Quantisation of Abstract Data Types

MINGSHENG YING, University of Technology Sydney, Australia
ZHICHENG ZHANG, Griffith University, Australia
KEAN CHEN, University of Pennsylvania, USA

In this paper, we introduce a notion of abstract quantum data type within the framework of universal algebra. This notion provides an algebraic foundation for describing data abstraction in quantum programming. We formally define a quantisation of classical data types and show that their equational specifications can be soundly lifted to the quantum setting. Two standard quantisation methods for classical functions, namely the bit oracle and the phase oracle, arise as special cases of this general construction. We illustrate the framework with applications to quantum arrays and quantum error-correcting codes, showing how they can be understood through the lens of data-type quantisation. We further establish conditions under which quantisation preserves structural relationships and constructions of classical data types, including embeddings, isomorphisms, and products.

CCS Concepts: • **Software and its engineering → Abstract data types**; • **Theory of computation → Quantum computation theory**.

Additional Key Words and Phrases: Quantum programming, abstract data types, universal algebra, quantisation, inheritance.

ACM Reference Format:
Mingsheng Ying, Zhicheng Zhang, and Kean Chen. 2026. Quantisation of Abstract Data Types. 1, 1 (September 2026), 37 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Algorithms + *Data Structures* = *Programs*

— Niklaus Wirth [38]

Quantum programming research has been conducted for more than 30 years, beginning with Knill's proposal of a set of conventions for writing quantum pseudocode [23], and it has become increasingly active in recent years [18, 39]. The majority of this research has focused on the algorithmic aspect of quantum programs. However, programming methodology must also study the structures, representations, and operations of data involved in large and complex programs [38]. To date, the data aspect of quantum programming has been less investigated.

1.1 Data types and data structures in quantum programming

Several basic quantum data types that abstract qubits and quantum registers are widely used in practical quantum programming languages, e.g. Qiskit [19], Q# [35], Cirq [6], Quipper [14], Guppy [24], Silq [3], Qrisp [32] and isQ [16]. Some more advanced quantum data types have been defined in theoretical research on quantum programming; for example, inductive quantum data types were introduced by Péchoux, Perdrix, Rennela and Zamdzhiev [31]. On the other hand, from the viewpoint of developer-friendly programming, a quantum programming language, Rhyme, with

Authors' Contact Information: Mingsheng Ying, Centre for Quantum Software and Information, University of Technology Sydney, Sydney, Australia, Mingsheng.Ying@uts.edu.au; Zhicheng Zhang, School of Information and Communication Technology, Griffith University, Brisbane, Australia, iszczhang@gmail.com; Kean Chen, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, USA, keanchen.gan@gmail.com.

2026. ACM XXXX-XXXX/2026/9-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

richer data types—including quantum extensions of floats, characters, arrays, and strings—was proposed by Varga, Aragonés-Soria and Oriol [36].

Various quantum data structures have been introduced for the design of quantum algorithms (see e.g. [1, 20, 21, 28, 37]). Yuan and Carbin's Tower [40] provides the first programming language support for data structures in quantum superposition, including using random-access memory and recursive definitions of operations. A library of quantisations of data structures *lists*, *stacks*, *queues*, *strings*, and *sets* is programmed in Tower.

1.2 Data abstraction: from classical to quantum programming

Data abstraction is a foundational concept in classical programming that allows a programmer to focus on what an object does rather than how it does it. It is the process of hiding complex implementation details while exposing only the essential features of an object or data type. Thus, it allows developers to work with abstract data types (e.g. integers, arrays, or linked lists) without worrying about memory layout or hardware details [25, 26].

We believe that data abstraction will play an even more critical role in quantum programming [8, 30] for at least the following two reasons:

(1) Concepts such as superposition, entanglement, and measurement impose physical constraints that quantum data must respect but programmers may overlook or ignore. Therefore, data abstraction is essential for writing correct and scalable quantum software.
(2) Hardware independence is essential in quantum programming because quantum hardware is evolving rapidly. High-level abstractions allow quantum algorithms to be expressed without being tied to a specific quantum device or gate set. Compilers then map these abstract representations onto real hardware, optimising for noise, connectivity, and error rates.

Building upon previous work on quantum data structures [40] and quantum data types [36], we then naturally ask the following:

Question 1. How can we realise data abstraction in quantum programming?

1.3 Abstract data types

The introduction of abstract data types (ADTs) is a key way of realising data abstraction [17]. An ADT specifies a data type by focusing on what it does rather than how it is implemented, and thus the internal representation is hidden from the user. ADTs are defined from a user's viewpoint. In contrast, data structures are defined from an implementer's viewpoint as concrete representations of data. ADTs encourage programmers to think at a higher level of abstraction. The main advantages of ADTs include:

- By separating the interface from the implementation, programmers can change how an ADT is implemented without affecting the rest of the program. For example, a stack ADT is defined by operations such as push, pop, and peek, regardless of whether it is implemented using an array or a linked list. This makes code easier to maintain, understand, and reuse.
- ADTs support modularity so that users interact only through well-defined operations. This can significantly improve software reliability because errors caused by directly manipulating data are reduced.

Turning to the realm of quantum programming, Question 1 should be concretised to the following:

Question 2. How can we define *abstract quantum data types*? Is there a principled way to lift classical data types to the quantum setting—what one might call the *quantisation of data types*?

1.4 Contributions of this paper

The aim of this paper is to introduce a method for the quantisation of data types that supports data abstraction in quantum programming. The main difference between our work and previous studies [36, 40] is that our approach operates at the level of abstract data types, whereas [36, 40] focus on less abstract levels, such as data structures and concrete data types. The technical contributions of this paper are threefold:

(1) We formally define the notion of an abstract quantum data type within a universal-algebraic framework.
(2) We introduce a quantisation of classical ADTs and prove that their equational specifications can be soundly lifted to the quantum setting. This notion includes generalisations of two basic quantisation methods for classical functions as important special cases.
(3) We identify conditions ensuring that quantisation is compatible with structural relationships and constructions of classical data types, including embeddings, isomorphisms, and products.

We complement these results with illustrative applications to quantum arrays and quantum error correction.

We hope that the introduction of abstract quantum data types will help programmers design clean, flexible, and robust software systems for quantum computers. In particular, *inheritance* is a fundamental feature of object-oriented programming [5, 12], playing a crucial role in managing software complexity through mechanisms such as code reuse and hierarchical classification. The embedding result in Contribution (3) shows that certain algebraic inheritance relationships between classical data types can, under suitable conditions, be preserved under quantisation and transferred to their quantum counterparts.

1.5 Organisation of the paper

For the convenience of the reader, the basic concepts of ADTs are reviewed in Section 2. The notion of a quantum data type is introduced in Section 3, where several simple examples are also given to illustrate how quantum ADTs can be used in quantum programming. Section 4 develops a quantisation of classical ADTs and shows how equational specifications can be lifted to the quantum setting. It also presents two standard quantisation methods as special cases and explains their relationship through phase kickback. The framework is then illustrated through quantum arrays in Section 5 and quantum error correction in Section 6. Sections 7 and 8 investigate the structural properties of quantisation through embeddings and products, respectively. The paper concludes with a summary of the results and a brief discussion about topics for further research. The proofs of the main theorems and propositions are deferred to Appendix A.

2 ADTs in Classical Computing

In this section, we briefly review the basics of ADTs, following the textbook [9]. For more details, the reader can consult related chapters of [9].

A data type is a collection of data domains, designated basic data items, and operations on these domains. However, a programmer may not be interested in the concrete representation of a data type but only in its properties at an abstract level. Thus, the notion of ADT is introduced to specify the kinds of data that can be stored, the operations that can be performed on that data, and the rules those operations satisfy, independently of their concrete representation.

2.1 Basics of Universal Algebra

ADTs can be properly formalised in the mathematical language of universal algebra. A signature specifies the interface of an ADT. Formally, we have:

Definition 2.1 (Signatures). A signature is a pair $Sig = (S, OP)$, where:

(1) S is a set of sorts;
(2) OP is a set of operation symbols. Each symbol $O \in OP$ is assigned input sorts $in(O) \in S^*$ (the set of strings of elements in S, including the empty string ϵ) and an output sort $out(O) \in S$.

We often write $O : s_1, \ldots, s_n \to s$ for $in(O) = s_1, \ldots, s_n$ and $out(O) = s$. In particular, a constant symbol is an operation symbol $c :\to s$ where the input sort $in(c) = \epsilon$ is empty.

Mathematically, a data type can then be modelled as an algebra as defined as follows:

Definition 2.2 (Algebras). An algebra A of a signature $Sig = (S, OP)$, or a Sig-algebra for short, consists of:

(1) for each $s \in S$, A_s is a nonempty set, called the domain of sort s;
(2) for each $O \in OP$ with $O : s_1, \ldots, s_n \to s$, O^A is a mapping $A_{s_1} \times \cdots \times A_{s_n} \to A_s$, called an operation. In particular, for each constant symbol $c :\to s$, c^A is an element of A_s.

Throughout this paper, we only consider finite domains. The rules that the operations of an ADT must follow are usually specified as a family of equations. To define them, for each $s \in S$, we assume a set X_s of variables of sort s. The variable sets X_s are required to be pairwise disjoint and also disjoint from the operation symbols OP. Then we can introduce the following definition.

Definition 2.3 (Terms). The terms are recursively defined as follows:

(1) every constant symbol $c :\to s$ and every variable $x \in X_s$ are terms of sort s;
(2) if $t_1, \ldots, t_n$ are terms of sorts $s_1, \ldots, s_n$, respectively, and $O : s_1, \ldots, s_n \to s \in OP$, then $O(t_1, \ldots, t_n)$ is a term of sort s.

A term without variables is called a ground term. The semantic interpretation of terms in a given algebra is presented through the notion of evaluation defined in the following:

Definition 2.4 (Evaluation). Let A be an Sig-algebra and σ an assignment in A that maps each variable $x \in X_s$ to an element $\sigma(x) \in A_s$ for every sort s. Then the evaluation $eval_\sigma(t)$ of a term t in assignment σ is recursively defined as follows:

(1) if t is a variable x, then $eval_\sigma(t) = \sigma(x)$, and if t is a constant c, then $eval_\sigma(t) = c^A$;
(2) if $t = O(t_1, \ldots, t_n)$, then $eval_\sigma(t) = O^A(eval_\sigma(t_1), \ldots, eval_\sigma(t_n))$.

As noted above, a large class of the rules for the operations in an ADT can be described by equations between terms. The notion of equations can be formally defined in the following:

Definition 2.5 (Equations and Validity).

(1) Let t_1 and t_2 be two terms of the same sort s. If all variables occurring in t_1 and t_2 are among $x_1, \ldots, x_n$, then
$$e = (\forall x_1 : s_1, \ldots, x_n : s_n)(t_1 = t_2) \tag{1}$$
is called an equation, where x_i is a variable of sort s_i for every $i = 1, \ldots, n$.
(2) The equation e is valid in an algebra A, or A satisfies e, written $A \models e$, if for any assignment σ in A, we have $eval_\sigma(t_1) = eval_\sigma(t_2)$.

It is easy to see that the validity of equation e only depends on the values $\sigma(x_i)$ $(i = 1, \ldots, n)$ of assignments σ. In the specifications of ADTs in practical applications, universal quantifiers $(\forall x_1 : s_1, \ldots, x_n : s_n)$ in Equation (1) are often omitted (see Example 2.7).

If both t_1 and t_2 are ground terms, then e is called a ground equation. The validity of a ground equation e in an algebra does not depend on any assignment σ in A.

2.2 Specification of ADTs

Now we can use the algebraic framework defined in the previous subsection to specify ADTs.

Definition 2.6 (Specifications).

(1) A specification is a triple $Spec = (S, OP, E)$, where $Sig = (S, OP)$ is a signature, and E is a set of equations over the signature Sig.
(2) An algebra of the specification $Spec$, or $Spec$-algebra for short, is an algebra A of signature Sig such that $A \models e$ for every $e \in E$.

Besides the equational specifications defined above, there are more sophisticated classes of ADT specifications, such as Horn specifications. In this paper, we mainly consider equational specifications.

For a better understanding of the notions introduced above, let us see the following simple example:

Example 2.7 (Abstract data type of strings). A string is a sequence of basic data items from a given domain, called the alphabet. Let $c_1, \ldots, c_n$ be constant symbols denoting the basic data items. The most useful operations of strings in practical applications are:

- *make*: it converts each basic data item into a string of length 1;
- *concat*: it concatenates two strings;
- *ladd*, *radd*: they add a basic data item to the left and right ends of a string, respectively.

A specification of strings is given as follows:

$$
\begin{aligned}
\mathtt{STRING} = \ & \textbf{sorts} : \mathtt{alphabet}, \mathtt{string} \\
& \textbf{operations} : c_1, \ldots, c_n :\longrightarrow \mathtt{alphabet} \\
& \qquad empty :\longrightarrow \mathtt{string} \\
& \qquad make : \mathtt{alphabet} \longrightarrow \mathtt{string} \\
& \qquad concat : \mathtt{string}, \mathtt{string} \longrightarrow \mathtt{string} \\
& \qquad ladd : \mathtt{alphabet}, \mathtt{string} \longrightarrow \mathtt{string} \\
& \qquad radd : \mathtt{string}, \mathtt{alphabet} \longrightarrow \mathtt{string} \\
& \textbf{equations} : a : \mathtt{alphabet}, s, s_1, s_2, s_3 : \mathtt{string} \\
& \qquad concat(s, empty) = s \\
& \qquad concat(empty, s) = s \\
& \qquad concat(concat(s_1, s_2), s_3) = concat(s_1, concat(s_2, s_3)) \\
& \qquad ladd(a, s) = concat(make(a), s) \\
& \qquad radd(s, a) = concat(s, make(a))
\end{aligned}
$$

We can define a STRING-algebra A as follows:

- Let $A_{\mathtt{alphabet}} = \{a_1, \ldots, a_n\}$ be a domain for the alphabet. Let $A_{\mathtt{string}} = (A_{\mathtt{alphabet}})^*$ be the set of strings of elements in $A_{\mathtt{alphabet}}$, including the empty string ϵ.
- For constant symbols, define $c_i^A = a_i$ for $i = 1, \ldots, n$ and $empty^A = \epsilon$.
- For nonconstant operation symbols, define
  (1) $make^A : A_{\mathtt{alphabet}} \longrightarrow A_{\mathtt{string}}$, $make^A(a) = a$ for all $a \in A_{\mathtt{alphabet}}$;
  (2) $concat^A : A_{\mathtt{string}} \times A_{\mathtt{string}} \longrightarrow A_{\mathtt{string}}$, $concat^A(u, v) = uv$ for all $u, v \in A_{\mathtt{string}}$;
  (3) $ladd^A : A_{\mathtt{alphabet}} \times A_{\mathtt{string}} \longrightarrow A_{\mathtt{string}}$, $ladd^A(a, u) = au$ for all $a \in A_{\mathtt{alphabet}}$ and $u \in A_{\mathtt{string}}$;
  (4) $radd^A : A_{\mathtt{string}} \times A_{\mathtt{alphabet}} \longrightarrow A_{\mathtt{string}}$, $radd^A(u, a) = ua$ for all $a \in A_{\mathtt{alphabet}}$ and $u \in A_{\mathtt{string}}$.

