{A}_m) = [\![P]\!](\mathbb{K}_C(\mathbf{A}_1), \ldots, \mathbb{K}_C(\mathbf{A}_m))$$

for any $\mathbf{A}_1, \ldots, \mathbf{A}_m \in \mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$. Then $\mathbb{K}_C^m \circ [\![P]\!]$ is continuous.

7.4.3 FIXED POINT SEMANTICS

Now we have all the ingredients needed to define the denotational semantics of quantum recursive programs using the standard fixed point technique. Let us consider a recursive program P declared by the system of equations:

$$D:\begin{cases}X_1 \Leftarrow P_1,\\ \quad\ \ \cdots\cdots\\ X_m \Leftarrow P_m,\end{cases}\tag{7.22}$$

where $P_i = P_i[X_1,\dots,X_m]$ is a program scheme containing at most procedure identifiers $X_1,\dots,X_m$ for every $1\le i\le m$. Using the functional $[\![\cdot]\!]$ and $\mathbb{K}_C$ defined in the last subsection, the system D of recursive equations naturally induces a semantic functional:

$$[\![D]\!]:\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})^m\to\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})^m$$

defined as follows:

$$\begin{aligned}[\![D]\!](\mathbf{A}_1,\dots,\mathbf{A}_m) = ((\mathbb{K}_C^m\circ[\![P_1]\!])(\mathbf{A}_1,\dots,\mathbf{A}_m),\dots,\\ (\mathbb{K}_C^m\circ[\![P_m]\!])(\mathbf{A}_1,\dots,\mathbf{A}_m))\end{aligned}\tag{7.23}$$

for all $\mathbf{A}_1,\dots,\mathbf{A}_m\in\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})$, where C is the set of "coins" appearing in D, that is, in one of $P_1,\dots,P_m$. It follows from Corollary 7.4.1 that

$$[\![D]\!]:(\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})^m,\sqsubseteq)\to(\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})^m,\sqsubseteq)$$

is continuous. Then the Knaster-Tarski Fixed Point Theorem (see Theorem 3.3.1) asserts that $[\![D]\!]$ has the least fixed point $\mu[\![D]\!]$, which is exactly what we need to define the semantics of P.

Definition 7.4.6. *The fixed point (denotational) semantics of the quantum recursive program P declared by D is*

$$[\![P]\!]_{fix}=[\![P]\!](\mu[\![D]\!]);$$

that is, if $\mu[\![D]\!]=(\mathbf{A}_1^*,\dots,\mathbf{A}_m^*)\in\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})^m$, then

$$[\![P]\!]_{fix}=[\![P]\!](\mathbf{A}_1^*,\dots,\mathbf{A}_m^*)$$

(see Definition 7.4.4).

7.4.4 SYNTACTIC APPROXIMATION

The fixed point semantics of quantum recursive programs was discussed in the last subsection. We now turn to consider the syntactic approximation technique for defining the semantics of quantum recursive programs. The semantics defined in this subsection will be proved to be equivalent to the fixed point semantics.

As discussed at the end of Section 7.2, a problem that was not present in the classical programming theory is that we have to carefully avoid the conflict of quantum "coin" variables when defining the notion of substitution. To overcome it, we assume that each "coin" variable $c\in C$ has infinitely many copies $c_0,c_1,c_2,\dots$ with $c_0=c$. The variables $c_1,c_2,\dots$ are used to represent a sequence of particles that are all identical to the particle $c_0=c$. Then the notion of a quantum program scheme defined in Section 7.1 will be used in a slightly broader way: a quantum program scheme may contain not only a "coin" c but also some of its copies $c_1, c_2, \ldots$. Such a quantum program scheme is called a generalized quantum program scheme. If such a generalized quantum program scheme contains no procedure identifiers, then it is called a generalized quantum program. With these assumptions, we can introduce the notion of substitution.

Definition 7.4.7. *Let $P = P[X_1, \ldots, X_m]$ be a generalized quantum program scheme that contains at most procedure identifiers $X_1, \ldots, X_m$, and let $Q_1, \ldots, Q_m$ be generalized quantum programs (without any procedure identifier). Then the simultaneous substitution*

$$P[Q_1/X_1, \ldots, Q_m/X_m]$$

of $X_1, \ldots, X_m$ by $Q_1, \ldots, Q_m$ in P is inductively defined as follows:

(i) *If $P =$ **abort**, **skip** or an unitary transformation, then*

$$P[Q_1/X_1, \ldots, Q_m/X_m] = P;$$

(ii) *If $P = X_i$ $(1 \leq i \leq m)$, then*

$$P[Q_1/X_1, \ldots, Q_m/X_m] = Q_i;$$

(iii) *If $P = P_1; P_2$, then*

$$\begin{aligned}P[Q_1/X_1, \ldots, Q_m/X_m] = P_1[Q_1/X_1, \ldots, Q_m/X_m];\\ P_2[Q_1/X_1, \ldots, Q_m/X_m];\end{aligned}$$

(iv) *If $P =$ **qif** $[c](\square i \cdot |i\rangle \rightarrow P_i)$ **fiq**, then*

$$P[Q_1/X_1, \ldots, Q_m/X_m] = \textbf{qif}\ [c](\square i \cdot |i\rangle \rightarrow P_i')\ \textbf{fiq}$$

where for every i, P_i' is obtained through replacing the jth copy c_j of c in $P_i[Q_1/X_1, \ldots, Q_m/X_m]$ by the $(j+1)$th copy c_{j+1} of c for all j.

Note that in Clause (iv) of this definition, since P is a generalized quantum program scheme, the "coin" c may not be an original "coin" but some copy d_k of an original "coin" $d \in C$. In this case, the jth copy of c is actually the $(k+j)$th copy of d: $c_j = (d_k)_j = d_{k+j}$ for $j \geq -d$.

The semantics of a generalized quantum program P can be given using Definition 7.1.2 in such a way that a "coin" c and its copies $c_1, c_2, \ldots$ are treated as distinct variables to each other. For each "coin" c, let n_c be the greatest index n such that the copy c_n appears in P. Then the semantics $[\![P]\!]$ of P is an operator in $\left(\bigotimes_{c \in C} \mathcal{H}_c^{\otimes n_c}\right) \otimes \mathcal{H}$. Furthermore, it can be identified with its cylindrical extension in $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$:

$$\sum_{\overline{m} \in \omega^C} \left(I(\overline{m}) \otimes [\![P]\!]\right),$$

where for each $\overline{m} \in \omega^C$, $I(\overline{m})$ is the identity operator in $\bigotimes_{c\in C} \mathcal{H}_c^{\otimes m_c}$. Based on this observation, the semantics of substitution defined previously is characterized by the following:

Lemma 7.4.4. *For any (generalized) quantum program scheme $P = P[X_1, \dots, X_m]$ and (generalized) quantum programs $Q_1, \dots, Q_m$, we have:*

$$\begin{aligned}\llbracket P[Q_1/X_1, \dots, Q_m/X_m]\rrbracket &= (\mathbb{K}_C^m \circ \llbracket P\rrbracket)(\llbracket Q_1\rrbracket, \dots, \llbracket Q_m\rrbracket)\\ &= \llbracket P\rrbracket(\mathbb{K}_C(\llbracket Q_1\rrbracket), \dots, \mathbb{K}_C(\llbracket Q_m\rrbracket)),\end{aligned}$$

where $\mathbb{K}_C$ is the creation functional with C being the set of "coins" in P.

Proof. We prove the lemma by induction on the structure of P.