It is easy to check that the above operations satisfy the equations in `STRING`, and hence A is a `STRING`-algebra.

3 Quantum Data Types

From now on, we generalise the algebraic specification techniques for classical ADTs into quantum computing. In this section, we formally define quantum data types in the universal-algebraic framework, analogously to their classical counterparts in Section 2, and illustrate it with some simple examples.

3.1 Basic Definitions

Let us first introduce the notion of a quantum signature as the quantum counterpart of Definition 2.1:

Definition 3.1 (Quantum Signatures). A quantum signature is a triple $QSig = (S_q, \mathcal{K}, \mathcal{U})$, where:

(1) S_q is a set of quantum sorts;
(2) $\mathcal{K}$ is a set of quantum state symbols. Each symbol $|\psi\rangle \in \mathcal{K}$ is assigned sort $sort(|\psi\rangle) \in S_q^+$ (the set of nonempty strings of elements in S_q);
(3) $\mathcal{U}$ is a set of quantum gate symbols. Each symbol $U \in \mathcal{U}$ is assigned sort $sort(U) \in S_q^+$;

Roughly speaking, $\mathcal{K}$ and $\mathcal{U}$ correspond to the constant symbols and operation symbols in Definition 2.1, respectively. But there are some subtle differences between them and their classical counterparts. For a classical constant symbol $c :\to s$, its input sort is empty and its output sort is a single sort $s \in S$. However, for a quantum state symbol $|\psi\rangle$, it is possible that $sort(|\psi\rangle) = s_1, \dots, s_n \in S_q^+$ with $n > 1$. The reason is that $|\psi\rangle$ may denote an entangled state between n subsystems, which cannot be decomposed into a string of the states of individual subsystems. We often write $|\psi\rangle : s_1, \dots, s_n$ for $sort(|\psi\rangle) = s_1, \dots, s_n$. Moreover, for a classical operation symbol $O : s_1, \dots, s_n \to s$, the input sorts $s_1, \dots, s_n$ and the output sort s need not coincide. In contrast, the input and output sorts of a quantum gate symbol U must be the same because it denotes a unitary operator. We write $U : s_1, \dots, s_n$ for $sort(U) = s_1, \dots, s_n$. Indeed, $sort(U)$ is both the input sort and output sort of U.

As the quantum counterparts of algebras described in Definition 2.2, quantum algebras are then defined by taking the domains of sorts to be Hilbert spaces and interpreting quantum state symbols and quantum gate symbols as pure quantum states and unitary operators, respectively.

Definition 3.2 (Quantum Algebras). A quantum algebra $\mathcal{A}$ of signature $QSig = (S_q, \mathcal{K}, \mathcal{U})$ consists of:

(1) for each $s \in S_q$, $\mathcal{H}_s^{\mathcal{A}}$ is a Hilbert space, called the quantum domain of sort s (the superscript $\mathcal{A}$ of $\mathcal{H}_s^{\mathcal{A}}$ is often omitted for simplicity);
(2) for each quantum state symbol $|\psi\rangle \in \mathcal{K}$ with $sort(|\psi\rangle) = s_1, \dots, s_n$, $|\psi\rangle^{\mathcal{A}}$ is a unit vector (i.e. a pure quantum state) in $\mathcal{H}_{s_1} \otimes \cdots \otimes \mathcal{H}_{s_n}$; and
(3) for each quantum gate symbol $U \in \mathcal{U}$ with $sort(U) = s_1, \dots, s_n$, $U^{\mathcal{A}}$ is a unitary operator (i.e. a quantum gate) on $\mathcal{H}_{s_1} \otimes \cdots \otimes \mathcal{H}_{s_n}$.

In this paper, we only consider quantum algebras where state symbols are interpreted as pure states and gate symbols as unitary operators. Of course, Definition 3.2 can be generalized such that state symbols are interpreted as mixed states (i.e. density operators) and gate symbols as noisy quantum gates (i.e. quantum channels) when needed in applications.

Given a quantum signature $QSig$, we assume a set Q of quantum variables. Each $q \in Q$ is assigned a sort $sort(q) \in S_q$. For a string $\overline{q} = q_1, \dots, q_n$ of quantum variables, we define $sort(\overline{q}) = \overline{s} = s_1, \dots, s_n$, where $s_i = sort(q_i)$. When the context is clear, we also use $\overline{q}$ to denote the set $\{q_1, \dots, q_n\}$. Then the notion of term is generalised to the quantum case as follows:

Definition 3.3 (Quantum Circuit Formulas and Terms).

(1) Quantum circuit formulas C and their quantum variables $qv(C)$ are inductively defined as follows:
  (a) if $U : s_1, \dots, s_n$ is a quantum gate symbol and $\overline{q} = q_1, \dots, q_n$ is a string of distinct quantum variables with $sort(q_i) = s_i$ for $i = 1, \dots, n$, then $C = U[\overline{q}]$ is a quantum circuit formula, and $qv(C) = \overline{q}$;
  (b) if C_1 and C_2 are quantum circuit formulas, then $C = C_1; C_2$ is a quantum circuit formula, and $qv(C) = qv(C_1) \cup qv(C_2)$.

(2) Suppose C is a quantum circuit formula. For $i = 1, \dots, n$, let $|\psi_i\rangle : \overline{s_i}$ be a quantum state symbol and $\overline{q_i}$ be a string of distinct quantum variables with $sort(\overline{q_i}) = \overline{s_i}$ and $\overline{q_i} \cap \overline{q_j} = \emptyset$ for $i \neq j$. Then $\tau = C\, |\psi_1\rangle_{\overline{q_1}} \dots |\psi_n\rangle_{\overline{q_n}}$ is a quantum term, and $qv(\tau) = \overline{q_1} \cup \dots \cup \overline{q_n} \cup qv(C)$. When the context is clear, $|\psi\rangle_{\overline{q}}$ may be used to denote the string $|\psi_1\rangle_{\overline{q_1}} \dots |\psi_n\rangle_{\overline{q_n}}$ of quantum state symbols, where $\overline{q} = \overline{q_1}, \dots, \overline{q_n}$. Define the set of free variables in quantum term τ to be $fv(\tau) = qv(\tau) \setminus \overline{q}$.

The definition of quantum terms above differs slightly from its classical counterpart in Definition 2.3. Obviously, a quantum circuit formula C stands for a (combinational) quantum circuit constructed from quantum gates denoted by elements of $\mathcal{U}$ applied to quantum variables in Q. Furthermore, a quantum term $C\, |\psi_1\rangle_{\overline{q_1}} \dots |\psi_n\rangle_{\overline{q_n}}$ stands for a quantum circuit formula C with quantum variables $\overline{q} = \overline{q_1}, \dots, \overline{q_n}$ initialised in the state $|\psi_1\rangle \dots |\psi_n\rangle$, where each $|\psi_i\rangle$ can be entangled. Conversely, a quantum circuit formula C can be seen as a quantum term with $\overline{q} = \emptyset$. A string of quantum state symbols $|\psi\rangle_{\overline{q}} = |\psi_1\rangle_{\overline{q_1}} \dots |\psi_n\rangle_{\overline{q_n}}$ can also be seen as a quantum term with empty C.

Given an algebra $\mathcal{A}$ of a quantum signature *QSig*, for any subset $Q' \subseteq Q$ of quantum variables, we write:

$$\mathcal{H}_{Q'} = \bigotimes_{q \in Q'} \mathcal{H}_{sort(q)}$$

for the Hilbert space of the composite system Q'. Then an assignment of quantum variables in Q' can be understood as a quantum state $|\varphi\rangle \in \mathcal{H}_{Q'}$, although there is a basic difference between it and an assignment of classical variables; that is, the values of two classical variables are independent, but $|\varphi\rangle$ could contain entanglement. Furthermore, the notion of evaluation can be generalised straightforwardly to the quantum case. As usual, for a unitary operator U, we implicitly identify it with its extension $U \otimes \mathbb{1}$ by tensoring with an identity operator $\mathbb{1}$.

Definition 3.4 (Evaluation).

(1) Let C be a quantum circuit formula. Then its evaluation $eval(C)$ in algebra $\mathcal{A}$ is a unitary operator on $\mathcal{H}_{qv(C)}$ recursively defined as follows:
  (a) if $C = U[\overline{q}]$ for some $U \in \mathcal{U}$, then $eval(C) = U^{\mathcal{A}}_{\overline{q}}$ (unitary operator $U^{\mathcal{A}}$ acting on the quantum system denoted by $\overline{q}$);
  (b) if $C = C_1; C_2$, then $eval(C_1; C_2) = eval(C_2) \circ eval(C_1)$ is a composition of unitary operators.

(2) Let $\tau = C\, |\psi_1\rangle_{\overline{q_1}} \dots |\psi_n\rangle_{\overline{q_n}}$ be a quantum term. Suppose $\overline{q} = \overline{q_1}, \dots, \overline{q_n}$ and $\overline{r}$ is a string of quantum variables such that $\overline{r} \supseteq fv(\tau)$ and $\overline{r} \cap \overline{q} = \emptyset$. Let $|\varphi\rangle$ be a quantum state in $\mathcal{H}_{\overline{r}}$. Then the evaluation of τ in state $|\varphi\rangle$ is defined as the quantum state:

$$eval_{|\varphi\rangle}(\tau) = eval(C)\Big(|\psi_1\rangle^{\mathcal{A}}_{\overline{q_1}} \otimes \dots \otimes |\psi_n\rangle^{\mathcal{A}}_{\overline{q_n}} \otimes |\varphi\rangle_{\overline{r}}\Big). \tag{2}$$

Here, $eval(C) = \mathbb{1}$ if C is empty.

The notion of equation can be smoothly generalised to the quantum setting too:

Definition 3.5 (Equations). Let $\tau_1 = C_1 |\psi_1\rangle_{\overline{q_1}}$ and $\tau_2 = C_2 |\psi_2\rangle_{\overline{q_2}}$ be two quantum terms that satisfy $\overline{q_1} \cap fv(\tau_2) = \overline{q_2} \cap fv(\tau_1) = \emptyset$. Suppose $\overline{r} = fv(\tau_1) \cup fv(\tau_2)$ and $sort(\overline{r}) = \overline{s}$. Then, for $\overline{p} \subseteq qv(\tau_1) \cap qv(\tau_2)$,

$$e = (\forall \overline{r} : \overline{s})(\tau_1 = \tau_2) \Downarrow \overline{p} \tag{3}$$

is called an equation, and $\overline{p}$ are called the principal variables of e. In particular, if $\overline{p} = qv(\tau_1) \cap qv(\tau_2)$, then $\Downarrow \overline{p}$ can be dropped from Equation (3).

Similar to the case of classical equations, universal quantifiers $(\forall \overline{r} : \overline{s})$ in Equation (3) are often omitted in practical specifications of quantum ADTs (see Example 3.8 and Proposition 5.1).

It is worth examining carefully the differences between classical Equation (1) and quantum Equation (3). First, we notice that in terms τ_1 and τ_2, quantum variables $\overline{q_1}$ and $\overline{q_2}$ are already initialised, but $\overline{r}$ are not. Thus, Equation (3) means that the equality between τ_1 and τ_2 is valid for all possible input states of $\overline{r}$. Second, an input state $|\varphi\rangle_{\overline{r}}$ of $\overline{r} = r_1, \dots, r_n$ may be entangled. Third, the evaluation of τ_1 (resp. τ_2) gives an output state $|\psi_1\rangle$ (resp. $|\psi_2\rangle$) of multiple quantum variables, but one may only be interested in a subset $\overline{p}$ of them. Thus, the variables in $\overline{p}$ are designated as the principal variables. Consequently, to define the validity of equations in a quantum algebra, we need the notion of a reduced density operator. For a quantum state $|\psi\rangle$ in Hilbert space $\mathcal{H}_{\overline{p}} \otimes \mathcal{H}_{\overline{p'}}$, the restriction of $|\psi\rangle$ onto $\overline{p}$ is defined as the reduced density operator

$$|\psi\rangle \downarrow \overline{p} = \mathrm{tr}_{\overline{p'}}(|\psi\rangle\langle\psi|)$$

where $\mathrm{tr}_{\overline{p'}}$ stands for the partial trace over subsystem $\overline{p'}$ (see Section 2.4.3 of [29] for its precise definition). Intuitively, $|\psi\rangle \downarrow \overline{p}$ describes the state of subsystem $\overline{p}$ when the whole system $\overline{p} \uplus \overline{p'}$ is in state $|\psi\rangle$. With this preparation, we are able to introduce:

Definition 3.6 (Validity). The equation e given in Equation (3) is valid in algebra $\mathcal{A}$, written $\mathcal{A} \models e$, if for any quantum state $|\varphi\rangle \in \mathcal{H}_{\overline{r}}$, we have:

$$eval_{|\varphi\rangle}(\tau_1) \downarrow \overline{p} = eval_{|\varphi\rangle}(\tau_2) \downarrow \overline{p}.$$

3.2 Specification of Quantum Data Types

As one may expect, the algebraic framework developed in the previous subsection can be used to specify ADTs in quantum computing. The following definition is a straightforward generalisation of Definition 2.6.

Definition 3.7 (Quantum Specification).

(1) A quantum specification is a quadruple $QSpec = (S_q, \mathcal{K}, \mathcal{U}, E)$, where $QSig = (S_q, \mathcal{K}, \mathcal{U})$ is a quantum signature, and E is a set of equations of the form in Equation (3).
(2) A $QSpec$-algebra is a quantum algebra $\mathcal{A}$ of signature $QSig$ such that $\mathcal{A} \models e$ for every $e \in E$.

For a better understanding of quantum ADTs, let us first see the following simple example:

Example 3.8 (Abstract data type of qubits). Consider the following quantum specification:

$$
\begin{aligned}
\mathtt{QBIT} = \textbf{sorts}:\ & \mathtt{qbit} \\
\textbf{states}:\ & |0\rangle, |1\rangle, |+\rangle, |-\rangle : \mathtt{qbit} \\
& |\beta_{00}\rangle, |\beta_{01}\rangle, |\beta_{10}\rangle, |\beta_{11}\rangle : \mathtt{qbit}, \mathtt{qbit} \\
\textbf{gates}:\ & \mathit{NOT}, H : \mathtt{qbit} \\
& \mathit{CNOT} : \mathtt{qbit}, \mathtt{qbit} \\
\textbf{equations}:\ & q, q_1, q_2 : \mathtt{qbit} \\
& \mathit{NOT}[q]\,|0\rangle_q = |1\rangle_q, \quad \mathit{NOT}[q]\,|1\rangle_q = |0\rangle_q \\
& H[q]\,|0\rangle_q = |+\rangle_q, \quad H[q]\,|1\rangle_q = |-\rangle_q \\
& \mathit{CNOT}[q_1, q_2]\,|+\rangle_{q_1}|0\rangle_{q_2} = |\beta_{00}\rangle_{q_1 q_2}, \quad \mathit{CNOT}[q_1, q_2]\,|+\rangle_{q_1}|1\rangle_{q_2} = |\beta_{01}\rangle_{q_1 q_2} \\
& \mathit{CNOT}[q_1, q_2]\,|-\rangle_{q_1}|0\rangle_{q_2} = |\beta_{10}\rangle_{q_1 q_2}, \quad \mathit{CNOT}[q_1, q_2]\,|-\rangle_{q_1}|1\rangle_{q_2} = |\beta_{11}\rangle_{q_1 q_2}
\end{aligned}
$$

We can define a QBIT-algebra $\mathcal{A}$ as follows:

- Let $\mathcal{H}^{\mathcal{A}}_{\mathtt{qbit}} = \mathcal{H}_2$ be the 2-dimensional Hilbert space.
- For quantum state symbols, define $|b\rangle^{\mathcal{A}} = |b\rangle$ for $b \in \{0, 1\}$, $|\pm\rangle^{\mathcal{A}} = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$, and $|\beta_{ab}\rangle^{\mathcal{A}} = \frac{1}{\sqrt{2}}(|0\rangle\,|b\rangle + (-1)^a\,|1\rangle\,|1-b\rangle)$ for $a, b \in \{0, 1\}$.
- For quantum gate symbols, define

$$\mathit{NOT}^{\mathcal{A}} = X, \qquad H^{\mathcal{A}} = H, \qquad \mathit{CNOT}^{\mathcal{A}} = \mathrm{CNOT},$$

  where X, H, and CNOT are the Pauli X, Hadamard, and controlled-NOT gates, respectively.

It is easy to check that $\mathcal{A}$ satisfies all equations in QBIT, and hence $\mathcal{A}$ is a QBIT-algebra.

3.3 A Simple Application Example: Programming Quantum Query Algorithms

Quantum query algorithms. To further understand the role of ADTs in quantum computing, let us consider a large class of quantum algorithms, namely quantum query algorithms. Let $\overline{q_w}$ and $\overline{q_o}$ be two disjoint strings of qubits of lengths l_w and l_o, called the work qubits and output qubits, respectively. Suppose we are given a quantum oracle O, which is a black-box unitary operator. Then a quantum query algorithm [2] with queries to O is usually described as a quantum circuit:

$$C = U_T O U_{T-1} \cdots U_2 O U_1 O U_0$$

where U_t $(t = 0, 1, \ldots, T)$ are quantum gates independent of O. The circuit C is initialised in state $|0\rangle_{\overline{q_w}}\,|0\rangle_{\overline{q_o}}$. Then the algorithm outputs x (a bit string) with probability

$$\mathrm{Prob}(x) = \left\| \langle x|_{\overline{q_o}}\, C\, |0\rangle_{\overline{q_w}}\,|0\rangle_{\overline{q_o}} \right\|^2.$$

Oracles as ADTs. Oracle synthesis functionality has already been integrated into quantum programming platforms like Qiskit [19] and Q# [35]. Ideally, a programmer does not need to know the implementation details of the oracle O when programming the quantum query algorithm. Instead, it can be viewed through the lens of ADTs. Consider a quantum specification $(S_q, \mathcal{K}, \mathcal{U}, E)$, where $\mathtt{qbit} \in S_q$, $O \in \mathcal{U}$, and $\mathit{sort}(O) = \underbrace{\mathtt{qbit}, \ldots, \mathtt{qbit}}_{l_w + l_o}$. The equations in E may specify properties that the oracle O should satisfy without fixing its implementation details. In this way, the quantum query algorithm can be programmed as:

$$\begin{aligned}P \equiv\ & \overline{q_w} := |0\rangle^{l_w}; \overline{q_o} := |0\rangle^{l_o};\\ & \texttt{for } i \texttt{ from } 0 \texttt{ to } T-1 \texttt{ do } U_i[\overline{q}];\\ & \qquad\qquad\qquad\qquad\qquad O[\overline{q}]; \quad \text{(calling the oracle operation)}\\ & U_T[\overline{q}];\\ & x := M[\overline{q_o}]\end{aligned}$$

where $\overline{q} = \overline{q_w} \cup \overline{q_o}$ and M is the measurement in the computational basis of $\overline{q_o}$.

4 Quantisation of Data Types

The abstract notion of quantum ADTs was introduced in Section 3. In practical applications, a quantum ADT does not stand alone; rather, it often arises as an extension of a corresponding classical ADT. An application scenario is that, when developing a quantum program, the programmer realises that they need to apply a function, or more generally an ADT, that does not comply with certain constraints (e.g., non-unitarity) imposed by quantum mechanics. In such cases, the function or ADT must be quantised, and what the programmer actually invokes in the program is its quantisation. A typical example is a quantum oracle (see Section 3.3), which is often lifted from a black-box classical function. So, in this section, we consider how a classical data type defined in Section 2 can be extended to a quantum data type introduced in Section 3. We call this extension process *quantisation.*

In this section, we make the following conventions. Given a classical signature, for each sort s, we assume a designated constant symbol $0_s :\longrightarrow s$. Given a quantum signature, for any sort s, we assume a designated quantum state symbol $|0_s\rangle : s$. For a string $\overline{s} = s_1, \ldots, s_n$ of sorts, the quantum state $|\overline{0}_{\overline{s}}\rangle^{\mathcal{A}}$ in a given algebra $\mathcal{A}$ is then $|0_{s_1}\rangle^{\mathcal{A}} \otimes \ldots \otimes |0_{s_n}\rangle^{\mathcal{A}}$. When the context is clear, we omit the subscript s and use $|\overline{0}\rangle$ to abbreviate $|0\rangle \ldots |0\rangle$. These designated symbols are implicitly assumed to be included in the signatures discussed below. Given an algebra A of a signature, for each sort s, we assume that A_s is finite and there is a designated bijection

$$k_s^A : A_s \to \mathbb{Z}_{|A_s|}, \qquad k_s^A(0_s^A) = 0 \tag{4}$$

where for a positive integer m, $\mathbb{Z}_m = \{0, 1, ..., m-1\}$. Intuitively, bijection $k_s(a)$ gives the default integer label for every element $a \in A_s$; that is, we use k_s to enumerate all elements of A_s.

Let us first consider how a classical signature can be quantised to a quantum signature.

Definition 4.1 (Quantisation of signature). Let $Sig = (S, OP)$ be a signature and $QSig = (S_q, \mathcal{K}, \mathcal{U})$ a quantum signature. A quantisation from Sig to $QSig$ is a mapping η such that

(1) each sort $s \in S$ is mapped to $s_q = \eta(s)$;
(2) each constant symbol $c :\longrightarrow s$ is mapped to a quantum state symbol $|\psi_c\rangle = \eta(c)$ of sort s_q. We additionally require $\eta(0_s) = |0_{s_q}\rangle$; and
(3) each nonconstant operation symbol $O : s_1, \ldots, s_n \to s$ is mapped to a triple of quantum gate symbols $(U_O, pre_O, post_O) = \eta(O)$, where $sort(U_O) = sort(pre_O) = sort(post_O) \in S_q^+$ is a string of sorts in one of the following forms:
   - (in-place) $s_{1q}, \ldots, s_{nq}, \overline{s_a}$ for some $\overline{s_a} \in S_q^*$, with $s_q = s_{iq}$ for some i;
   - (out-of-place) $s_{1q}, \ldots, s_{nq}, s_q, \overline{s_a}$ for some $\overline{s_a} \in S_q^*$.
(4) In the above, the mappings $s \mapsto s_q$, $c \mapsto |\psi_c\rangle$, and $O \mapsto U_O$ are injective.

We need to carefully explain the above condition (3). For a nonconstant operation O, it is natural to expect its quantisation lifts its behaviour on classical data to the quantum setting. However, there are multiple ways to represent classical data as quantum data, e.g., encoding into computational-basis quantum states, encoding into relative phases, encoding into quantum amplitudes. Concrete examples can be found in Sections 4.1, 4.2, 5 and 6. Therefore, the quantum gate symbols pre_O and $post_O$ in $\eta(O)$ serve for the representation of classical data in the quantum world, while U_O serves for the lifted O in the quantum world. In particular, the quantum gate symbol pre_O (resp. $post_O$) corresponds to how classical data is preprocessed into (resp. postprocessed from) quantum data. Then, U_O corresponds to the quantisation of O that has quantum input and output. Their concrete interpretations in an algebra is given later. The injectivity of $O \mapsto U_O$ in condition (4) means U_O is unique for each O, but pre_O and $post_O$ can be the same for different O.

For convenience, if an operation symbol O is quantised in the in-place form, we assume the index i for $s_q = s_{iq}$ is implicitly fixed and used consistently in the other quantisation notions. The string $\overline{s_a}$ of sorts represents ancilla sorts needed for the quantised operation.

Next, we consider how an algebra can be quantised along the quantisation of the signature.

Definition 4.2 (Quantisation of algebra). Let η be a quantisation from *Sig* to *QSig*. Then a quantum algebra $\mathcal{A}$ of signature *QSig* is a quantisation of an algebra A of signature *Sig* along η if:

(1) for each sort $s \in S$, $\{|a\rangle \mid a \in A_s\}$ forms an orthonormal basis of $\mathcal{H}^{\mathcal{A}}_{s_q}$;
(2) for each constant symbol $c :\rightarrow s$, $|\psi_c\rangle^{\mathcal{A}} = |c^A\rangle$. In particular, $|0_{s_q}\rangle^{\mathcal{A}} = |0^A_s\rangle$; and
(3) for each nonconstant operation symbol $O : s_1, \ldots, s_n \rightarrow s$, and for all $a_1 \in A_{s_1}, \ldots, a_n \in A_{s_n}$, we have:
$$\left\| \langle O^A(a_1, \ldots, a_n)| \, post^{\mathcal{A}}_O U^{\mathcal{A}}_O pre^{\mathcal{A}}_O \Big(|a_1\rangle \cdots |a_n\rangle \, |\bar{0}\rangle \Big) \right\|^2 = 1. \tag{5}$$
Here, the subsystems involved in quantum states and gates are implicitly assumed to be ordered according to $sort(U_O)$. In particular,
   - (in-place) $|\bar{0}\rangle$ denotes $|\bar{0}_{\overline{s_a}}\rangle^{\mathcal{A}}$, and $|O^A(a_1, \ldots, a_n)\rangle$ is a state in $\mathcal{H}^{\mathcal{A}}_{s_{iq}}$, where i is the fixed index in Definition 4.1; and
   - (out-of-place) $|\bar{0}\rangle$ denotes $|\bar{0}_{s_q,\overline{s_a}}\rangle^{\mathcal{A}}$, and $|O^A(a_1, \ldots, a_n)\rangle$ is a state in $\mathcal{H}^{\mathcal{A}}_{s_q}$.

Intuitively, in Equation (5), $pre^{\mathcal{A}}_O$ converts a computational-basis input (the default quantum representation of classical data) into a quantum input for $U^{\mathcal{A}}_O$, and $post^{\mathcal{A}}_O$ converts the quantum output of $U^{\mathcal{A}}_O$ into a computational-basis output. The condition in Equation (5) says that, under the preprocessing $pre^{\mathcal{A}}_O$ and postprocessing $post^{\mathcal{A}}_O$, the behaviour of $U^{\mathcal{A}}_O$ agrees with O^A on all classical inputs (with probability 1), so $U^{\mathcal{A}}_O$ quantises O^A. In Sections 4.1, 4.2, 5 and 6, we will present illustrative examples, where the utility of the preprocessing pre_O and postprocessing $post_O$ becomes more concrete.

It remains to consider how a classical specification can be quantised. To this end, we need to introduce a quantisation of classical terms.

Definition 4.3 (Quantisation of terms). Let η be a quantisation from *Sig* to *QSig*. Suppose *QSig* contains a quantum gate symbol $SUM_s : s_q, s_q$ for each sort s in *Sig*, where $s_q = \eta(s)$. Let t be a term in *Sig*. For each variable $x : s$ occurring in t, designate a distinct quantum variable $q_x : s_q$.

The quantisation $\eta(t)$ is defined recursively as follows:

(1) If $t = c :\rightarrow s$ is a constant symbol, choose a fresh variable $p : s_q$ as the principal variable and define $\eta(t) = |\psi_c\rangle_p$, where $|\psi_c\rangle = \eta(c)$;
(2) If $t = x$ is a variable of sort s, choose a fresh variable $q : s_q$ as the principal variable and define $\eta(t) = SUM_s[q_x, q] \, |0\rangle_q$; and
(3) Suppose $t = O(t_1, \ldots, t_n)$ is a term of sort s and $\eta(t_i) = C_i \, |\psi_i\rangle$ for $i = 1, ..., n$, where each $\eta(t_j)$ has principal variable p_j of sort s_{jq}. Let $\overline{p} = p_1, \ldots, p_n$. By renaming variables if necessary,

assume that, apart from the designated q_x, all other quantum variables in $\eta(t_1), \ldots, \eta(t_n)$ are distinct.

We define

$$\eta(t) = C_1; \ldots; C_n; pre_O[\overline{p}, \overline{r}]; U_O[\overline{p}, \overline{r}]; post_O[\overline{p}, \overline{r}] \, |\psi_1\rangle \ldots |\psi_n\rangle \, |\overline{0}\rangle_{\overline{r}}.$$

Here $\overline{r}$ is a sequence of fresh variables such that

- (in-place) $sort(\overline{r}) = \overline{s_a}$ and p_i is chosen to be the principal variable, where i is the fixed index in Definition 4.1;
- (out-of-place) $sort(\overline{r}) = s_q, \overline{s_a}$ and $\overline{r} = r_0, \overline{r_a}$, where r_0 is chosen to be the principal variable.

In Definition 4.3, note that the same q_x is used for every occurrence of variable x in t. In clause (2) of this definition, the design $\eta(t) = SUM_s[q_x, q] \, |0\rangle_q$ is subtle and needs to be carefully explained. Intuitively, if the gate symbol SUM is interpreted properly in an algebra, $SUM_s[q_x, q] \, |0\rangle_q$ copies the computational-basis state of quantum variable q_x to q, which is initialised in state $|0\rangle_q$. It has two purposes. First, copying in the computational basis reversibly handles the multiple occurrences of variable x in term t, where q_x is used to store the input of x and each fresh q corresponds to a single occurrence of x. Second, since q_x are not chosen to be principal variables, tracing out q_x at the end "dephases" each corresponding q in the computational basis. This is important for the correctness of Theorem 4.5, when we transition from a classical equation with a universal quantifier $(\forall \overline{x})$ over classical inputs to a quantum equation with quantifier $(\forall \overline{q})$ over quantum inputs.