- Case 1. $P =$ **abort**, **skip** or an unitary transformation. Obvious.
- Case 2. $P = X_j$ $(1 \le j \le m)$. Then

$$P[Q_1/X_1, \dots, Q_m/X_m] = Q_m.$$

On the other hand, since the set of "coins" in P is empty,

$$\mathbb{K}_C(\llbracket Q_i\rrbracket) = \llbracket Q_i\rrbracket$$

for all $1 \le i \le m$. Thus, by clause (iv) of Definition 7.4.4 we obtain:

$$\begin{aligned}\llbracket P[Q_1/X_1, \dots, Q_m/X_m]\rrbracket &= \llbracket Q_m\rrbracket\\ &= \llbracket P\rrbracket(\llbracket Q_1\rrbracket, \dots, \llbracket Q_m\rrbracket)\\ &= \llbracket P\rrbracket(\mathbb{K}_C(\llbracket Q_1\rrbracket), \dots, \mathbb{K}_C(\llbracket Q_m\rrbracket)).\end{aligned}$$

- Case 3. $P = P_1; P_2$. Then by clause (iii) of Definition 7.1.2, clause (v) of Definition 7.4.4 and the induction hypothesis, we have:

$$\begin{aligned}\llbracket P[Q_1/X_1, \dots, Q_m/X_m]\rrbracket &= \llbracket P_1[Q_1/X_1, \dots, Q_m/X_m]; P_2[Q_1/X_1, \dots, Q_m/X_m]\rrbracket\\ &= \llbracket P_2[Q_1/X_1, \dots, Q_m/X_m]\rrbracket \cdot \llbracket P_1[Q_1/X_1, \dots, Q_m/X_m]\rrbracket\\ &= \llbracket P_2\rrbracket(\mathbb{K}_C(\llbracket Q_1\rrbracket), \dots, \mathbb{K}_C(\llbracket Q_m\rrbracket)) \cdot \llbracket P_1\rrbracket(\mathbb{K}_C(\llbracket Q_1\rrbracket), \dots, \mathbb{K}_C(\llbracket Q_m\rrbracket))\\ &= \llbracket P_1; P_2\rrbracket(\mathbb{K}_C(\llbracket Q_1\rrbracket), \dots, \mathbb{K}_C(\llbracket Q_m\rrbracket))\\ &= \llbracket P\rrbracket(\mathbb{K}_C(\llbracket Q_1\rrbracket), \dots, \mathbb{K}_C(\llbracket Q_m\rrbracket)).\end{aligned}$$

- Case 4. $P =$ **qif** $[c](\square i \cdot |i\rangle \to P_i)$ **fiq**. Then

$$P[Q_1/X_1, \dots, Q_m/X_m] = \textbf{qif } [c](\square i \cdot |i\rangle \to P_i') \textbf{ fiq},$$

where P_i' is obtained according to clause (iv) of Definition 7.4.7. For each i, by the induction hypothesis we obtain:

$$\llbracket P_i[Q_1/X_1, \dots, Q_m/X_m]\rrbracket = \llbracket P_i\rrbracket(\mathbb{K}_{C\setminus\{c\}}(\llbracket Q_1\rrbracket), \dots, \mathbb{K}_{C\setminus\{c\}}(\llbracket Q_m\rrbracket))$$

because the "coin" c does not appear in P_i'. Furthermore, it follows that

$$\begin{aligned}
[\![P_i']\!] &= \mathbb{K}_c([\![P_i[Q_1/X_1,\dots,Q_m/X_m]]\!]) \\
&= \mathbb{K}_c([\![P_i]\!](\mathbb{K}_{C\setminus\{c\}}([\![Q_1]\!]),\dots,\mathbb{K}_{C\setminus\{c\}}([\![Q_m]\!]))) \\
&= [\![P_i]\!]((\mathbb{K}_c\circ\mathbb{K}_{C\setminus\{c\}})([\![Q_1]\!]),\dots,(\mathbb{K}_c\circ\mathbb{K}_{C\setminus\{c\}})([\![Q_m]\!])) \\
&= [\![P_i]\!](\mathbb{K}_C([\![Q_1]\!]),\dots,\mathbb{K}_C([\![Q_m]\!])).
\end{aligned}$$

Therefore, by clause (iv) of Definition 7.1.2, clause (vi) of Definition 7.4.4 and equation (7.21), we have:

$$\begin{aligned}
[\![P[Q_1/X_1,\dots,Q_m/X_m]]\!] &= \sum_i \left(|i\rangle\langle i| \otimes [\![P_i']\!]\right) \\
&= \square(c,|i\rangle \to [\![P_i]\!](\mathbb{K}_C([\![Q_1]\!]),\dots,\mathbb{K}_C([\![Q_m]\!])) \\
&= [\![P]\!](\mathbb{K}_C([\![Q_1]\!]),\dots,\mathbb{K}_C([\![Q_m]\!])).
\end{aligned}$$

□

Essentially, the previous lemma shows that the semantic functional of a generalized quantum program scheme is compositional modulo the creation functional.

Now the notion of syntactic approximation can be properly defined based on Definition 7.4.7.

Definition 7.4.8

(i) *Let $X_1,\dots,X_m$ be procedure identifiers declared by the system D of recursive equations (7.22). Then for each $1 \le k \le m$, the nth syntactic approximation $X_k^{(n)}$ of X_k is inductively defined as follows:*

$$\begin{cases} X_k^{(0)} = \textbf{abort}, \\ X_k^{(n+1)} = P_k[X_1^{(n)}/X_1,\dots,X_m^{(n)}/X_m] \ \ \textit{for } n \ge 0. \end{cases}$$

(ii) *Let $P = P[X_1,\dots,X_m]$ be a quantum recursive program declared by the system D of equations (7.22). Then for each $n \ge 0$, its nth syntactic approximation $P^{(n)}$ is inductively defined as follows:*

$$\begin{cases} P^{(0)} = \textbf{abort}, \\ P^{(n+1)} = P[X_1^{(n)}/X_1,\dots,X_m^{(n)}/X_m] \ \ \textit{for } n \ge 0. \end{cases}$$

Syntactic approximation actually gives an operational semantics of quantum recursive programs. As in the theory of classical programming, substitution represents an application of the so-called *copy rule*:

- At runtime, a procedure call is treated like the procedure body inserted at the place of call.

Of course, simplifications may happen within $X_k^{(n)}$ by operations of linear operators; for example,

$$CNOT[q_1,q_2]; X[q_2]; CNOT[q_1,q_2]$$

can be replaced by $X[q_2]$, where q_1, q_2 are principal system variables, *CNOT* is the controlled-NOT gate and X is the NOT gate. To simplify the presentation, we choose not to explicitly describe these simplifications.

The major difference between the classical case and the quantum case is that in the latter we need to continuously introduce new "coin" variables to avoid variable conflict when we unfold a quantum recursive program using its syntactic approximations: for each $n \geq 0$, a new copy of each "coin" in P_k is created in the substitution

$$X_k^{(n+1)} = P_k[X_1^{(n)}/X_1, \ldots, X_m^{(n)}/X_m]$$

(see clause (iv) of Definition 7.4.7). Thus, a quantum recursive program should be understood as a quantum system with variable particle number and described in the second quantization formalism.

Note that for all $1 \leq k \leq m$ and $n \geq 0$, the syntactic approximation $X_k^{(n)}$ is a generalized quantum program containing no procedure identifiers. Thus, its semantics $[\![X_k^{(n)}]\!]$ can be given by a slightly extended version of Definition 7.1.2: a "coin" c and its copies $c_1, c_2, \ldots$ are allowed to appear in the same (generalized) program and they are considered as distinct variables. As before, the principal system is the composite system of the subsystems denoted by principal variables appearing in $P_1, \ldots, P_m$ and its state Hilbert space is denoted by $\mathcal{H}$. Assume that C is the set of "coin" variables appearing in $P_1, \ldots, P_m$. For each $c \in C$, we write $\mathcal{H}_c$ for the state Hilbert space of quantum "coin" c. Then it is easy to see that $[\![X_k^{(n)}]\!]$ is an operator in

$$\bigoplus_{j=0}^{n} \left(\mathcal{H}_C^{\otimes n_j} \otimes \mathcal{H}\right)$$

where $\mathcal{H}_C = \bigotimes_{c \in C} \mathcal{H}_c$. So, we can imagine that $[\![X_k^{(n)}]\!] \in \mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$. Furthermore, we have:

Lemma 7.4.5. *For each $1 \leq k \leq m$, $\left\{[\![X_k^{(n)}]\!]\right\}_{n=0}^{\infty}$ is an increasing chain with respect to the flat order, and thus*

$$[\![X_k^{(\infty)}]\!] = \lim_{n\to\infty} [\![X_k^{(n)}]\!] \triangleq \bigsqcup_{n=0}^{\infty} [\![X_k^{(n)}]\!] \tag{7.24}$$

exists in $(\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$.