With quantisations of classical terms, we can further define quantisations of equations in a classical specification.

Definition 4.4 (Quantisation of equations). Let η be a quantisation from Sig to $QSig$, and let $e = (\forall \overline{x} : \overline{s})(t_1 = t_2)$ be a classical equation, where $\overline{x} = x_1, \ldots, x_n$ is a string of all variables occurring in t_1 or t_2. We assume that the designated variables $\overline{q} = q_{x_1}, \ldots, q_{x_n}$ (see Definition 4.3) in t_1 and t_2 are aligned, and $\eta(t_1)$ and $\eta(t_2)$ have the same principal variable, say p. Then, the quantisation $\eta(e)$ of e along η is defined as follows:

$$\eta(e) = (\forall \overline{q} : \overline{s_q})(\eta(t_1) = \eta(t_2)) \Downarrow p,$$

where $\overline{s_q}$ is the quantisation of $\overline{s}$.

With the above preparation, we are able to present the main result of this section, which describes a way of deriving a specification of a quantised ADT by lifting from the classical specification. To describe this lifting, for $a, b \in A_s$, we define

$$a \boxplus_s^A b = (k_s^A)^{-1}\Big(k_s^A(a) + k_s^A(b) \bmod |A_s|\Big) \tag{6}$$

where k_s^A is given in equation (4). Intuitively, the operator $\boxplus_s^A$ naturally extends the binary XOR $\oplus$ operator up to the bijection k_s^A. In particular, in Equation (6), the inputs a, b are mapped to the integer domain $\mathbb{Z}_{|A_s|}$, added in $\mathbb{Z}_{|A_s|}$, and then mapped back to the domain A_s. It is easy to verify that $a \boxplus_s^A 0_s^A = 0_s^A \boxplus_s^A a = a$ for any a. When the algebra A is clear, we simply write k_s and $\boxplus_s$ for k_s^A and $\boxplus_s^A$, respectively.

Theorem 4.5 (Lifting of specification). *Let $Spec = (Sig, E)$ be a classical specification, and let η be a quantisation from Sig to QSig. Suppose QSig contains a quantum gate symbol $SUM_s : s_q, s_q$ for each sort s in Sig, where $s_q = \eta(s)$. We define $QSpec = (QSig, \eta(E))$, where $\eta(E) = \{\eta(e) \mid$ equation $e \in E\}$. If*

1. *A is an algebra of Spec; and*
2. *$\mathcal{A}$ is a quantisation of A along η and $SUM_s^{\mathcal{A}} : |a\rangle \, |b\rangle \mapsto |a\rangle \, |a \boxplus_s b\rangle$ for any sort s and any $a, b \in A_s$,*

then $\mathcal{A}$ is a quantum algebra of QSpec.

Here, corresponding to the discussion below Definition 4.3, the unitary operator $SUM_s^{\mathcal{A}}$: $|a\rangle\,|b\rangle \mapsto |a\rangle\,|a \boxplus_s b\rangle$ is a canonical way to realise the copying in the computational basis when $b = 0$. The proof of Theorem 4.5 is deferred to Appendix A.

The above definitions of quantisation are general. In the following subsections, we examine two specific quantisation methods that appear commonly in quantum algorithm design.

4.1 Bit Oracle Quantisation

The first quantisation method is bit oracle quantisation (BO-quantisation for short). The BO-quantisation of a classical algebra is defined as follows.

Definition 4.6 (BO-quantisation). Let η be a quantisation from *Sig* to *QSig* and A be a *Sig*-algebra. Then, a quantisation $\mathcal{A}$ of A along η is a BO-quantisation if for each nonconstant operation symbol $O : s_1, \ldots, s_n \to s$:

(1) U_O is out-of-place with $\overline{s_a} = \emptyset$;
(2) $pre_O^{\mathcal{A}} = post_O^{\mathcal{A}} = \mathbb{1}_{sort(U_O)}$; and
(3) for any $a_1 \in A_{s_1}, \ldots, a_n \in A_{s_n}, a \in A_s$,

$$U_O^{\mathcal{A}}(|a_1\rangle \cdots |a_n\rangle\,|a\rangle) = |a_1\rangle \cdots |a_n\rangle\,|a \boxplus_s O^A(a_1, \ldots, a_n)\rangle\,. \tag{7}$$

It is easy to check that $U_O^{\mathcal{A}}$ defined in equation (7) indeed satisfies the condition (5). When *QSig* includes the gate symbol SUM_s for each sort s, the BO-quantisation of terms and equations can be naturally induced by Definition 4.6, according to Definitions 4.1 and 4.4.

Definition 4.6 naturally yields the common bit oracle in quantum algorithms as follows, where the designated $k_{\texttt{bit}}$ is the identity and $\boxplus_{\texttt{bit}}$ reduces to XOR.

Example 4.7. Consider the following

- Signature $Sig = (S, OP)$ with $S = \{\texttt{bit}\}$ and $OP = \{0_{\texttt{bit}}, f\}$, where $0_{\texttt{bit}} :\longrightarrow \texttt{bit}$ and $f : \underbrace{\texttt{bit}, \ldots, \texttt{bit}}_{n} \to \texttt{bit}$.
- Algebra A of signature *Sig* with $A_{\texttt{bit}} = \{0, 1\}$ and $0_{\texttt{bit}}^A = 0$, where $f^A : \{0,1\}^n \longrightarrow \{0, 1\}$ is a n-ary Boolean function.

Then the following gives a BO-quantisation of A:

- Signature $QSig = (S_q, \mathcal{K}, \mathcal{U})$ with $S_q = \{\texttt{qbit}\}$, $\mathcal{K} = \left\{|0_{\texttt{qbit}}\rangle\right\}$, and $\mathcal{U} = \left\{U_f, pre_f, post_f\right\}$, where $|0_{\texttt{qbit}}\rangle : \texttt{qbit}$ and $U_f, pre_f, post_f : \underbrace{\texttt{qbit}, \ldots, \texttt{qbit}}_{n+1}$.
- Algebra $\mathcal{A}$ of signature *QSig*: $\mathcal{H}_{\texttt{qbit}}^{\mathcal{A}} = \mathbb{C}^2$, $|0_{\texttt{qbit}}\rangle^{\mathcal{A}} = |0\rangle$, $pre_f^{\mathcal{A}} = post_f^{\mathcal{A}} = \mathbb{1}$, and

$$U_f^{\mathcal{A}}\,|\overline{x}, b\rangle = |\overline{x}, b \oplus f^A(\overline{x})\rangle \tag{8}$$

for any $\overline{x} \in \{0,1\}^n$ and $b \in \{0, 1\}$.

In this example, the unitary operator $U_f^{\mathcal{A}}$ is often known as the bit oracle or standard oracle used by a quantum algorithm to query a black-box function f. The bit oracle has been employed in the design of many quantum algorithms, e.g., the Deutsch-Jozsa algorithm [7] and the modular exponentiation subroutine of Shor's algorithm [34]. It should be noted that on the RHS of Equation (8), the first n qubits are used to record the input $\overline{x}$, and the last qubit is used to store the computational outcome $f(\overline{x})$. In this way, U_f is made invertible and thus a unitary operator.

4.2 Phase Oracle Quantisation

The second quantisation method is phase oracle quantisation (PO-quantisation for short). The PO-quantisation of a classical algebra is defined as follows.

Definition 4.8 (PO-quantisation). Let η be a quantisation from *Sig* to *QSig* and A be a *Sig*-algebra. Then, a quantisation $\mathcal{A}$ of A along η is a PO-quantisation if for each nonconstant operation symbol $O : s_1, \ldots, s_n \to s$:

(1) U_O is out-of-place with $\overline{s_a} = \emptyset$.
(2) $pre_O^{\mathcal{A}} = \mathbb{1}_{s_{1q},\ldots,s_{nq}} \otimes QFT_{|A_s|}$ and $post_O^{\mathcal{A}} = \mathbb{1}_{s_{1q},\ldots,s_{nq}} \otimes QFT_{|A_s|}^{\dagger}$, where $QFT_{|A_s|}$ is the quantum Fourier transform on $\mathbb{Z}_{|A_s|}$ defined by

$$QFT_{|A_s|} |a\rangle = \frac{1}{\sqrt{|A_s|}} \sum_{b \in A_s} \omega^{k_s(a) \cdot k_s(b)} |b\rangle$$

for any $a \in A_s$, where $\omega = e^{2\pi i/|A_s|}$; and
(3) for any $a_1 \in A_{s_1}, \ldots, a_n \in A_{s_n}, a \in A_s$,

$$U_O^{\mathcal{A}} |a_1, \ldots, a_n\rangle |a\rangle = \omega^{k_s(a) \cdot k_s(O^A(a_1,\ldots,a_n))} |a_1, \ldots, a_n\rangle |a\rangle . \tag{9}$$

It is also easy to check that $U_O^{\mathcal{A}}$ defined in equation (9) satisfies the condition (5). When *QSig* includes the gate symbol SUM_s for each sort s, the PO-quantisation of terms and equations can be naturally induced by Definition 4.8, according to Definitions 4.1 and 4.4.

Definition 4.8 naturally yields the common phase oracle (or more accurately, the controlled version of the common phase oracle) in quantum algorithms, where the designated $k_{\texttt{bit}}$ is the identity, $\omega = -1$ and $QFT_2 = H$ is the Hadamard operator.

Example 4.9. Consider the same signature *Sig* and algebra A as in Example 4.7. Then, the following gives a PO-quantisation of A:

- Signature $QSig = (S_q, \mathcal{K}, \mathcal{U})$: $S_q = \{\texttt{qbit}\}$, $\mathcal{K} = \left\{|0_{\texttt{qbit}}\rangle\right\}$, and $\mathcal{U} = \left\{U_f, pre_f, post_f\right\}$, where $|0_{\texttt{qbit}}\rangle : \texttt{qbit}$ and $U_f, pre_f, post_f : \underbrace{\texttt{qbit}, \ldots, \texttt{qbit}}_{n+1}$.
- Algebra $\mathcal{A}$ of signature *QSig*: $\mathcal{H}^{\mathcal{A}}_{\texttt{qbit}} = \mathbb{C}^2$, $|0_{\texttt{qbit}}\rangle^{\mathcal{A}} = |0\rangle$, $pre_f^{\mathcal{A}} = post_f^{\mathcal{A}} = \mathbb{1}_n \otimes H$, and

$$U_f^{\mathcal{A}} |\overline{x}, b\rangle = (-1)^{b \cdot f^A(\overline{x})} |\overline{x}, b\rangle$$

for any $\overline{x} \in \{0, 1\}^n$ and $b \in \{0, 1\}$.

In this example, U_f is often known as the (controlled) phase oracle for a quantum algorithm to query a black-box function f. The phase oracle has been applied in Grover's search [15], amplitude amplification [4], and several other quantum algorithms.

4.3 Phase Kickback Trick

In quantum algorithms, the phase kickback trick provides a useful connection between a bit oracle and a phase oracle, and now we can view it through the lens of ADTs. For example, in the Deutsch-Jozsa algorithm that decides whether a given Boolean function $f : \{0, 1\}^n \to \{0, 1\}$ is constant or balanced, the bit oracle U_f is defined by

$$U_f |\overline{x}, y\rangle = |\overline{x}, y \oplus f(\overline{x})\rangle$$

for any $\overline{x} \in \{0,1\}^n$ and $y \in \{0,1\}$. A key idea in this algorithm is that the target register is initialised in the superposition $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. We observe:

$$U_f\,|\overline{x}, -\rangle = |\overline{x}\rangle \otimes (-1)^{f(\overline{x})}\,|-\rangle = (-1)^{f(\overline{x})}\,|\overline{x}\rangle \otimes |-\rangle\,. \tag{10}$$

In the final expression above, the phase $(-1)^{f(\overline{x})}$ appears in front of the quantum state. On the other hand, if the target register is initialised in $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, then

$$U_f\,|\overline{x}, +\rangle = |\overline{x}\rangle \otimes |+\rangle\,. \tag{11}$$

Combining Equations (10) and (11) gives

$$(\mathbb{1}_n \otimes H)U_f(\mathbb{1}_n \otimes H)\,|\overline{x}, b\rangle = (-1)^{b\cdot f(\overline{x})}\,|\overline{x}, b\rangle\,. \tag{12}$$

More generally, the phase kickback implies a correspondence between BO-quantisation and PO-quantisation. Suppose η is a quantisation from *Sig* to *QSig*. Let $\mathcal{A}$ and $\mathcal{B}$ be BO- and PO-quantisations of the same algebra A along η, respectively. For each sort s in *Sig*, suppose that $\mathcal{A}$ and $\mathcal{B}$ have the same quantum domain

$$\mathcal{H}^{\mathcal{A}}_{s_q} = \mathcal{H}^{\mathcal{B}}_{s_q} = \operatorname{span}\{|a\rangle : a \in A_s\}.$$

Then, analogous to Equation (12), for any operation $O : s_1, \dots, s_n \to s$, we have:

$$U^{\mathcal{B}}_O = \left(\mathbb{1}_{s_{1q},\dots,s_{nq}} \otimes QFT_{|A_s|}\right)U^{\mathcal{A}}_O\left(\mathbb{1}_{s_{1q},\dots,s_{nq}} \otimes QFT^{\dagger}_{|A_s|}\right), \tag{13}$$

which is essentially equivalent to

$$post^{\mathcal{A}}_O U^{\mathcal{A}}_O pre^{\mathcal{A}}_O = post^{\mathcal{B}}_O U^{\mathcal{B}}_O pre^{\mathcal{B}}_O.$$

This means that BO-quantisation and PO-quantisation can be related by changing how classical data are encoded into quantum data through pre_O and $post_O$.

5 Illustrative Application I: Quantum Arrays

Arrays are a fundamental data type widely used in classical programming. To illustrate the framework developed in Sections 3 and 4, we show how quantum arrays can be defined via the quantisation of classical arrays. In particular, this approach offers a new perspective for understanding Quantum Random Access Memories (QRAMs) [10, 11].

5.1 Classical Arrays

For convenience of the reader, let us recall the basic theory of arrays. First, we describe the signature $Sig = (S, OP)$ of the ADT of arrays. Let $\mathcal{I}$ be a set of index sorts and $\mathcal{V}$ a set of value sorts, so that $S = \mathcal{I} \cup \mathcal{V}$. The value sorts and their dimensions are generated as follows:

(1) $\mathtt{val} \in \mathcal{V}$ is a primitive value sort, and $\dim(\mathtt{val}) = 0$;
(2) if $I \in \mathcal{I}$ and $\tau \in \mathcal{V}$, then $\mathtt{array}(I, \tau) \in \mathcal{V}$, and $\dim(\mathtt{array}(I, \tau)) = \dim(\tau) + 1$.

A canonical algebra A of *Sig* gives the following interpretation. For every $I \in \mathcal{I}$ and $\tau \in \mathcal{V}$, define

$$A_{\mathtt{array}(I,\tau)} = A_I \to A_\tau.$$

Consequently, if

$$\tau = \mathtt{array}(I_1, \mathtt{array}(I_2, \dots, \mathtt{array}(I_n, \mathtt{val}) \dots)),$$

then

$$A_\tau = A_{I_1} \to \left(A_{I_2} \to \cdots \to \left(A_{I_n} \to A_{\mathtt{val}}\right) \cdots\right).$$

Equivalently, $A_\tau = (A_{I_1} \times \cdots \times A_{I_n}) \to A_{\mathtt{val}}$.

For the operation symbols, let OP contain two classes of symbols:

$$\begin{aligned} &read_{I,\tau} : \mathtt{array}(I,\tau), I \to \tau \\ &write_{I,\tau} : \mathtt{array}(I,\tau), I, \tau \to \mathtt{array}(I,\tau) \end{aligned} \tag{14}$$

where $I \in \mathcal{I}$ and $\tau \in \mathcal{V}$. They are also called *selector* and *store* operations, respectively. For simplicity, the subscripts I, τ of $read_{I,\tau}$ and $write_{I,\tau}$ will often be omitted. In the canonical algebra A, they are interpreted as follows:

$$\begin{aligned} read^A_{I,\tau} &: (a, j) \mapsto a[j] \\ write^A_{I,\tau} &: (a, j, v) \mapsto a' \end{aligned} \tag{15}$$

for $a \in A_{\mathtt{array}(I,\tau)}$, $j \in A_I$, and $v \in A_\tau$, where $a'[i] = a[i]$ if $i \neq j$ and $a'[j] = v$. Intuitively, $read(a, j)$ denotes the value stored in array a at index j, and $write(a, j, v)$ denotes the array obtained by updating a with value v stored at index j.

We assume countably many variables and constants of each sort. The terms are constructed from variables and constants using operation symbols, and their sorts are defined in the usual way. A basic theory of arrays was first introduced by McCarthy [27], where the functions *read* and *write* are characterised by the following axioms. For every $I \in \mathcal{I}$ and $\tau \in \mathcal{V}$:

- **Axiom 1 (read-over-write)**:

$$\begin{aligned} \forall a : \mathtt{array}(I,\tau), i, j : I, v : \tau.(&i = j \to read(write(a, i, v), j) = v) \\ &(i \neq j \to read(write(a, i, v), j) = read(a, j)) \end{aligned}$$

- **Axiom 2 (extensionality)**:

$$\forall a, b : \mathtt{array}(I,\tau).(\forall i : I.read(a, i) = read(b, i)) \to a = b$$

It should be pointed out that, essentially, the above two axioms do not constitute an equational specification as in Definition 2.6. Instead, they are formulated in a first-order logical language. However, they are easy to understand for our purposes.

5.2 Quantum Arrays

Now let us examine quantum arrays through the lens of the quantisation of the classical ADT of arrays (see Section 4). Suppose $QSig = (S_q, \mathcal{K}, \mathcal{U})$ is the quantum signature obtained from a quantisation η of the classical signature for arrays. For $I \in \mathcal{I}$ and $\tau \in \mathcal{V}$, write

$$I_q = \eta(I), \qquad \tau_q = \eta(\tau), \qquad \mathtt{qarray}(I_q, \tau_q) = \eta(\mathtt{array}(I,\tau)).$$

In particular, write $\mathtt{qval} = \eta(\mathtt{val})$. Then

$$S_q = \mathcal{I}_q \cup \mathcal{V}_q, \qquad \mathcal{I}_q = \{\eta(I) : I \in \mathcal{I}\}, \qquad \mathcal{V}_q = \{\eta(\tau) : \tau \in \mathcal{V}\}.$$

Let A be the canonical algebra for the signature of classical arrays considered previously. We consider the following quantised algebra $\mathcal{A}$ of A along η. We define:

(1) $\mathcal{H}^{\mathcal{A}}_{\mathtt{qval}} = \mathrm{span}\{|v\rangle : v \in A_{\mathtt{val}}\}$;
(2) for every $I \in \mathcal{I}$, $\mathcal{H}^{\mathcal{A}}_{I_q} = \mathrm{span}\{|j\rangle : j \in A_I\}$; and
(3) for every $\tau \in \mathcal{V}$, $\mathcal{H}^{\mathcal{A}}_{\tau_q} = \mathrm{span}\{|t\rangle : t \in A_\tau\}$. Furthermore, for every $I \in \mathcal{I}$ and $\tau \in \mathcal{V}$,

$$\mathcal{H}^{\mathcal{A}}_{\mathtt{qarray}(I_q,\tau_q)} = \mathrm{span}\left\{|a\rangle : a \in A_{\mathtt{array}(I,\tau)}\right\} = \bigotimes_{j \in A_I} \mathcal{H}^{\mathcal{A}}_{\tau_q}, \tag{16}$$

where we denote $|a\rangle = \bigotimes_{j\in A_I} |a[j]\rangle$, and the indices j are ordered by keeping the designated $k_I(j)$ increasing (see Section 4). By expanding the RHS of Equation (16) recursively, if

$$\tau = \mathtt{array}(I_1, \mathtt{array}(I_2, \ldots, \mathtt{array}(I_n, \mathtt{val}) \ldots)),$$

then

$$\mathcal{H}^{\mathcal{A}}_{\tau_q} = \bigotimes_{(j_1,\ldots,j_n)\in A_{I_1}\times\cdots\times A_{I_n}} \mathcal{H}^{\mathcal{A}}_{\mathtt{qval}}.$$

Next, we explain how the *read* and *write* symbols in *OP* are quantised to gate symbols in $\mathcal{U}$. As in Section 4, for each $I \in \mathcal{I}$ and $\tau \in \mathcal{V}$, we assume that the designated 0 satisfies $0^A_{\mathtt{array}(I,\tau)}[j] = 0^A_\tau$ for any $j \in A_I$. Let k_I and k_τ be the designated bijections for A_I and A_τ, respectively. We recursively choose the designated bijection for the corresponding array sort to be

$$k_{\mathtt{array}(I,\tau)}(a) = \sum_{j\in A_I} |A_\tau|^{k_I(j)} \cdot k_\tau(a[j]). \tag{17}$$

For the concrete one-dimensional QRAM examples below, fix an index sort $\mathtt{idx} \in \mathcal{I}$ and write

$$\mathtt{qidx} = \eta(\mathtt{idx}), \qquad \mathtt{array} = \mathtt{array}(\mathtt{idx}, \mathtt{val}), \qquad \mathtt{qarray} = \mathtt{qarray}(\mathtt{qidx}, \mathtt{qval}).$$

Quantum read. The operation $read^A_{I,\tau}$ can be quantised in the same way as the BO-quantisation in Definition 4.6:

(1) U_{read}, pre_{read}, and $post_{read}$ have the sort $\mathtt{qarray}(I_q, \tau_q), I_q, \tau_q$.
(2) $pre^{\mathcal{A}}_{read} = post^{\mathcal{A}}_{read} = \mathbb{1}$.
(3) $U^{\mathcal{A}}_{read} |a\rangle |j\rangle |v\rangle = |a\rangle |j\rangle |v \boxplus_\tau a[j]\rangle$ for any $a \in A_{\mathtt{array}(I,\tau)}$, $j \in A_I$, and $v \in A_\tau$.

We simply use *qread* to denote the gate U_{read} for convenience.

What $qread^{\mathcal{A}}$ effectively implements is a QRAM read operation. Consider the case when $A_{\mathtt{idx}} = \mathbb{Z}_N$, $A_{\mathtt{val}} = \mathbb{Z}_M$, and their designated bijections are the identity maps. Then, an array a of sort $\mathtt{array}$ stores N classical values $a[0], a[1], \ldots, a[N-1]$ from $A_{\mathtt{val}}$ at addresses $0, 1, \ldots, N-1$. In classical computing, RAM (Random Access Memory) allows one to access a memory cell by providing an address. In QRAM (Quantum Random Access Memory), the key idea is that the address itself can be in a quantum superposition. This means one can query many memory locations simultaneously in superposition. A QRAM read implements the following transformation:

$$\sum_j \beta_j |a[0], \ldots, a[j], \ldots, a[N-1]\rangle |j\rangle |0\rangle \mapsto \sum_j \beta_j |a[0], \ldots, a[j], \ldots, a[N-1]\rangle |j\rangle |a[j]\rangle, \tag{18}$$

which is an instantiation of $qread^{\mathcal{A}}$. So, in one operation, we can "load" data from multiple addresses coherently.

Quantum write. The operation $write^A_{I,\tau}$ can be quantised in-place as follows:

- U_{write}, pre_{write}, and $post_{write}$ have the sort $\mathtt{qarray}(I_q, \tau_q), I_q, \tau_q$.
- $pre^{\mathcal{A}}_{write} = post^{\mathcal{A}}_{write} = \mathbb{1}$.
- $U^{\mathcal{A}}_{write} |a\rangle |j\rangle |v\rangle = |a'\rangle |j\rangle |a[j]\rangle$ for any $a \in A_{\mathtt{array}(I,\tau)}$, $j \in A_I$, and $v \in A_\tau$, where $a'[i] = a[i]$ if $i \neq j$ and $a'[j] = v$.

We simply use *qwrite* to denote the gate U_{write} for convenience.

What $qwrite^{\mathcal{A}}$ effectively implements is a QRAM swap operation. Again, consider the case when $A_{\mathtt{idx}} = \mathbb{Z}_N$, $A_{\mathtt{val}} = \mathbb{Z}_M$, and their designated bijections are the identity maps. Then, an array a of sort $\mathtt{array}$ stores N classical values $a[0], a[1], \ldots, a[N-1]$ from $A_{\mathtt{val}}$ at addresses $0, 1, \ldots, N-1$. A QRAM swap implements the following transformation:

$$\sum_j \beta_j |a[0], \ldots, a[j], \ldots, a[N-1]\rangle |j\rangle |v\rangle \mapsto \sum_j \beta_j |a[0], \ldots, v, \ldots, a[N-1]\rangle |j\rangle |a[j]\rangle . \quad (19)$$

Therefore, in one operation, *qwrite* can "write" data to multiple addresses coherently.

According to Theorem 4.5, by properly reexpressing **Axiom 1** and **Axiom 2** using equations we can derive specifications for the ADT of quantum arrays. Beyond that, quantum arrays actually satisfy extra properties. The following proposition shows that quantum array operations *qread* and *qwrite* satisfy an equation similar to the first part of the **Axiom 1** for classical array operations *read* and *write*. This property is specified as a quantum equation (see Definition 3.5). Suppose that for each sort s in *Sig*, $SUM_s : s_q, s_q$ is a gate symbol in *QSig* and $SUM_s^{\mathcal{A}} |a\rangle |b\rangle = |a\rangle |a \boxplus_s b\rangle$ for any $a, b \in A_s$.

Proposition 5.1. *The algebra of quantum arrays with gates qread and qwrite satisfies the following specification:*

$$\begin{aligned} &\forall q_a : \mathtt{qarray}, q_i : \mathtt{qidx}, q_v, q_u : \mathtt{qval} \\ &\qquad (qwrite[q_a, q_i, q_u]; qread[q_a, q_i, q_v]) = (SUM[q_u, q_v]; qwrite[q_a, q_i, q_u]) \end{aligned} \quad (20)$$

However, a natural quantum analogue of **Axiom 2** is not true. Let us consider the simplest case when $A_{\mathtt{idx}} = A_{\mathtt{val}} = \mathbb{Z}_2$. Even if we restrict attention to the principal variable, the equation

$$(qread[q_a, q_i, q_v] |\Phi\rangle = qread[q_a, q_i, q_v] |\Psi\rangle) \Downarrow q_v$$

does not imply $|\Phi\rangle = |\Psi\rangle$, because the state of a composite quantum system is not uniquely determined by the states of its subsystems, which is indeed one of the fundamental differences between classical and quantum systems.

6 Illustrative Application II: Quantum Error Correction

Quantum error correction is fundamental to fault-tolerant quantum computing [13, 22]. To further illustrate the framework developed in Sections 3 and 4, we show how a quantum error-correcting code can be described as a quantisation of a classical error-correcting code. In particular, we first consider the quantisation of a classical 3-bit repetition code and then show how its 9-bit extension gives rise to the Shor code.

6.1 Classical Repetition Code

We first describe a classical repetition code as an ADT. Consider the signature $Sig = (S, OP)$, where $S = \{\mathtt{lbit}, \mathtt{pbit}, \mathtt{idx}, \mathtt{array}\}$, $\mathtt{lbit}$ and $\mathtt{pbit}$ denote sorts of logical bits and physical bits, respectively. Following Section 5.1, $\mathtt{idx}$ is an index sort, and $\mathtt{array} = \mathtt{array}(\mathtt{idx}, \mathtt{pbit})$ is the sort for an array of physical bits. For simplicity, consider the canonical algebra A of *Sig* for the 3-bit repetition code. It gives:

$$A_{\mathtt{lbit}} = A_{\mathtt{pbit}} = \{0, 1\}, \qquad A_{\mathtt{idx}} = \{0, 1, 2\}, \qquad A_{\mathtt{array}} = A_{\mathtt{idx}} \to A_{\mathtt{pbit}}.$$

Thus, $A_{\mathtt{array}}$ consists of arrays of 3 physical bits indexed by $\{0, 1, 2\}$. For the operation symbols, let *OP* contain the following symbols:

$$\begin{aligned} enc &: \mathtt{lbit} \to \mathtt{array}, \\ flip &: \mathtt{array}, \mathtt{idx} \to \mathtt{array}, \\ corr &: \mathtt{array} \to \mathtt{array}. \end{aligned}$$

For $a = a[0], a[1], a[2] \in A_{\texttt{array}}$, define

$$maj(a) = \left\lfloor \frac{\sum_{i=0}^{2} a[i]}{2} \right\rfloor$$

to be the majority of $a[0], a[1], a[2]$.

In the canonical algebra A, the operations are interpreted as

$$\begin{aligned} enc^A &: x \mapsto x, x, x \\ flip^A &: (a, i) \mapsto a' \\ corr^A &: a \mapsto maj(a), maj(a), maj(a), \end{aligned}$$

where $a'[j] = a[j]$ if $j \neq i$ and $a'[i] = a[i] \oplus 1$. Intuitively, $enc(x)$ is the code word that encodes x by repeating it three times, $flip(a, i)$ flips the physical bit $a[i]$ in the array a, and $corr(a)$ produces a code word by replacing every entry of a with $maj(a)$. Then, the algebra A satisfies the following equations:

$$\begin{aligned} \forall x : \texttt{lbit}. &\quad corr(enc(x)) = enc(x), \\ \forall x : \texttt{lbit},\ i : \texttt{idx}. &\quad corr(flip(enc(x), i)) = enc(x). \end{aligned} \tag{21}$$

The first equation states that *corr* leaves a code word unchanged. The second equation states that it recovers the code word after a bit flip on any physical bit.