Proof. We show that

$$[\![X_k^{(n)}]\!] \sqsubseteq [\![X_k^{(n+1)}]\!]$$

by induction on n. The case of $n = 0$ is trivial because

$$[\![X_k^{(0)}]\!] = [\![\mathbf{abort}]\!] = 0.$$

In general, by the induction hypothesis on $n-1$ and Corollary 7.4.1, we have:

$$\begin{aligned}[\![X_k^{(n)}]\!] &= [\![P_k]\!](\mathbb{K}_C([\![X_1^{(n-1)}]\!]),\ldots,\mathbb{K}_C([\![X_m^{(n-1)}]\!])) \\ &\sqsubseteq [\![P_k]\!](\mathbb{K}_C([\![X_1^{(n)}]\!]),\ldots,\mathbb{K}_C([\![X_m^{(n)}]\!])) \\ &= [\![X_k^{(n+1)}]\!],\end{aligned}$$

where C is the set of "coins" in D. Then existence of the least upper bound (7.24) follows immediately from Lemma 7.4.1. □

We now are ready to define the operational semantics of quantum recursive programs.

Definition 7.4.9. *Let P be a quantum recursive program declared by the system D of equations (7.22). Then its operational semantics is*

$$[\![P]\!]_{op} = [\![P]\!]\left([\![X_1^{(\infty)}]\!],\ldots,[\![X_m^{(\infty)}]\!]\right).$$

The operator $[\![P]\!]_{op}$ is called the operational semantics for the reason that it is defined based on the copy rule. But it is actually not an operational semantics in the strict sense because the notion of limit is involved in $[\![X_i^{(\infty)}]\!]$ $(1 \le i \le m)$.

The operational semantics of quantum recursive program P can be characterized by the limit of its syntactic approximations (with respect to its declaration D).

Proposition 7.4.1. *It holds in the domain $(\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H}),\sqsubseteq)$ that*

$$[\![P]\!]_{op} = \bigsqcup_{n=0}^{\infty}[\![P^{(n)}]\!].$$

Proof. It follows from Lemma 7.4.4 that

$$\begin{aligned}\bigsqcup_{n=0}^{\infty}[\![P^{(n)}]\!] &= \bigsqcup_{n=0}^{\infty}[\![P[X_1^{(n)}/X_1,\ldots,X_m^{(n)}/X_m]]\!] \\ &= \bigsqcup_{n=0}^{\infty}[\![P]\!]\left(\mathbb{K}_C([\![X_1^{(n)}]\!]),\ldots,\mathbb{K}_C([\![X_m^{(n)}]\!])\right)\end{aligned}$$

where $\mathbb{K}_C$ is the creation functional with respect to the "coins" C in P. However, all the "coin" C in P do not appear in $X_1^{(n)},\ldots,X_m^{(n)}$ (see the condition in Definition 7.1.3). So,

$$\mathbb{K}_C\left([\![X_k^{(n)}]\!]\right) = [\![X_k^{(n)}]\!]$$

for every $1 \leq k \leq m$, and by Theorem 7.4.1 we obtain:

$$\begin{aligned}\bigsqcup_{n=0}^{\infty}[\![P^{(n)}]\!] &= \bigsqcup_{n=0}^{\infty}[\![P]\!]\left([\![X_1^{(n)}]\!],\ldots,[\![X_m^{(n)}]\!]\right)\\ &= [\![P]\!]\left(\bigsqcup_{n=0}^{\infty}[\![X_1^{(n)}]\!],\ldots,\bigsqcup_{n=0}^{\infty}[\![X_m^{(n)}]\!]\right)\\ &= [\![P]\!]\left([\![X_1^{\infty}]\!],\ldots,[\![X_m^{\infty}]\!]\right)\\ &= [\![P]\!]_{op}.\end{aligned}$$

□

Intuitively, for each $n \geq 0$, $[\![P^{(n)}]\!]$ denotes the partial computational result of recursive program P up to the nth step. Thus, the preceding proposition shows that the complete computational result can be approximated by partial computational results.

Finally, the equivalence between denotational and operational semantics of recursive programs is established in the following:

Theorem 7.4.2 (Equivalence of Denotational Semantics and Operational Semantics). *For any quantum recursive program P, we have:*

$$[\![P]\!]_{fix} = [\![P]\!]_{op}.$$

Proof. By Definitions 7.4.6 and 7.4.9, it suffices to show that $\left([\![X_1^{(\infty)}]\!],\ldots,[\![X_m^{(\infty)}]\!]\right)$ is the least fixed point of semantic functional $[\![D]\!]$, where D is the declaration of procedure identifiers in P. With Theorem 7.4.1 and Lemmas 7.4.3 and 7.4.4, we obtain:

$$\begin{aligned}[\![X_k^{(\infty)}]\!] &= \bigsqcup_{n=0}^{\infty}[\![X_k^{(n)}]\!]\\ &= \bigsqcup_{n=0}^{\infty}[\![P_k[X_1^{(n)}/X_1,\ldots,X_m^{(n)}/X_m]]\!]\\ &= \bigsqcup_{n=0}^{\infty}[\![P_k]\!]\left(\mathbb{K}_C([\![X_1^{(n)}]\!]),\ldots,\mathbb{K}_C([\![X_m^{(n)}]\!])\right)\\ &= [\![P_k]\!]\left(\mathbb{K}_C\left(\bigsqcup_{n=0}^{\infty}[\![X_1^{(n)}]\!]\right),\ldots,\mathbb{K}_C\left(\bigsqcup_{n=0}^{\infty}[\![X_m^{(n)}]\!]\right)\right)\\ &= [\![P_k]\!](\mathbb{K}_C([\![X_1^{(\infty)}]\!]),\ldots,\mathbb{K}_C([\![X_m^{(\infty)}]\!]))\end{aligned}$$

for every $1 \leq k \leq m$, where C is the set of “coins” in D. So, $\left([\![X_1^{(\infty)}]\!],\ldots,[\![X_m^{(\infty)}]\!]\right)$ is a fixed point of $[\![D]\!]$. On the other hand, if $(\mathbf{A}_1,\ldots,\mathbf{A}_m) \in \mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})^m$ is a fixed point of $[\![D]\!]$, then we can prove that for every $n \geq 0$,

$$\left(\llbracket X_1^{(n)}\rrbracket,\ldots,\llbracket X_m^{(n)}\rrbracket\right)\sqsubseteq(\mathbf{A}_1,\ldots,\mathbf{A}_m)$$

by induction on n. Indeed, the case of $n = 0$ is obvious. In general, using the induction hypothesis on $n-1$, Corollary 7.4.1 and Lemma 7.4.4 we obtain:

$$\begin{aligned}
(\mathbf{A}_1,\ldots,\mathbf{A}_m) &= \llbracket D\rrbracket(\mathbf{A}_1,\ldots,\mathbf{A}_m)\\
&= \left((\mathbb{K}_C^m\circ\llbracket P_1\rrbracket)(\mathbf{A}_1,\ldots,\mathbf{A}_m),\ldots,(\mathbb{K}_C^m\circ\llbracket P_m\rrbracket)(\mathbf{A}_1,\ldots,\mathbf{A}_m)\right)\\
&\sqsupseteq \Big((\mathbb{K}_C^m\circ\llbracket P_1\rrbracket)\left(\llbracket X_1^{(n-1)}\rrbracket,\ldots,\llbracket X_m^{(n-1)}\rrbracket\right),\ldots,\\
&\qquad\qquad (\mathbb{K}_C^m\circ\llbracket P_m\rrbracket)\left(\llbracket X_1^{(n-1)}\rrbracket,\ldots,\llbracket X_m^{(n-1)}\rrbracket\right)\Big)\\
&= \left(\llbracket X_1^{(n)}\rrbracket,\ldots,\llbracket X_m^{(n)}\rrbracket\right).
\end{aligned}$$

Therefore, it holds that

$$\left(\llbracket X_1^{(\infty)}\rrbracket,\ldots,\llbracket X_m^{(\infty)}\rrbracket\right)=\bigsqcup_{n=0}^{\infty}\left(\llbracket X_1^{(n)}\rrbracket,\ldots,\llbracket X_m^{(n)}\rrbracket\right)\sqsubseteq(\mathbf{A}_1,\ldots,\mathbf{A}_m),$$

and $\left(\llbracket X_1^{(\infty)}\rrbracket,\ldots,\llbracket X_m^{(\infty)}\rrbracket\right)$ is the least fixed point of $\llbracket D\rrbracket$. □

In light of this theorem, we will simply write $\llbracket P\rrbracket$ for both the denotational (fixed point) and operational semantics of a recursive program P. But we should carefully distinguish the semantics $\llbracket P\rrbracket\in\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})$ of a recursive program $P = P[X_1,\ldots,X_m]$ declared by a system of equations about $X_1,\ldots,X_m$ from the semantic functional

$$\llbracket P\rrbracket:\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})^m\to\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})$$

of program scheme $P = P[X_1,\ldots,X_m]$. Usually, such a difference can be recognized from the context.