6.2 Quantum Repetition Code

Now let us show the 3-qubit repetition code as a quantisation of the classical 3-bit repetition code. Suppose $QSig = (S_q, \mathcal{K}, \mathcal{U})$ is the quantum signature obtained from a quantisation η of the classical signature *Sig* in Section 6.1. Let

$$\texttt{lqbit} = \eta(\texttt{lbit}), \qquad \texttt{pqbit} = \eta(\texttt{pbit}), \qquad \texttt{qidx} = \eta(\texttt{idx}), \qquad \texttt{qarray} = \eta(\texttt{array}).$$

In addition, let qsyn be an ancilla sort for syndromes. Then

$$S_q = \{\texttt{lqbit}, \texttt{pqbit}, \texttt{qidx}, \texttt{qarray}, \texttt{qsyn}\}.$$

Consider a quantum algebra $\mathcal{A}$ with

$$\begin{aligned} \mathcal{H}^{\mathcal{A}}_{\texttt{lqbit}} = \mathcal{H}^{\mathcal{A}}_{\texttt{pqbit}} &= \mathrm{span}\{|0\rangle, |1\rangle\}, & \mathcal{H}^{\mathcal{A}}_{\texttt{qidx}} &= \mathrm{span}\{|0\rangle, |1\rangle, |2\rangle\}, \\ \mathcal{H}^{\mathcal{A}}_{\texttt{qarray}} &= \bigotimes_{i=0}^{2} \mathcal{H}^{\mathcal{A}}_{\texttt{pqbit}}, & \mathcal{H}^{\mathcal{A}}_{\texttt{qsyn}} &= \mathrm{span}\{|0\rangle, |1\rangle, |2\rangle, |\bot\rangle\}. \end{aligned}$$

Here, $|\bot\rangle$ is the designated $|0_{\texttt{qsyn}}\rangle$, which is a computational-basis state orthogonal to $|0\rangle, |1\rangle, |2\rangle$. As in Section 5, we denote $|a\rangle = |a[0], a[1], a[2]\rangle$. In particular, the designated $|0_{\texttt{qarray}}\rangle = |0, 0, 0\rangle$. We choose the designated functions $k_{\texttt{lbit}}$, $k_{\texttt{pbit}}$, and $k_{\texttt{idx}}$ to be identity functions, and define $k_{\texttt{array}}$ as in Equation (17).

Next, we explain how the operation symbols *enc*, *flip*, and *corr* are quantised to gate symbols in $\mathcal{U}$.

Quantum encoding. The operation enc^A can be quantised out-of-place as follows:

(1) U_{enc}, pre_{enc}, and $post_{enc}$ have sort $\texttt{lqbit}, \texttt{qarray}$.
(2) $pre^{\mathcal{A}}_{enc} = post^{\mathcal{A}}_{enc} = \mathbb{1}$.
(3) $U^{\mathcal{A}}_{enc} |x\rangle |0, 0, 0\rangle = |0\rangle |x, x, x\rangle$ for any $x \in A_{\texttt{lbit}}$.

We simply use *qenc* to denote the gate U_{enc} for convenience.

Compared with the classical *enc* in Section 6.1, the gate *qenc* encodes $|x\rangle$ as $|x, x, x\rangle$ in the computational basis and resets the logical bit to $|0\rangle$. Note that when the logical qubit is initially in superposition, the physical qubits become entangled.

Quantum flip. The operation $flip^A$ can be quantised in-place as follows:

(1) U_{flip}, pre_{flip}, and $post_{flip}$ have sort $\mathtt{qarray}, \mathtt{qidx}$.
(2) $pre^{\mathcal{A}}_{flip} = post^{\mathcal{A}}_{flip} = \mathbb{1}$.
(3) $U^{\mathcal{A}}_{flip} |a\rangle |i\rangle = |flip^A(a, i)\rangle |i\rangle$ for any $a \in A_{\mathtt{array}}$ and $i \in A_{\mathtt{idx}}$.

As usual, we simply use *qflip* to denote the gate U_{flip} for convenience.

Note that *qflip* implements the classical operation *flip* in the computational basis. It leaves the index unchanged and applies a Pauli X gate to the physical qubit indexed by i.

Quantum correction. The operation $corr^A$ can be quantised in-place as follows:

(1) U_{corr}, pre_{corr}, and $post_{corr}$ have sort $\mathtt{qarray}, \mathtt{qsyn}$.
(2) $pre^{\mathcal{A}}_{corr} = post^{\mathcal{A}}_{corr} = \mathbb{1}$.
(3) $U^{\mathcal{A}}_{corr} |a\rangle |\perp\rangle = |corr^A(a)\rangle |syn(a)\rangle$ for any $a \in A_{\mathtt{array}}$, where

$$syn(a) = \begin{cases} \perp, & a = corr^A(a), \\ j, & j \text{ is the unique index such that } a[j] \neq maj(a). \end{cases} \tag{22}$$

As usual, we simply use *qcorr* to denote the gate U_{corr} for convenience.

Intuitively, *qcorr* implements the classical *corr* in the computational basis and produces the syndrome $syn(a)$. In the computational basis, when a is a code word, the syndrome is $\perp$; otherwise, it is the index of the unique physical bit that differs from $maj(a)$, which is the error location when a is obtained from a bit-flip on a code word. Note that *qcorr* coherently performs the syndrome extraction and error correction together.

According to Theorem 4.5, the equations in Equation (21) can be lifted to a specification of the quantum repetition code. Here, we give a direct specification that the quantum repetition code corrects a single Pauli X error.

Proposition 6.1. *The quantum algebra $\mathcal{A}$ with gates qenc, qflip, and qcorr satisfies the following specification:*

$$\forall q_l : \mathtt{lqbit}. \quad \Big(qenc[q_l, q_a]; qcorr[q_a, q_s] |0\rangle_{q_a} |\perp\rangle_{q_s} = qenc[q_l, q_a] |0\rangle_{q_a} |\perp\rangle_{q_s}\Big) \Downarrow q_a, \tag{23}$$

$$\begin{aligned} \forall q_l : \mathtt{lqbit},\ q_i : \mathtt{qidx}. \quad \Big(&qenc[q_l, q_a]; qflip[q_a, q_i]; qcorr[q_a, q_s] |0\rangle_{q_a} |\perp\rangle_{q_s} \\ &= qenc[q_l, q_a] |0\rangle_{q_a} |\perp\rangle_{q_s}\Big) \Downarrow q_a. \end{aligned} \tag{24}$$

The first equation states that *qcorr* leaves a quantum code word unchanged. The second equation states that it recovers the quantum code word after *qflip* introduces a Pauli X error on any physical qubit. In both equations, q_a is the only principal variable.

6.3 Shor Code

The 3-qubit quantum repetition code in Section 6.2 can only correct a Pauli X error. Inspired by the repetition code, the Shor code [33] is the first quantum error-correcting code that can correct an arbitrary single-qubit error. Let us examine the 9-qubit Shor code as a quantisation of a 9-bit repetition code. We reuse the classical signature *Sig* in Section 6.1 with slight modifications: the operation symbol *flip* in Section 6.1 is replaced by two operation symbols $\mathit{flipX}, \mathit{flipZ} : \mathtt{array}, \mathtt{idx} \longrightarrow \mathtt{array}$ of the same sort.

Consider the canonical algebra B for a 9-bit repetition code. It gives

$$B_{\mathtt{lbit}} = B_{\mathtt{pbit}} = \{0, 1\}, \qquad B_{\mathtt{idx}} = \{0, 1, \ldots, 8\}, \qquad B_{\mathtt{array}} = B_{\mathtt{idx}} \longrightarrow B_{\mathtt{pbit}}.$$

For $a \in B_{\mathtt{array}}$ and $r \in \{0, 1, 2\}$, we define

$$m(a)[r] = \left\lfloor \frac{\sum_{t=0}^{2} a[3r+t]}{2} \right\rfloor, \qquad \mathit{maj}(a) = \left\lfloor \frac{\sum_{r=0}^{2} m(a)[r]}{2} \right\rfloor.$$

Intuitively, the 9 physical bits of a are split into three groups, where the r^{th} group consists of $a[3r], a[3r+1], a[3r+2]$. The value $m(a)[r]$ is the majority of the r^{th} group, while $\mathit{maj}(a)$ is the majority of $m(a)[0], m(a)[1], m(a)[2]$.

In the algebra B, the operations are interpreted as

$$\begin{aligned} \mathit{enc}^B(x)[i] &= x, \\ \mathit{flipX}^B(a,i)[j] = \mathit{flipZ}^B(a,i)[j] &= \begin{cases} a[j] \oplus 1, & j = i, \\ a[j], & j \neq i, \end{cases} \\ \mathit{corr}^B(a)[i] &= \mathit{maj}(a), \end{aligned}$$

where $x \in B_{\mathtt{lbit}}$, $a \in B_{\mathtt{array}}$, and $i, j \in B_{\mathtt{idx}}$. Here, although both *flipX* and *flipZ* are interpreted as a bit flip error in algebra B, their quantisations will correspond to different quantum errors later.

Similar to the algebra A for the 3-bit repetition code (see Section 6.1), the algebra B satisfies

$$\begin{aligned} \forall x : \mathtt{lbit}.\quad & \mathit{corr}(\mathit{enc}(x)) = \mathit{enc}(x), \\ \forall x : \mathtt{lbit}, i : \mathtt{idx}.\quad & \mathit{corr}(\mathit{flipX}(\mathit{enc}(x), i)) = \mathit{enc}(x), \\ \forall x : \mathtt{lbit}, i : \mathtt{idx}.\quad & \mathit{corr}(\mathit{flipZ}(\mathit{enc}(x), i)) = \mathit{enc}(x). \end{aligned} \tag{25}$$

Suppose $\mathit{QSig} = (S_q, \mathcal{K}, \mathcal{U})$ is the quantum signature obtained from a quantisation η of this signature. Let

$$\mathtt{lqbit} = \eta(\mathtt{lbit}), \qquad \mathtt{pqbit} = \eta(\mathtt{pbit}), \qquad \mathtt{qidx} = \eta(\mathtt{idx}), \qquad \mathtt{qarray} = \eta(\mathtt{array}).$$

In addition, let $\mathtt{qsyn}$ be an ancilla sort for syndromes, with designated state $|0_{\mathtt{qsyn}}\rangle = |\bot, \bot, \bot, \bot\rangle$. Consider a quantum algebra $\mathcal{B}$ with

$$\begin{aligned} \mathcal{H}^{\mathcal{B}}_{\mathtt{lqbit}} = \mathcal{H}^{\mathcal{B}}_{\mathtt{pqbit}} &= \mathrm{span}\{|0\rangle, |1\rangle\}, & \mathcal{H}^{\mathcal{B}}_{\mathtt{qidx}} &= \mathrm{span}\{|0\rangle, |1\rangle, \ldots, |8\rangle\}, \\ \mathcal{H}^{\mathcal{B}}_{\mathtt{qarray}} &= \bigotimes_{i=0}^{8} \mathcal{H}^{\mathcal{B}}_{\mathtt{pqbit}}, & \mathcal{H}^{\mathcal{B}}_{\mathtt{qsyn}} &= \mathrm{span}\big\{|s\rangle : s \in \{0,1,2,\bot\}^4\big\}. \end{aligned}$$

As usual, we denote $|a\rangle = |a[0], \ldots, a[8]\rangle$. In particular, the designated state $|0_{\mathtt{qarray}}\rangle = |0\rangle^{\otimes 9}$. For a single-qubit unitary operator U, let $U_i = \mathbb{1}^{\otimes i} \otimes U \otimes \mathbb{1}^{\otimes(8-i)}$, which applies U on the qubit indexed by i. The designated functions $k_{\mathtt{lbit}}$, $k_{\mathtt{pbit}}$, $k_{\mathtt{idx}}$, and $k_{\mathtt{array}}$ are chosen as in Section 6.2.

Next, we explain how the operation symbols *enc*, *flipX*, *flipZ*, and *corr* are quantised to gate symbols in $\mathcal{U}$.

Quantum encoding. For $x \in \{0, 1\}$, let

$$|\Psi_x\rangle = \left(\frac{|000\rangle + (-1)^x |111\rangle}{\sqrt{2}} \right)^{\otimes 3}. \tag{26}$$

The operation enc^B can be quantised out-of-place as follows:

(1) U_{enc}, $\mathit{pre}_{\mathit{enc}}$, and $\mathit{post}_{\mathit{enc}}$ have sort $\mathtt{lqbit}, \mathtt{qarray}$.

(2) $pre^{\mathcal{B}}_{enc} = \mathbb{1}$, and $post^{\mathcal{B}}_{enc} |0\rangle |\Psi_x\rangle = |0\rangle |enc^B(x)\rangle$ for any $x \in B_{\texttt{1bit}}$.
(3) $U^{\mathcal{B}}_{enc} |x\rangle |0\rangle^{\otimes 9} = |0\rangle |\Psi_x\rangle$ for any $x \in B_{\texttt{1bit}}$.

We simply use *qenc* to denote the gate U_{enc} for convenience.

Compared with Section 6.2, the gate *qenc* here encodes the computational-basis $|x\rangle$ into a Shor code word $|\Psi_x\rangle$. If the logical qubit is initially in superposition, the physical qubits are in the corresponding superposition of $|\Psi_0\rangle$ and $|\Psi_1\rangle$.

Quantum flip. The operations $flipX^B$ and $flipZ^B$ can be quantised in-place as follows:

(1) U_{flipX}, U_{flipZ}, pre_{flipX}, pre_{flipZ}, $post_{flipX}$, and $post_{flipZ}$ have sort `qarray`, `qidx`.
(2) $pre^{\mathcal{B}}_{flipX} = post^{\mathcal{B}}_{flipX} = \mathbb{1}$ and $pre^{\mathcal{B}}_{flipZ} = post^{\mathcal{B}}_{flipZ} = (\prod_{i=0}^{8} H_i) \otimes \mathbb{1}$,
(3) $U^{\mathcal{B}}_{flipX} = \sum_{i=0}^{8} X_i \otimes |i\rangle\langle i|$ and $U^{\mathcal{B}}_{flipZ} = \sum_{i=0}^{8} Z_i \otimes |i\rangle\langle i|$.

We simply use *qflipX* and *qflipZ* to denote the gates U_{flipX} and U_{flipZ}, respectively, for convenience.

The gates *qflipX* and *qflipZ* leave the index $|i\rangle$ unchanged and apply Pauli X and Z gates, respectively, on the physical qubit indexed by i. Thus, they introduce the corresponding Pauli errors.