7.5 RECOVERING SYMMETRY AND ANTISYMMETRY

The last section developed the techniques for solving quantum recursive equations in the free Fock space. However, the solutions found in the free Fock space are still not what we really need because they may not preserve symmetry or antisymmetry and thus cannot directly apply to the symmetric Fock space for bosons or the antisymmetric Fock space for fermions. In this section, we introduce the technique of symmetrization that allows us to transform every solution in the free Fock space to a solution in the bosonic or fermionic Fock spaces.

7.5.1 SYMMETRIZATION FUNCTIONAL

Let us first isolate a special subdomain of $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$, namely the domain of symmetric operators. As in Subsection 7.4.1, let $\mathcal{H}$ be the state Hilbert space of the principal system and C the set of "coins," and

$$\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H} = \left(\bigotimes_{c\in C} \mathcal{F}(\mathcal{H}_c)\right) \otimes \mathcal{H} = \bigoplus_{\overline{n}\in\omega^C} \left[\left(\bigotimes_{c\in C} \mathcal{H}_c^{\otimes n_c}\right) \otimes \mathcal{H}\right],$$

where ω is the set of nonnegative integers, and for each $c \in C$, $\mathcal{F}(\mathcal{H}_c)$ is the free Fock space over the state Hilbert space $\mathcal{H}_c$ of "coin" c. As a simple generalization of Definition 7.3.6, we have:

Definition 7.5.1. *For any operator* $\mathbf{A} = \sum_{\overline{n}\in\omega^C} \mathbf{A}(\overline{n}) \in \mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$, *we say that* $\mathbf{A}$ *is symmetric if for each* $\overline{n} \in \omega^c$, *for each* $c \in C$ *and for each permutation* π *of* $0, 1, \ldots, n_c - 1$, P_π *and* $\mathbf{A}(\overline{n})$ *commute; that is,*

$$P_\pi \mathbf{A}(\overline{n}) = \mathbf{A}(\overline{n}) P_\pi.$$

Note that in this definition P_π actually stands for its cylindrical extension

$$P_\pi \otimes \left(\bigotimes_{d\in C\setminus\{c\}} I_d\right) \otimes I$$

in $\left(\bigotimes_{d\in C} \mathcal{H}_d^{\otimes n_d}\right) \otimes \mathcal{H}$, where I_d is the identity operator in $\mathcal{H}_d^{\otimes n_d}$ for every $d \in C\setminus\{c\}$, and I is the identity operator in $\mathcal{H}$.

We write $\mathcal{SO}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$ for the set of all symmetric operators $\mathbf{A} \in \mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$. Its lattice-theoretic structure is presented in the following:

Lemma 7.5.1. $(\mathcal{SO}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$ *is a complete sub-partial order of CPO* $(\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$.

Proof. It suffices to observe that symmetry of operators is preserved by the least upper bound in $(\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}), \sqsubseteq)$; that is, if $\mathbf{A}_i$ is symmetric, so is $\bigsqcup_i \mathbf{A}_i$, as constructed in the proof of Lemma 7.4.1. □

Now we can generalize the symmetrization functional defined by equations (7.14) and (7.15) into the space $\mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$.

Definition 7.5.2

(i) *For each* $\overline{n} \in \omega^C$, *the symmetrization functional* $\mathbb{S}$ *over operators in the space* $\left(\bigotimes_{c\in C} \mathcal{H}_c^{\otimes n_c}\right) \otimes \mathcal{H}$ *is defined by*

$$\mathbb{S}(\mathbf{A}) = \left(\prod_{c\in C} \frac{1}{n_c!}\right) \cdot \sum_{\{\pi_c\}} \left[\left(\bigotimes_{c\in C} P_{\pi_c}\right) \mathbf{A} \left(\bigotimes_{c\in C} P_{\pi_c}^{-1}\right)\right]$$

for every operator $\mathbf{A}$ *in* $\left(\bigotimes_{c\in C}\mathcal{H}_c^{\otimes n_c}\right)\otimes\mathcal{H}$, *where* $\{\pi_c\}$ *traverses over all* C*-indexed families with* π_c *being a permutation of* $0, 1, \ldots, n_c - 1$ *for every* $c \in C$.

(ii) *The symmetrization functional can be extended to* $\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})$ *in a natural way:*

$$\mathbb{S}(\mathbf{A}) = \sum_{\overline{n}\in\omega^C}\mathbb{S}(\mathbf{A}(\overline{n}))$$

for any $\mathbf{A} = \sum_{\overline{n}\in\omega^C}\mathbf{A}(\overline{n}) \in \mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})$.

Obviously, $\mathbb{S}(\mathbf{A}) \in \mathcal{SO}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H})$. Clause (i) of the preceding definition is essentially the same as equation (7.15) but applied to a more complicated space $\left(\bigotimes_{c\in C}\mathcal{H}_c^{\otimes n_c}\right)\otimes\mathcal{H}$. Clause (ii) is then a component-wise generalization of clause (i). Furthermore, the following lemma establishes continuity of the symmetrization functional with respect to the flat order.

Lemma 7.5.2. *The symmetrization functional*

$$\mathbb{S} : (\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H}), \sqsubseteq) \to (\mathcal{SO}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H}), \sqsubseteq)$$

is continuous.

Proof. What we need to prove is that

$$\mathbb{S}\left(\bigsqcup_i \mathbf{A}_i\right) = \bigsqcup_i \mathbb{S}(\mathbf{A}_i)$$

for any chain $\{\mathbf{A}_i\}$ in $(\mathcal{O}(\mathcal{G}(\mathcal{H}_C)\otimes\mathcal{H}), \sqsubseteq)$. Assume that $\mathbf{A} = \bigsqcup_i \mathbf{A}_i$. Then by the proof of Lemma 7.4.1, we can write

$$\mathbf{A} = \sum_{\overline{n}\in\omega}\mathbf{A}(\overline{n}) \ \text{ and } \mathbf{A}_i = \sum_{\overline{n}\in\Omega_i}\mathbf{A}(\overline{n})$$

for some Ω_i with $\bigcup_i \Omega_i = \omega^C$. So, it holds that

$$\bigsqcup_i \mathbb{S}(\mathbf{A}_i) = \bigsqcup_i \sum_{\overline{n}\in\Omega_i}\mathbb{S}(\mathbf{A}(\overline{n})) = \sum_{\overline{n}\in\omega^C}\mathbb{S}(\mathbf{A}(\overline{n})) = \mathbb{S}(\mathbf{A}).$$

□

7.5.2 SYMMETRIZATION OF THE SEMANTICS OF QUANTUM RECURSIVE PROGRAMS

With the preparation in the previous subsection, now we can directly apply the symmetrization functional to the solutions of quantum recursive equations in the free Fock space to give the semantics of quantum recursive programs in the symmetric or antisymmetric Fock space.