Quantum correction. For $a \in B_{\texttt{array}}$, we define $syn(a) \in \{0, 1, 2, \perp\}^4$ as follows. For $r \in \{0, 1, 2\}$,

$$syn(a)[r] = \begin{cases} \perp, & a[3r+t] = m(a)[r] \text{ for every } t \in \{0,1,2\}, \\ t, & t \text{ is the unique index such that } a[3r+t] \neq m(a)[r]. \end{cases}$$

Further, we define

$$syn(a)[3] = \begin{cases} \perp, & m(a)[r] = maj(a) \text{ for every } r \in \{0,1,2\}, \\ r, & r \text{ is the unique index such that } m(a)[r] \neq maj(a). \end{cases}$$

Intuitively, $syn(a)$ is the syndrome produced for an arbitrary a. For each $r \in \{0, 1, 2\}$, if there is an entry in the r^{th} group of a that differs from $m(a)[r]$, then $syn(a)[r]$ records the index of this entry; otherwise, it records $\perp$. Similarly, if an entry $m(a)[r]$ differs from $maj(a)$, then $syn(a)[3]$ records the index r; otherwise, it records $\perp$. Consequently, it is easy to check that $a \mapsto (maj(a), syn(a))$ is a bijection.

For $P \in \{\mathbb{1}, X, Y, Z\}$, $i = 3r + t$, and $k \in \{0, 1, 2, 3\}$, where $r, t \in \{0, 1, 2\}$, let us define $\sigma(P, i) \in \{0, 1, 2, \perp\}^4$ by

$$\sigma(P,i)[k] = \begin{cases} t, & k = r \text{ and } P \in \{X, Y\}, \\ r, & k = 3 \text{ and } P \in \{Z, Y\}, \\ \perp, & o.w. \end{cases} \tag{27}$$

Intuitively, $\sigma(P, i)$ is the syndrome needed to correct the Pauli error P_i. Suppose we start with a Shor code word. A Pauli X error changes one physical qubit, so its syndrome records the information about $i = 3r + t$. A Pauli Z error on any of the three physical qubits in the same group r has the same effect, so its syndrome only records the r. A Pauli Y error combines X and Z, so its syndrome contains both kinds of information.

Let $\{|\Phi_{x,s}\rangle : x \in \{0, 1\}, s \in \{0, 1, 2, \perp\}^4\}$ be an orthonormal basis of $\mathcal{H}^{\mathcal{B}}_{\texttt{qarray}}$ such that

$$|\Phi_{x,\sigma(P,i)}\rangle = P_i |\Psi_x\rangle \tag{28}$$

for $x \in B_{\texttt{1bit}}$, $P \in \{\mathbb{1}, X, Y, Z\}$, and $i \in B_{\texttt{idx}}$. The existence of such a basis is proved in Lemma A.2. Here, in $|\Phi_{x,s}\rangle$, x records the logical bit and s records the syndrome.

The operation $corr^B$ can be quantised in-place as follows:

(1) U_{corr}, pre_{corr}, and $post_{corr}$ have sort `qarray`, `qsyn`.

(2) $pre^{\mathcal{B}}_{corr} \ket{a}\ket{\bot,\bot,\bot,\bot} = \ket{\Phi_{maj(a),syn(a)}}\ket{\bot,\bot,\bot,\bot}$ and $post^{\mathcal{B}}_{corr}\ket{\Psi_x}\ket{s} = \ket{enc^B(x)}\ket{s}$ for every $x \in B_{\texttt{1bit}}$, $a \in B_{\texttt{array}}$, and $s \in \{0,1,2,\bot\}^4$.
(3) $U^{\mathcal{B}}_{corr}\ket{\Phi_{x,s}}\ket{\bot,\bot,\bot,\bot} = \ket{\Psi_x}\ket{s}$ for every $x \in B_{\texttt{1bit}}$ and $s \in \{0,1,2,\bot\}^4$.

We simply use *qcorr* to denote the gate U_{corr} for convenience.

Intuitively, pre_{corr} implements the bijection $a \mapsto (maj(a), syn(a))$ by mapping $\ket{a}$ to $\ket{\Phi_{maj(a),syn(a)}}$, while leaving the initial syndrome in q_s unchanged. The quantum gate *qcorr* maps $\ket{\Phi_{x,s}}$ to the corrected Shor code word $\ket{\Psi_x}$ and records the syndrome in q_s. Finally, $post_{corr}$ maps $\ket{\Psi_x}$ to the computational-basis repetition code word $\ket{enc^B(x)}$.

As in Section 5, we can derive specifications for the Shor code by lifting Equation (21) using Theorem 4.5. Beyond that, the Shor code actually satisfies stronger error-correction properties. The following equations specify that the Shor code corrects a single Pauli X, Z, or Y error.

Proposition 6.2. *The quantum algebra $\mathcal{B}$ with gates qenc, qflipX, qflipZ, and qcorr satisfies the following specification:*

$$\forall q_l : \texttt{lqbit}.\quad \Big(qenc[q_l,q_a]; qcorr[q_a,q_s]\ket{0}_{q_a}\ket{\bot,\bot,\bot,\bot}_{q_s} = qenc[q_l,q_a]\ket{0}_{q_a}\ket{\bot,\bot,\bot,\bot}_{q_s}\Big) \Downarrow q_a, \tag{29}$$

$$\forall q_l : \texttt{lqbit},\ q_i : \texttt{qidx}.\quad \Big(qenc[q_l,q_a]; qflipX[q_a,q_i]; qcorr[q_a,q_s]\ket{0}_{q_a}\ket{\bot,\bot,\bot,\bot}_{q_s} = qenc[q_l,q_a]\ket{0}_{q_a}\ket{\bot,\bot,\bot,\bot}_{q_s}\Big) \Downarrow q_a, \tag{30}$$

$$\forall q_l : \texttt{lqbit},\ q_i : \texttt{qidx}.\quad \Big(qenc[q_l,q_a]; qflipZ[q_a,q_i]; qcorr[q_a,q_s]\ket{0}_{q_a}\ket{\bot,\bot,\bot,\bot}_{q_s} = qenc[q_l,q_a]\ket{0}_{q_a}\ket{\bot,\bot,\bot,\bot}_{q_s}\Big) \Downarrow q_a, \tag{31}$$

$$\forall q_l : \texttt{lqbit},\ q_i : \texttt{qidx}.\quad \Big(qenc[q_l,q_a]; qflipX[q_a,q_i]; qflipZ[q_a,q_i]; qcorr[q_a,q_s]\ket{0}_{q_a}\ket{\bot,\bot,\bot,\bot}_{q_s} = qenc[q_l,q_a]\ket{0}_{q_a}\ket{\bot,\bot,\bot,\bot}_{q_s}\Big) \Downarrow q_a. \tag{32}$$

The first equation states that *qcorr* leaves an error-free Shor code word unchanged. The second and third equations state that *qcorr* recovers the code word after a Pauli X or Z error, respectively. The fourth equation covers a Pauli Y error through sequential application of *qflipX* and *qflipZ*. In all equations, q_a is the only principal variable. Following the standard quantum error-correction argument, being able to correct Pauli X, Y, Z errors means that the Shor code corrects an arbitrary single-qubit error.

7 Embeddings of Quantum Data Types

So far, we have only considered the quantisation of individual ADTs. In classical computing, various relationships between different ADTs have been exploited for practical applications in software engineering. In particular, an embedding allows values of one ADT to be represented as values of another while preserving the relevant operations and semantics. Consequently, embeddings between ADTs can be used for applications such as data refinement and software component reuse. To enable these kinds of applications for quantum ADTs, in this section we ask under what conditions quantisation preserves structural relationships between algebras. In classical universal algebra, an embedding faithfully maps one algebra into another while preserving all operations. This section investigates when such an embedding lifts to an embedding between the corresponding quantum algebras.

Definition 7.1 (Homomorphisms of algebras). Let A and A' be two algebras of signature $Sig = (S, OP)$. Then:

(1) A homomorphism from A to A' is a family $g = (g_s)_{s\in S}$ of mappings $g_s : A_s \to A'_s$ such that for each operation symbol $O : s_1, \ldots, s_n \to s$, and for all $a_1 \in A_{s_1}, \ldots, a_n \in A_{s_n}$, it holds that

$$g_s(O^A(a_1, \ldots, a_n)) = O^{A'}(g_{s_1}(a_1), \ldots, g_{s_n}(a_n)).$$

Equivalently, the following diagram commutes:

$$\begin{tikzcd} A_{s_1} \times \cdots \times A_{s_n} \arrow[r, "O^A"] \arrow[d, "g_{s_1}\times\cdots\times g_{s_n}"'] & A_s \arrow[d, "g_s"] \\ A'_{s_1} \times \cdots \times A'_{s_n} \arrow[r, "O^{A'}"'] & A'_s \end{tikzcd}$$

(2) If all g_s are injective, then g is called an embedding.
(3) If all g_s are bijective, then g is called an isomorphism.

Let $\mathcal{H}$ and $\mathcal{H}'$ be two Hilbert spaces. A linear operator $T : \mathcal{H} \to \mathcal{H}'$ is called an isometric embedding if

$$\langle T(x)|T(y)\rangle_{\mathcal{H}'} = \langle x|y\rangle_{\mathcal{H}}$$

for all $x, y \in \mathcal{H}$, where $\langle\cdot|\cdot\rangle_{\mathcal{H}}, \langle\cdot|\cdot\rangle_{\mathcal{H}'}$ denote the inner products in $\mathcal{H}$ and $\mathcal{H}'$, respectively. It is easy to see that T is injective and norm-preserving:

$$\|T(x)\|_{\mathcal{H}'} = \|x\|_{\mathcal{H}}$$

for every $x \in \mathcal{H}$, and the image $T(\mathcal{H})$ is a closed subspace of $\mathcal{H}'$.

Definition 7.2 (Embeddings between quantum algebras). Let $\mathcal{A}$ and $\mathcal{A}'$ be two quantum algebras of signature $Sig_q = (S_q, \mathcal{K}, \mathcal{U})$. Then an embedding from $\mathcal{A}$ to $\mathcal{A}'$ is a family $G = (G_s)_{s\in S_q}$ of isometric embeddings $G_s : \mathcal{H}_s \to \mathcal{H}'_s$ such that

(1) for each quantum state symbol $|\psi\rangle \in \mathcal{K}$ of sort $s_1, \ldots, s_n$, it holds that

$$(G_{s_1} \otimes \cdots \otimes G_{s_n})(|\psi\rangle^{\mathcal{A}}) = |\psi\rangle^{\mathcal{A}'}; \tag{33}$$

(2) for each quantum gate symbol $U \in \mathcal{U}$ of sort $s_1, \ldots, s_n$, and for any $|\varphi\rangle \in \bigotimes_{i=1}^n \mathcal{H}_{s_i}$,

$$(G_{s_1} \otimes \cdots \otimes G_{s_n})(U^{\mathcal{A}}|\varphi\rangle) = U^{\mathcal{A}'}((G_{s_1} \otimes \cdots \otimes G_{s_n})|\varphi\rangle). \tag{34}$$

Equivalently, the following diagram commutes:

$$\begin{tikzcd} \bigotimes_{i=1}^n \mathcal{H}_{s_i} \arrow[r, "U^{\mathcal{A}}"] \arrow[d, "G_{s_1}\otimes\cdots\otimes G_{s_n}"'] & \bigotimes_{i=1}^n \mathcal{H}_{s_i} \arrow[d, "G_{s_1}\otimes\cdots\otimes G_{s_n}"] \\ \bigotimes_{i=1}^n \mathcal{H}'_{s_i} \arrow[r, "U^{\mathcal{A}'}"'] & \bigotimes_{i=1}^n \mathcal{H}'_{s_i} \end{tikzcd}$$

Since $G_{s_1}, \ldots, G_{s_n}$ are all isometric embeddings, $\mathcal{H}' \triangleq (G_{s_1} \otimes \cdots \otimes G_{s_n})\left(\bigotimes_{i=1}^n \mathcal{H}_{s_i}\right)$ is a closed subspace of $\bigotimes_{i=1}^n \mathcal{H}'_{s_i}$. Moreover, $G_{s_1} \otimes \cdots \otimes G_{s_n} : \bigotimes_{i=1}^n \mathcal{H}_{s_i} \to \mathcal{H}'$ and its inverse $(G_{s_1} \otimes \cdots \otimes G_{s_n})^\dagger : \mathcal{H}' \to \bigotimes_{i=1}^n \mathcal{H}_{s_i}$ are unitary operators. Thus, the condition in Equation (34) can be rewritten as:

$$U^{\mathcal{A}} = (G_{s_1} \otimes \cdots \otimes G_{s_n})^\dagger \circ U^{\mathcal{A}'} \circ (G_{s_1} \otimes \cdots \otimes G_{s_n})$$

. If all G_s are unitary operators, then G is called an isomorphism. In this case, $G_s(\mathcal{H}_s) = \mathcal{H}'_s$ for every $s \in S_q$, and $\mathcal{H}' = \bigotimes_{i=1}^n \mathcal{H}'_{s_i}$.

In the remainder of this section, we give conditions under which an embedding of algebras induces an embedding of their quantisations.

7.1 Embeddings of Quantisations

Let A and A' be two algebras of the same signature $Sig = (S, OP)$, and let $g = (g_s)_{s \in S} : A \to A'$ be an embedding. Let η be a quantisation from Sig to $QSig = (S_q, \mathcal{K}, \mathcal{U})$ that satisfies $S_q = \eta(S)$. Let $\mathcal{A}$ and $\mathcal{A}'$ be quantisations of A and A' along η, respectively.

Recall that the quantisation of $s \in S$ is $s_q \in S_q$ along η. For each $s \in S$, we define $G_{s_q} : \mathcal{H}^{\mathcal{A}}_{s_q} \to \mathcal{H}^{\mathcal{A}'}_{s_q}$ to be the linear operator that satisfies the following condition:

$$G_{s_q}(|a\rangle) = |g_s(a)\rangle \text{ for every } a \in A_s. \tag{35}$$

We call $G = (G_{s_q})_{s \in S}$ the quantisation of $g = (g_s)_{s \in S}$ along η. Since each g_s is injective, $\{|g_s(a)\rangle : a \in A_s\}$ is an orthonormal set, which means each G_{s_q} is an isometric embedding. If g is further an isomorphism, then each G_{s_q} is a unitary.

For concreteness, we consider BO- and PO-quantisations in Section 4. For each $s \in S$, let $k^A_s : A_s \to \mathbb{Z}_{|A_s|}$ and $k^{A'}_s : A'_s \to \mathbb{Z}_{|A'_s|}$ be the two designated bijections as in Section 4, and let $\boxplus^A_s$ and $\boxplus^{A'}_s$ be defined as in Section 4.

We first examine when a BO-quantisation can be embedded into another.