Definition 7.5.3. *Let $P = P[X_1, \ldots, X_m]$ be a quantum recursive program declared by the system D of equations (7.22). Then its symmetric semantics $[\![P]\!]_{sym}$ is the symmetrization of its semantics $[\![P]\!]$ in the free Fock space:*

$$[\![P]\!]_{sym} = \mathbb{S}([\![P]\!])$$

where $\mathbb{S}$ is the symmetrization functional,

$$[\![P]\!] = [\![P]\!]_{fix} = [\![P]\!]_{op} \in \mathcal{O}(\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H})$$

(see Theorem 7.4.2), C is the set of "coins" in D, and $\mathcal{H}$ is the state Hilbert space of the principal system of D.

Intuitively, for each "coin" $c \in C$, we use $v_c = +$ or $-$ to indicate that c is a boson or a fermion, respectively. Moreover, we write v for the sequence $\{v_c\}_{c \in C}$. Then

$$\mathcal{G}_v(\mathcal{H}_C) \triangleq \bigotimes_{c \in C} \mathcal{F}_{v_c}(\mathcal{H}_c) \subsetneq \mathcal{G}(\mathcal{H}_C).$$

According to the principle of symmetrization, a physically meaningful input to program P should be a state $|\Psi\rangle$ in $\mathcal{G}_v(\mathcal{H}_C) \otimes \mathcal{H}$. However, the output $[\![P]\!](|\Psi\rangle)$ is not necessarily in $\mathcal{G}_v(\mathcal{H}_C) \otimes \mathcal{H}$ and thus might not be meaningful. Nevertheless, it holds that

$$[\![P]\!]_{sym}(|\Psi\rangle) = \mathbb{S}([\![P]\!])(|\Psi\rangle) \in \mathcal{G}_v(\mathcal{H}_C) \otimes \mathcal{H}.$$

As a symmetrization of Proposition 7.4.1, we have a characterisation of symmetric semantics in terms of syntactic approximations:

Proposition 7.5.1. $[\![P]\!]_{sym} = \bigsqcup_{n=0}^{\infty} \mathbb{S}([\![P^{(n)}]\!])$.

Proof. It follows from Proposition 7.4.1 and Lemma 7.5.2 (continuity of the symmetrization functional) that

$$[\![P]\!]_{sym} = \mathbb{S}([\![P]\!]) = \mathbb{S}\left(\bigsqcup_{n=0}^{\infty} [\![P^{(n)}]\!]\right) = \bigsqcup_{n=0}^{\infty} \mathbb{S}([\![P^{(n)}]\!]).$$

□

7.6 PRINCIPAL SYSTEM SEMANTICS OF QUANTUM RECURSION

In the last section, the symmetric semantics of quantum recursive programs was defined by symmetrizing their semantics in the free Fock space. Let P be a quantum recursive program with $\mathcal{H}$ being the state Hilbert space of its principal variables and C being the set of its "coin." Then semantics $[\![P]\!]$ is an operator in space $\mathcal{G}(\mathcal{H}_C) \otimes \mathcal{H}$, where $\mathcal{G}(\mathcal{H}_C) = \bigotimes_{c \in C} \mathcal{F}(\mathcal{H}_c)$, and for each $c \in C$, $\mathcal{H}_c$ is the state Hilbert space of "coin" c, $\mathcal{F}(\mathcal{H}_c)$ is the free Fock space over $\mathcal{H}_c$. Furthermore, we put

$$\mathcal{G}_v(\mathcal{H}_C) = \bigotimes_{c\in C} \mathcal{F}_{v_c}(\mathcal{H}_c)$$

where v is the sequence $\{v_c\}_{c\in C}$, and for each $c \in C$, $v_c = +$ or $-$ if "coin" c is implemented by a boson or a fermion, respectively. Then symmetric semantics $[\![P]\!]_{sym}$ is an operator in $\mathcal{G}_v(\mathcal{H}_C) \otimes \mathcal{H}$. As we know from the discussions in the previous sections, quantum "coins" in C (and their copies) are only introduced to help the execution of program P but do not actually participate in the computation. What really concerns us is the computation done by the principal system. More precisely, we consider the computation of P with input $|\psi\rangle \in \mathcal{H}$ of principal variables. Assume that the "coins" are initialized in state $|\Psi\rangle \in \mathcal{G}_v(\mathcal{H}_C)$. Then the computation of the program P starts in state $|\Psi\rangle \otimes |\psi\rangle$. At the end, the computational result of P will be stored in the Hilbert space $\mathcal{H}$ of the principal system. This observation leads to the following:

Definition 7.6.1. *Given a state $|\Psi\rangle \in \mathcal{G}_v(\mathcal{H}_C)$. The principal system semantics of program P with respect to "coin" initialization $|\Psi\rangle$ is the mapping $[\![P,\Psi]\!]$ from pure states in $\mathcal{H}$ to partial density operators, i.e., positive operators with trace ≤ 1 (see Section 3.2), in $\mathcal{H}$:*

$$[\![P,\Psi]\!](|\psi\rangle) = tr_{\mathcal{G}_v(\mathcal{H}_C)}(|\Phi\rangle\langle\Phi|)$$

for each pure state $|\psi\rangle$ in $\mathcal{H}$, where

$$|\Phi\rangle = [\![P]\!]_{sym}(|\Psi\rangle \otimes |\psi\rangle),$$

$[\![P]\!]_{sym}$ is the symmetric semantics of P, and $tr_{\mathcal{G}_v(\mathcal{H}_C)}$ is the partial trace over $\mathcal{G}_v(\mathcal{H}_C)$ (see Definition 2.1.22).

As a corollary of Proposition 7.5.1, we have the following characterization of the principal system semantics in terms of syntactic approximation:

Proposition 7.6.1. *For any quantum recursive program P, any initial "coin" state $|\Psi\rangle$ and any principal system state $|\psi\rangle$,*

$$[\![P,\Psi]\!](|\psi\rangle) = \bigsqcup_{n=0}^{\infty} tr_{\bigotimes_{c\in C}\mathcal{H}_{v_c}^{\otimes n}}(|\Phi_n\rangle\langle\Phi_n|)$$

where C is the set of "coins" in P,

$$|\Phi_n\rangle = \mathbb{S}([\![P^{(n)}]\!](|\Psi\rangle \otimes |\psi\rangle))$$

and $P^{(n)}$ is the nth syntactic approximation of P for every $n \geq 0$.

Exercise 7.6.1. *Prove the above proposition.*

7.7 ILLUSTRATIVE EXAMPLES: REVISIT RECURSIVE QUANTUM WALKS

A general theory of quantum recursive programs has been developed in the previous sections. To illustrate the ideas proposed there, let us reconsider two simple recursive quantum walks defined in Section 7.2.

Example 7.7.1 (Unidirectionally recursive Hadamard walk). *Recall from Example 7.2.1 that the unidirectionally recursive Hadamard walk is defined as quantum recursive program X declared by*

$$X \Leftarrow T_L[p] \oplus_{H[d]} (T_R[p]; X).$$

(i) *For each $n \geq 0$, the semantics of the nth approximation of the walk is*

$$[\![X^{(n)}]\!] = \sum_{i=0}^{n-1} \left[\left(\bigotimes_{j=0}^{i-1} |R\rangle_{d_j}\langle R| \otimes |L\rangle_{d_i}\langle L| \right) \mathbf{H}(i) \otimes T_L T_R^i \right] \tag{7.25}$$

where $d_0 = d$, $\mathbf{H}(i)$ is the operator in $\mathcal{H}_d^{\otimes i}$ defined from the Hadamard operator H by equation (7.16). This can be easily shown by induction on n, starting from the first three approximations displayed in equation (7.10). Therefore, the semantics of the unidirectionally recursive Hadamard walk in the free Fock space $\mathcal{F}(\mathcal{H}_d) \otimes \mathcal{H}_p$ is the operator:

$$\begin{aligned} [\![X]\!] &= \lim_{n\to\infty} [\![X^{(n)}]\!] \\ &= \sum_{i=0}^{\infty} \left[\left(\bigotimes_{j=0}^{i-1} |R\rangle_{d_j}\langle R| \otimes |L\rangle_{d_i}\langle L| \right) \mathbf{H}(i) \otimes T_L T_R^i \right] \\ &= \left[\sum_{i=0}^{\infty} \left(\bigotimes_{j=0}^{i-1} |R\rangle_{d_j}\langle R| \otimes |L\rangle_{d_i}\langle L| \right) \otimes T_L T_R^i \right] (\mathbf{H} \otimes I) \end{aligned} \tag{7.26}$$

where $\mathcal{H}_d = span\{|L\rangle, |R\rangle\}$, $\mathcal{H}_p = span\{|n\rangle : n \in \mathbb{Z}\}$, I is the identity operator in the position Hilbert space $\mathcal{H}_p$, $\mathbf{H}(i)$ is as in equation (7.25), and

$$\mathbf{H} = \sum_{i=0}^{\infty} \mathbf{H}(i)$$

is the extension of H in the free Fock space $\mathcal{F}(\mathcal{H}_d)$ over the direction Hilbert space $\mathcal{H}_d$.

(ii) *For each $i \geq 0$, we compute the symmetrization:*

$$\begin{aligned}
&\mathbb{S}\left(\bigotimes_{j=0}^{i-1} |R\rangle_{d_j}\langle R| \otimes |L\rangle_{d_i}\langle L|\right) \\
&= \frac{1}{(i+1)!}\sum_{\pi} P_\pi \left(\bigotimes_{j=0}^{i-1} |R\rangle_{d_j}\langle R| \otimes |L\rangle_{d_i}\langle L|\right) P_\pi^{-1} \\
&\quad (\text{ where } \pi \text{ traverses over all permutations of } 0,1,\dots,i) \\
&= \frac{1}{i+1}\sum_{j=0}^{i}(|R\rangle_{d_0}\langle R| \otimes \dots \otimes |R\rangle_{d_{j-1}}\langle R| \otimes |L\rangle_{d_j}\langle L| \\
&\qquad \otimes |R\rangle_{d_{j+1}}\langle R| \otimes \dots \otimes |R\rangle_{d_i}\langle R|) \\
&\stackrel{\triangle}{=} G_i.
\end{aligned}$$

Therefore, the symmetric semantics of the unidirectionally recursive Hadamard walk is

$$\mathbb{S}([\![X]\!]) = \left(\sum_{i=0}^{\infty} G_i \otimes T_L T_R^i\right)(\mathbf{H} \otimes I).$$

Example 7.7.2 (Bidirectionally recursive Hadamard walk). *Let us consider the semantics of the bidirectionally recursive Hadamard walk. Recall from Example 7.2.1 that it is declared by equations:*

$$\begin{cases} X \Leftarrow T_L[p] \oplus_{H[d]} (T_R[p]; Y), \\ Y \Leftarrow (T_L[p]; X) \oplus_{H[d]} T_R[p]. \end{cases} \tag{7.27}$$

To simplify the presentation, we first introduce several notations. For any string $\Sigma = \sigma_0\sigma_1 \dots \sigma_{n-1}$ of symbols L and R, its dual is defined to be

$$\overline{\Sigma} = \overline{\sigma_0}\overline{\sigma_1} \dots \overline{\sigma_{n-1}}$$

where $\overline{L} = R$ and $\overline{R} = L$. We write the pure state

$$|\Sigma\rangle = |\sigma_0\rangle_{d_0} \otimes |\sigma_1\rangle_{d_1} \otimes \dots \otimes |\sigma_{n-1}\rangle_{d_{n-1}}$$

in the space $\mathcal{H}_d^{\otimes n}$. Then its density operator representation is

$$\rho_\Sigma = |\Sigma\rangle\langle\Sigma| = \bigotimes_{j=0}^{n-1} |\sigma_j\rangle_{d_j}\langle\sigma_j|.$$

Moreover, we write the composition of left and right translations:

$$T_\Sigma = T_{\sigma_{n-1}} \dots T_{\sigma_1} T_{\sigma_0}.$$

(i) *The semantics of procedures X and Y in the free Fock space are*

$$\begin{aligned} [\![X]\!] &= \left[\sum_{n=0}^{\infty}\left(\rho_{\Sigma_n}\otimes T_n\right)\right]\left(\mathbf{H}\otimes I_p\right), \\ [\![Y]\!] &= \left[\sum_{n=0}^{\infty}\left(\rho_{\overline{\Sigma_n}}\otimes T'_n\right)\right]\left(\mathbf{H}\otimes I_p\right), \end{aligned} \tag{7.28}$$

where $\mathbf{H}$ *is as in Example 7.7.1, and*

$$\Sigma_n = \begin{cases} (RL)^k L & \text{if } n = 2k+1, \\ (RL)^k RR & \text{if } n = 2k+2, \end{cases}$$

$$T_n = T_{\Sigma_n} = \begin{cases} T_L & \text{if } n \text{ is odd}, \\ T_R^2 & \text{if } n \text{ is even}, \end{cases}$$

$$T'_n = T_{\overline{\Sigma_n}} = \begin{cases} T_R & \text{if } n \text{ is odd}, \\ T_L^2 & \text{if } n \text{ is even}. \end{cases}$$

It is clear from equations (7.26) and (7.28) that the behaviors of unidirectionally and bidirectionally recursive Hadamard walks are very different: the former can go to any one of the positions $-1, 0, 1, 2, \dots$, *but in the latter walk* X *can only go to the positions* -1 *and* 2, *and* Y *can only go to the positions* 1 *and* -2.

(ii) *The symmetric semantics of the bidirectionally recursive Hadamard walk specified by equaltion (7.27) is:*

$$\begin{aligned} [\![X]\!] &= \left[\sum_{n=0}^{\infty}\left(\gamma_n\otimes T_n\right)\right]\left(\mathbf{H}\otimes I_p\right), \\ [\![Y]\!] &= \left[\sum_{n=0}^{\infty}\left(\delta_n\otimes T_n\right)\right]\left(\mathbf{H}\otimes I_p\right) \end{aligned}$$

where:

$$\begin{aligned} \gamma_{2k+1} &= \frac{1}{\binom{2k+1}{k}}\sum_{\Gamma}\rho_{\Gamma}, \\ \delta_{2k+1} &= \frac{1}{\binom{2k+1}{k}}\sum_{\Delta}\rho_{\Delta} \end{aligned}$$

with Γ ranging over all strings of $(k+1)$ Ls and k Rs and Δ ranging over all strings of k Ls and $(k+1)$ Rs, and

$$\gamma_{2k+2} = \frac{1}{\begin{pmatrix} 2k+2 \\ k \end{pmatrix}} \sum_{\Gamma} \rho_{\Gamma},$$

$$\sigma_{2k+2} = \frac{1}{\begin{pmatrix} 2k+2 \\ k \end{pmatrix}} \sum_{\Delta} \rho_{\Delta}$$

with Γ ranging over all strings of k Ls and $(k+2)$ Rs and Δ ranging over all strings of $(k+2)$ Ls and k Rs.

(iii) *Finally, we consider the principal system semantics of the bidirectionally recursive Hadamard walk. Suppose that it starts from the position* 0.

(a) *If the "coins" are bosons initialized in state*

$$|\Psi\rangle = |L, L, \ldots, L\rangle_{+} = |L\rangle_{d_0} \otimes |L\rangle_{d_1} \otimes \ldots \otimes |L\rangle_{d_{n-1}},$$

then we have

$$[\![X]\!]_{sym}(|\Psi\rangle \otimes |0\rangle) = \begin{cases} \dfrac{1}{\sqrt{2^n}\begin{pmatrix} 2k+1 \\ k \end{pmatrix}} & \sum_{\Gamma} |\Gamma\rangle \otimes |-1\rangle \\ & \quad \text{if } n = 2k+1, \\ \dfrac{1}{\sqrt{2^n}\begin{pmatrix} 2k+2 \\ k \end{pmatrix}} & \sum_{\Delta} |\Delta\rangle \otimes |2\rangle \\ & \quad \text{if } n = 2k+2, \end{cases}$$

where Γ traverses over all strings of $(k+1)$ L's and k R's, and Δ traverses over all strings of k L's and $(k+2)$ R's. Therefore, the principal system semantics with the "coin" initialisation $|\Psi\rangle$ is:

$$[\![X, \Psi]\!](|0\rangle) = \begin{cases} \frac{1}{2^n}|-1\rangle\langle -1| & \text{if } n \text{ is odd,} \\ \frac{1}{2^n}|2\rangle\langle 2| & \text{if } n \text{ is even.} \end{cases}$$

(b) *For each single-particle state $|\psi\rangle$ in $\mathcal{H}_d$, the corresponding coherent state of bosons in the symmetric Fock space $\mathcal{F}_{+}(\mathcal{H}_d)$ over $\mathcal{H}_d$ is defined as*

$$|\psi\rangle_{coh} = \exp\left(-\frac{1}{2}\langle\psi|\psi\rangle\right) \sum_{n=0}^{\infty} \frac{[a^{\dagger}(\psi)]^n}{n!} |\mathbf{0}\rangle$$

where $|\mathbf{0}\rangle$ is the vacuum state and $a^{\dagger}(\cdot)$ the creation operator. If the "coins" are initialized in the coherent state $|L\rangle_{coh}$ of bosons corresponding to $|L\rangle$, then we have:

$$
\begin{aligned}
&[\![X]\!]_{sym}(|L\rangle_{\ coh} \otimes |0\rangle) \\
&= \frac{1}{\sqrt{e}}\left(\sum_{k=0}^{\infty} \frac{1}{\sqrt{2^{2k+1}\binom{2k+1}{k}}} \sum_{\Gamma_k} |\Gamma_k\rangle\right) \otimes |-1\rangle \\
&\quad + \frac{1}{\sqrt{e}}\sum_{k=0}^{\infty}\left(\frac{1}{\sqrt{2^{2k+2}\binom{2k+2}{k}}} \sum_{\Delta_k} |\Delta_k\rangle\right) \otimes |2\rangle,
\end{aligned}
$$

where Γ_k ranges over all strings of $(k+1)$ Ls and k Rs, and Δ_k ranges over all strings of k Ls and $(k+2)$ Rs. So, the principal system semantics with "coin" initialization $|L\rangle_{\ coh}$ is:

$$
\begin{aligned}
[\![X, L_{\ coh}]\!](|0\rangle) &= \frac{1}{\sqrt{e}}\left(\sum_{k=0}^{\infty}\frac{1}{2^{2k+1}}|-1\rangle\langle -1| + \sum_{k=0}^{\infty}\frac{1}{2^{2k+2}}|2\rangle\langle 2|\right) \\
&= \frac{1}{\sqrt{e}}\left(\frac{2}{3}|-1\rangle\langle -1| + \frac{1}{3}|2\rangle\langle 2|\right).
\end{aligned}
$$

It is interesting to compare termination in subclauses (a) and (b) of clause (iii) in this example. In case (a), the "coins" start in an n-particle state, so the quantum recursive program X terminates within n steps; that is, termination probability within n steps is $p_T^{(\leq n)} = 1$. But in case (b), the "coins" start in a coherent state, and program X does not terminate within a finite number of steps although it almost surely terminates; that is, $p_T^{(\leq n)} < 1$ for all n and $\lim_{n\to\infty} p_T^{(\leq n)} = 1$.

To conclude this section, we point out that only the behavior of the two simplest among the recursive quantum walks defined in Section 7.2 were examined here. The analysis of the others, in particular the recursive quantum walks defined by equations (7.7) and (7.9), seems very difficult, and is left as a topic for further studies.

7.8 QUANTUM WHILE-LOOPS (WITH QUANTUM CONTROL)

The general form of quantum recursion was carefully studied in previous sections. In this section, as an application of the theory developed before, we consider a special class of quantum recursive programs, namely quantum loops with quantum control flows.

Arguably, the **while**-loop is the simplest and most popular form of recursion used in various programming languages. Recall that in classical programming, the **while**-loop:

$$\textbf{while}\ b\ \textbf{do}\ S\ \textbf{od} \tag{7.29}$$

can be seen as the recursive program X declared by the equation:

$$X \Leftarrow \textbf{if } b \textbf{ then } S; X \textbf{ else skip fi} \tag{7.30}$$

where b is a Boolean expression. In Chapter 3, we defined a measurement-based **while**-loop:

$$\textbf{while } M[\overline{q}] = 1 \textbf{ do } P \textbf{ od} \tag{7.31}$$

as a quantum extension of loop (7.29), where M is a quantum measurement. As pointed out before, the control flow of this loop is classical before it is determined by the outcomes of measurement M.

It was shown that loop (7.31) is the solution of the recursive equation with classical case statement in quantum programming:

$$\begin{aligned} X \Leftarrow \textbf{if } & M[\overline{q}] = 0 \rightarrow \textbf{ skip} \\ \square & \qquad\quad 1 \rightarrow P; X \\ \textbf{fi} & \end{aligned} \tag{7.32}$$

In Chapter 6, we pointed out that, in the quantum setting, the notion of case statement splits into two different versions: (a) the one used in equation (7.32), and (b) the quantum case statement with quantum control. Furthermore, the notion of quantum choice was defined based on the quantum case statement. Now we can define a kind of quantum **while**-loop by using quantum case statement and quantum choice in the place of the classical case statement **if** ... **then** ... **else** ... **fi** in equation (7.30) or (7.32). Inherited from the quantum case statement and quantum choice, the control flow of this new quantum loop is genuinely quantum.

Example 7.8.1 (Quantum **while**-loops with quantum control).

(i) *The first form of quantum **while**-loop with quantum control:*

$$\textbf{qwhile } [c] = |1\rangle \textbf{ do } U[q] \textbf{ od} \tag{7.33}$$

is defined to be the quantum recursive program X declared by

$$\begin{aligned} X \Leftarrow \textbf{qif}[c] \; & |0\rangle \rightarrow \textbf{skip} \\ \square \quad & |1\rangle \rightarrow U[q]; X \\ \textbf{fiq} & \end{aligned} \tag{7.34}$$

where c is a quantum "coin" variable denoting a qubit, q is a principal quantum variable, and U is a unitary operator in the state Hilbert space $\mathcal{H}_q$ of system q.

(ii) *The second form of quantum **while**-loop with quantum control:*

$$\textbf{qwhile } V[c] = |1\rangle \textbf{ do } U[q] \textbf{ od} \tag{7.35}$$

is defined to be the quantum recursive program X declared by

$$\begin{aligned} X \Leftarrow \mathbf{skip} \oplus_{V[c]} (U[q]; X) \\ \equiv V[c]; \mathbf{qif}[c]\ |0\rangle \rightarrow \mathbf{skip} \\ \square \quad |1\rangle \rightarrow U[q]; X \\ \mathbf{fiq} \end{aligned} \tag{7.36}$$

Note that the quantum recursive equation (7.36) is obtained by replacing the quantum case statement **qif** . . . **fiq** *in equation (7.34) with the quantum choice* $\oplus_{V[c]}$.

(iii) *Actually, quantum loops* (7.33) *and* (7.35) *are not very interesting because there is not any interaction in them between the quantum "coin" c and the principal quantum system q. This situation corresponds to the trivial case of classical loop* (7.30) *where the loop guard b is irrelevant to the loop body S. The classical loop* (7.30) *becomes truly interesting only when the loop guard b and the loop body S share some program variables. Likewise, a much more interesting form of quantum* **while***-loop with quantum control is*

$$\mathbf{qwhile}\ W[c; q] = |1\rangle\ \mathbf{do}\ U[q]\ \mathbf{od} \tag{7.37}$$

which is defined to be the program X declared by the quantum recursive equation

$$\begin{aligned} X \Leftarrow \ W[c, q]; \mathbf{qif}[c]\ |0\rangle \rightarrow \mathbf{skip} \\ \square \quad |1\rangle \rightarrow U[q]; X \\ \mathbf{fiq} \end{aligned}$$

where W is a unitary operator in the state Hilbert space $\mathcal{H}_c \otimes \mathcal{H}_q$ *of the composed system of the quantum "coin" c and the principal system q. The operator W describes the interaction between the "coin" c and the principal system q. It is obvious that the loop* (7.37) *degenerates to the loop* (7.35) *whenever* $W = V \otimes I$, *where I is the identity operator in* $\mathcal{H}_q$.

It is very interesting to compare quantum loops in the preceding example with the measurement-based **while**-loop (7.31). First of all, we stress once again that the control flow of the former is defined by a quantum "coin" and thus it is quantum; in contrast, the control flow of the latter is determined by the outcomes of a measurement and thus it is classical. In order to further understand the difference between loops (7.31) and (7.37), let us have a close look at the semantics of loop (7.37).

- A routine calculation shows that the semantics of the loop (7.37) in the free Fock space is the operator:

$$\llbracket X \rrbracket = \sum_{k=1}^{\infty} (|1\rangle_{c_0}\langle 1| \otimes (|1\rangle_{c_1}\langle 1| \otimes \ldots (|1\rangle_{c_{k-2}}\langle 1| \otimes (|0\rangle_{c_{k-1}}\langle 0| \otimes U^{k-1}[q])$$
$$W[c_{k-1}, q])W[c_{k-2}, q]\ldots)W[c_1, q])W[c_0, q]$$
$$= \sum_{k=1}^{\infty}\left[\left(\bigotimes_{j=0}^{k-2} |1\rangle_{c_j}\langle 1| \otimes |0\rangle_{c_{k-1}}\langle 0| \otimes U^{k-1}[q]\right)\prod_{j=0}^{k-1} W[c_j, q]\right].$$

- Furthermore, the symmetric semantics of the loop is:

$$\llbracket X \rrbracket_{sym} = \sum_{k=1}^{\infty}\left[\left(\mathbf{A}(k) \otimes U^{k-1}[q]\right)\prod_{j=0}^{k-1} W[c_j, q]\right],$$

where:

$$\mathbf{A}(k) = \frac{1}{k}\sum_{j=0}^{k-1} |1\rangle_{c_0}\langle 1| \otimes \ldots \otimes |1\rangle_{c_{j-1}}\langle 1| \otimes |0\rangle_{c_j}\langle 0|$$
$$\otimes |1\rangle_{c_{j+1}}\langle 1| \otimes \ldots \otimes |1\rangle_{c_{k-1}}\langle 1|.$$

- Now we consider the principal semantics of loop (7.37) in a special case. Let q be a qubit, $U = H$ (the Hadamard gate) and $W = CNOT$ (the controlled-NOT). If the "coins" are initialized in the n-boson state

$$|\Psi_n\rangle = |0, 1, \ldots, 1\rangle_+ = \frac{1}{n}\sum_{j=0}^{n-1} |1\rangle_{c_0}\ldots|1\rangle_{c_{j-1}}|0\rangle_{c_j}|1\rangle_{c_{j+1}}\ldots|1\rangle_{c_{n-1}}$$

and the principal system q starts in state $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, then we have:

$$|\Phi_n\rangle \triangleq \llbracket X \rrbracket_{sym}(|\Psi_n\rangle \otimes |-\rangle) = (-1)^n\frac{1}{n}|\Psi_n\rangle \otimes |\psi_n\rangle$$

where

$$|\psi_n\rangle = \begin{cases} |+\rangle & \text{if } n \text{ is even,} \\ |-\rangle & \text{if } n \text{ is odd.} \end{cases}$$

Consequently, the principal system semantics is

$$\llbracket X, \Psi_n \rrbracket(|-\rangle) = tr_{\mathcal{F}_T(\mathcal{H}_v)}(|\Phi_n\rangle\langle\Phi_n|) = \frac{1}{n^3}|\psi_n\rangle\langle\psi_n|.$$

We conclude this chapter by leaving a series of problems for further studies.

Problem 7.8.1. *Although in this chapter we presented several examples of quantum recursion, it is still not well understood what kind of computational problems can be solved more conveniently by using quantum recursion. Another* *important question is: what kind of physical systems can be used to implement quantum recursion where new "coins" must be continuously created?*

Problem 7.8.2. *We do not even fully understand how should a quantum recursion use its "coins" in its computational process. In the definition of the principal system semantics of a recursive program (Definition 7.6.1), a state $|\Psi\rangle$ in the Fock space of "coin" is given a priori. This means that the states of a "coin" and its copies are given once for all. Another possibility is that the states of the copies of a "coin" are created step by step. As an example, let us consider the recursive program X declared by*

$$\begin{aligned} X \Leftarrow a_c^\dagger(|0\rangle); R_y[c,p]; \mathbf{qif}\ [c]\ |0\rangle &\rightarrow \mathbf{skip} \\ \square\ |1\rangle &\rightarrow T_R[p]; X \\ \mathbf{fiq} & \end{aligned}$$

where $a^\dagger$ is the creation operator, c is a "coin" variable with state space $\mathcal{H}_c = span\{|0\rangle, |1\rangle\}$, the variable p and operator T_R are as in the Hadamard walk,

$$R_y[c,p] = \sum_{n=0}^{\infty} \left[R_y\left(\frac{\pi}{2^{n+1}}\right) \otimes |n\rangle_p\langle n| \right]$$

and $R_y(\theta)$ is the rotation of a qubit about the y-axis in the Bloch sphere (see Example 2.2.3). Intuitively, $R_y[c,p]$ is a controlled rotation where the position of p is used to determine the rotated angle. It is worth noting that this program X is a quantum loop defined in equation (7.37) but modified by adding a creation operator at the beginning. Its initial behavior starting at position 0 with the "coin" c being in the vacuum state $|\mathbf{0}\rangle$ is visualized by the following transitions:

$$\begin{aligned} |\mathbf{0}\rangle|0\rangle_p \xrightarrow{a_d^\dagger(|0\rangle)} |0\rangle|0\rangle_p &\xrightarrow{R_x[d,p]} \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)|0\rangle_p \\ &\xrightarrow{\mathbf{qif}\ldots\mathbf{fiq}} \frac{1}{\sqrt{2}}\left[\langle E, |0\rangle|0\rangle_p\rangle + \langle X, |1\rangle|1\rangle_p\rangle\right]. \end{aligned}$$

The first configuration at the end of the preceding equation terminates, but the second continues the computation as follows:

$$|1\rangle|1\rangle_p \xrightarrow{a_d^\dagger(|0\rangle)} |0,1\rangle_v|0\rangle_p \xrightarrow{R_x[d,p]} \cdots .$$

It is clear from this example that the computation of a recursive program with the creation operator is very different from that without it. A careful study of quantum recursions that allow the creation operator to appear in their syntax is certainly interesting.

Problem 7.8.3. *The theory of quantum recursive programs in this chapter was developed in the language of Fock spaces. The second quantization method* *can be equivalently presented using the occupation number representation (see [163], Section 2.1.7; also see Definition 7.3.9 for a related notion – the particle number operator). Give an occupation number restatement of the theory of quantum recursion.*

Problem 7.8.4. *Floyd-Hoare logic for quantum programs with classical control flows was presented in Chapter 4. How can you develop a Floyd-Hoare logic for quantum programs with quantum control flows defined in the last and this chapter?*

7.9 BIBLIOGRAPHIC REMARKS

Quantum recursion with classical control flow (in the superposition-of-data paradigm) was discussed in Section 3.4. This chapter can be seen as the counterpart of Section 3.4 in the superposition-of-programs paradigm, dealing with quantum recursion with quantum control flow. The exposition of this chapter is based on the draft paper [222].

Quantum recursion studied both in Section 3.4 and in this chapter is quantum generalization of classical recursion in imperative programming. Two good references for the theory of classical recursive programs are [21] (Chapters 4 and 5) and [158] (Chapter 5). The book [162] contains many examples of recursive programs. It is very interesting to examine the quantum counterparts of these examples.

The lambda calculus is a suitable formalism for coping with recursion and high-order computation, and it provides a solid basis for functional programming. Both the lambda calculus and functional programming have been extended into the quantum setting; see Section 8.3 and the references cited there. But so far, only quantum recursion with classical control has been considered in functional quantum programming.

The main mathematical tool employed in this chapter is the second quantization method. The materials about second quantization in Section 7.3 are standard and can be found in many textbooks of advanced quantum mechanics. Our presentation in Section 7.3 largely follows [163].