Proposition 7.3 (Embeddings of BO-quantisations). *Suppose that every state symbol in $\mathcal{K}$ is introduced by η, and that every gate symbol in $\mathcal{U}$ is introduced by η. Let $\mathcal{A}$ and $\mathcal{A}'$ be BO-quantisations of A and A', respectively, along η. If for each $s \in S$ and $a, b \in A_s$,*

$$g_s(a \boxplus^A_s b) = g_s(a) \boxplus^{A'}_s g_s(b),$$

or equivalently, the following diagram commutes:

$$\begin{tikzcd} A_s \times A_s \arrow[r, "\boxplus^A_s"] \arrow[d, "g_s \times g_s"'] & A_s \arrow[d, "g_s"] \\ A'_s \times A'_s \arrow[r, "\boxplus^{A'}_s"'] & A'_s \end{tikzcd}$$

then G in Equation (35) *is an embedding from $\mathcal{A}$ to $\mathcal{A}'$.*

Next, we examine when a PO-quantisation is isomorphic to another. Unlike the BO-quantisations, compatibility with the quantum Fourier transform in Definition 4.8 forces every g_s to be surjective. Consequently, the corresponding result for PO-quantisations is stated only for isomorphisms.

Proposition 7.4 (Isomorphisms of PO-quantisations). *Suppose that every state symbol in $\mathcal{K}$ is introduced by η, and that every gate symbol in $\mathcal{U}$ is introduced by η. Suppose g is an isomorphism. Let $\mathcal{A}$ and $\mathcal{A}'$ be PO-quantisations of A and A', respectively, along η. If for each $s \in S$,*

$$k^{A'}_s \circ g_s = k^A_s,$$

or equivalently, the following diagram commutes (noting that $|A_s| = |A'_s|$):

$$\begin{tikzcd} A_s \arrow[r, "k_s^A"] \arrow[d, "g_s"'] & \mathbb{Z}_{|A_s|} \arrow[d, equal] \\ A'_s \arrow[r, "k_s^{A'}"'] & \mathbb{Z}_{|A'_s|} \end{tikzcd}$$

then G in Equation (35) *is an isomorphism from $\mathcal{A}$ to $\mathcal{A}'$.*

8 Products of Quantum Data Types

Embeddings discussed in the last section describe one of the simplest structural relationships between quantum ADTs. We now turn to how to construct a new quantum ADT from simpler ones. In classical computing, products of ADTs enable us to compose independently specified abstractions while preserving their interfaces and invariants, and they find applications in composite data structures, modular software composition, and other settings. To introduce the same capability into quantum software engineering, in this section we define the notion of the product of ADTs. At the classical level, the domains of two algebras are combined using Cartesian products. At the quantum level, the corresponding quantum domains are combined using tensor products. Furthermore, we present the conditions under which these products are compatible with the quantisations of ADTs.

Definition 8.1. Let A and B be two algebras of signature $Sig = (S, OP)$. Then their product is defined as the algebra $C = A \times B$ of the same signature Sig satisfying the following conditions:

(1) for each sort $s \in S$, $C_s = A_s \times B_s$ (Cartesian product);
(2) for each operation symbol $O : s_1, \ldots, s_n \to s$, O^C is defined as follows: for any $a_i \in A_{s_i}$ and $b_i \in B_{s_i}$ $(i = 1, \ldots, n)$,

$$O^C((a_1, b_1), \ldots, (a_n, b_n)) = (O^A(a_1, \ldots, a_n), O^B(b_1, \ldots, b_n)). \tag{36}$$

In particular, for each constant symbol $c :\to s$, we have $c^C = (c^A, c^B)$.

We next define the corresponding product of quantum algebras. Let $\overline{s} = s_1, \ldots, s_n$ be a string of quantum sorts. Define the canonical unitary operator

$$T_{\overline{s}} : \bigotimes_{i=1}^{n} \left(\mathcal{H}_{s_i}^{\mathcal{A}} \otimes \mathcal{H}_{s_i}^{\mathcal{B}}\right) \to \left(\bigotimes_{i=1}^{n} \mathcal{H}_{s_i}^{\mathcal{A}}\right) \otimes \left(\bigotimes_{i=1}^{n} \mathcal{H}_{s_i}^{\mathcal{B}}\right)$$

by

$$T_{\overline{s}}\left(\bigotimes_{i=1}^{n} |\alpha_i\beta_i\rangle\right) = \left(\bigotimes_{i=1}^{n} |\alpha_i\rangle\right) \otimes \left(\bigotimes_{i=1}^{n} |\beta_i\rangle\right) \tag{37}$$

for any $|\alpha_i\rangle \in \mathcal{H}_{s_i}^{\mathcal{A}}$ and $|\beta_i\rangle \in \mathcal{H}_{s_i}^{\mathcal{B}}$ $(i = 1, \ldots, n)$ together with linear extension.

Definition 8.2. Let $\mathcal{A}$ and $\mathcal{B}$ be two quantum algebras of signature $QSig = (S_q, \mathcal{K}, \mathcal{U})$. Then their product is defined as the quantum algebra $C = \mathcal{A} \otimes \mathcal{B}$ of the same signature $QSig$ satisfying the following conditions:

(1) for each sort $s \in S_q$, $\mathcal{H}_s^C = \mathcal{H}_s^{\mathcal{A}} \otimes \mathcal{H}_s^{\mathcal{B}}$ (tensor product);
(2) for each quantum state symbol $|\psi\rangle \in \mathcal{K}$ of sort $s_1, \ldots, s_n$, $|\psi\rangle^C = T_{s_1,\ldots,s_n}^{-1}(|\psi\rangle^{\mathcal{A}} \otimes |\psi\rangle^{\mathcal{B}})$;
(3) for each quantum gate symbol $U \in \mathcal{U}$ of sort $s_1, \ldots, s_n$,

$$U^C = T_{s_1,\ldots,s_n}^{-1} \circ (U^{\mathcal{A}} \otimes U^{\mathcal{B}}) \circ T_{s_1,\ldots,s_n} \tag{38}$$

is a unitary operator on $\bigotimes_{i=1}^{n} \mathcal{H}_{s_i}^{C} = \bigotimes_{i=1}^{n} (\mathcal{H}_{s_i}^{\mathcal{A}} \otimes \mathcal{H}_{s_i}^{\mathcal{B}})$, where $U^{\mathcal{A}} \otimes U^{\mathcal{B}}$ is defined by

$$\left(U^{\mathcal{A}} \otimes U^{\mathcal{B}}\right)(|\alpha_1 \dots \alpha_n\rangle \otimes |\beta_1 \dots \beta_n\rangle) = U^{\mathcal{A}}(|\alpha_1 \dots \alpha_n\rangle) \otimes U^{\mathcal{B}}(|\beta_1 \dots \beta_n\rangle), \tag{39}$$

for any $|\alpha_i\rangle \in \mathcal{H}_{s_i}^{\mathcal{A}}$ and $|\beta_i\rangle \in \mathcal{H}_{s_i}^{\mathcal{B}}$ $(i = 1, \dots, n)$ and extending linearly. Equivalently, Equation (38) can be expressed by the following commutative diagram:

$$\begin{tikzcd}
\bigotimes_{i=1}^{n} \mathcal{H}_{s_i}^{C} \arrow[r, "U^{C}"] \arrow[d, "T_{s_1,\dots,s_n}"'] & \bigotimes_{i=1}^{n} \mathcal{H}_{s_i}^{C} \\
\left(\bigotimes_{i=1}^{n} \mathcal{H}_{s_i}^{\mathcal{A}}\right) \otimes \left(\bigotimes_{i=1}^{n} \mathcal{H}_{s_i}^{\mathcal{B}}\right) \arrow[r, "U^{\mathcal{A}} \otimes U^{\mathcal{B}}"'] & \left(\bigotimes_{i=1}^{n} \mathcal{H}_{s_i}^{\mathcal{A}}\right) \otimes \left(\bigotimes_{i=1}^{n} \mathcal{H}_{s_i}^{\mathcal{B}}\right) \arrow[u, "T_{s_1,\dots,s_n}^{-1}"']
\end{tikzcd}$$

8.1 Products of Quantisations

Let A and B be two algebras of the same signature $Sig = (S, OP)$, and let $C = A \times B$. Suppose η is a quantisation from Sig to $QSig$ that satisfies $S_q = \eta(S)$. Suppose that $\mathcal{A}$, $\mathcal{B}$, and C are quantisations of A, B, and C, respectively, along η.

Recall that the quantisation of $s \in S$ is $s_q \in S_q$ along η. For each $s \in S$, we define $J_{s_q} : \mathcal{H}_{s_q}^{C} \to \mathcal{H}_{s_q}^{\mathcal{A}} \otimes \mathcal{H}_{s_q}^{\mathcal{B}}$ to be a linear operator that satisfies the following condition:

$$J_{s_q} |(a, b)\rangle = |a\rangle \otimes |b\rangle \tag{40}$$

for every $a \in A_s$ and $b \in B_s$. Note that J_{s_q} is a unitary. Let $J = (J_{s_q})_{s \in S}$.

For concreteness, we consider when a BO-quantisation (and PO-quantisation) is compatible with products. For each $D \in \{A, B, C\}$ and $s \in S$, let $k_s^D : D_s \to \mathbb{Z}_{|D_s|}$ be the designated bijection, and let $\boxplus_s^D$ be defined as in Section 4.

Proposition 8.3 (Products of BO-quantisations). *Suppose that every state symbol in $\mathcal{K}$ is introduced by η, and that every gate symbol in $\mathcal{U}$ is introduced by η. Let $\mathcal{A}$, $\mathcal{B}$, and C be BO-quantisations of A, B, and $C = A \times B$, respectively, along the same quantisation η. If*

$$(a, b) \boxplus_s^C (a', b') = \left(a \boxplus_s^A a', b \boxplus_s^B b'\right) \tag{41}$$

for any $s \in S$, $a, a' \in A_s$, and $b, b' \in B_s$, then J in Equation (40) *is an isomorphism from C to $\mathcal{A} \otimes \mathcal{B}$.*

Proposition 8.4 (Products of PO-quantisations). *Suppose that every state symbol in $\mathcal{K}$ is introduced by η, and that every gate symbol in $\mathcal{U}$ is introduced by η. Let $\mathcal{A}$, $\mathcal{B}$, and C be PO-quantisations of A, B, and $C = A \times B$, respectively, along the same quantisation η. If*

$$\exp\left(\frac{2\pi i\, k_s^C((a,b))k_s^C((a',b'))}{|C_s|}\right) = \exp\left(\frac{2\pi i\, k_s^A(a)k_s^A(a')}{|A_s|}\right)\exp\left(\frac{2\pi i\, k_s^B(b)k_s^B(b')}{|B_s|}\right) \tag{42}$$

for any $s \in S$, $a, a' \in A_s$, and $b, b' \in B_s$, then J is an isomorphism from C to $\mathcal{A} \otimes \mathcal{B}$.

9 Conclusion

This paper developed a universal-algebraic framework for data abstraction in quantum programming. We formally defined the notion of abstract quantum data types and a quantisation of classical data types and proved that their equational specifications can be soundly lifted to their quantisations. Generalised bit-oracle and phase-oracle quantisations arise as instances of this construction and are related through phase kickback. Further, the examples of quantum arrays, the three-qubit repetition code, and the nine-qubit Shor code illustrate how quantum data and operations can be organised and specified within the framework. We also established conditions under which data type quantisation preserves embeddings, isomorphisms, and products.

As a first step toward introducing data abstraction into quantum programming, in this paper, we examine only two typical forms of quantisation, namely bit oracles and phase oracles. One can expect that more forms of quantisation will emerge in the development of future quantum algorithms and programs. We also consider only purely quantum abstract data types in this paper. In practical applications, quantum and classical data are often involved in the same program, and measurement outcomes of quantum data are stored as classical data that subsequently participate in classical computations. This observation suggests another interesting topic for future research: classical-quantum hybrid abstract data types.

We can expect that data abstraction in quantum programming enables programmers to reason about complex quantum behaviour in a manageable way as in classical programming. By hiding mathematical and physical complexity while enforcing quantum constraints, abstraction makes quantum software development more accessible, safer, and more scalable. As quantum computing matures, well-designed abstractions will be crucial for bridging the gap between theoretical algorithms and practical implementations.

Acknowledgments

This work was supported in part by the Australian Research Council under Grant DP250102952.

References

[1] Andris Ambainis. 2007. Quantum walk algorithm for element distinctness. *SIAM J. Comput.* 37 (2007), 210–239.
[2] Andris Ambainis. 2018. Understanding quantum algorithms via query complexity. In *Proceedings of the International Congress of Mathematicians (ICM)*. 3265–3285.
[3] Benjamin Bichsel, Maximilian Baader, Timon Gehr, and Martin Vechev. 2020. Silq: a high-level quantum language with safe uncomputation and intuitive semantics. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. 286–300.
[4] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. 2002. Quantum amplitude amplification and estimation. In *Quantum Computation and Information*. Contemporary Mathematics, Vol. 305. 53–74.
[5] Luca Cardelli and Peter Wegner. 1985. On understanding types, data abstraction, and polymorphism. *Comput. Surveys* 17, 4 (1985), 471–523.
[6] Cirq Developers. 2026. *Cirq*. Zenodo. https://doi.org/10.5281/zenodo.4062499
[7] David Deutsch and Richard Jozsa. 1992. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A* 439 (1992), 553–558.
[8] Olivia Di Matteo. 2025. The art of abstraction in quantum software. In *2025 IEEE/ACM International Workshop on Quantum Software Engineering (Q-SE)*. 25–26.
[9] Hartmut Ehrig and Bernd Mahr. 1985. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*. Springer.
[10] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. 2008. Architectures for a quantum random access memory. *Physical Review A* 78, Article 052310 (2008).
[11] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. 2008. Quantum random access memory. *Physical Review Letters* 100, Article 160501 (2008).
[12] Joseph A. Goguen and José Meseguer. 1992. Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science* 105 (1992), 217–273.
[13] Daniel Gottesman. 2024. Surviving as a quantum computer in a classical world. Textbook manuscript preprint.
[14] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. 2013. Quipper: a scalable quantum programming language. In *Proceedings of the 34th ACM Conference on Programming Language Design and Implementation (PLDI)*. 333–342.
[15] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*. 212–219.
[16] Jingzhe Guo, Huazhe Lou, Jintao Yu, Riling Li, Wang Fang, Junyi Liu, Peixun Long, Shenggang Ying, and Mingsheng Ying. 2023. isQ: An Integrated Software Stack for Quantum Programming. *IEEE Transactions on Quantum Engineering*

4, Article 2500116 (2023), 16 pages.
[17] John V. Guttag. 1977. Abstract data types and the development of data structures. *Commun. ACM* 20, 6 (1977), 396–404.
[18] Bettina Heim, Mathias Soeken, Sarah Marshall, Chris Granade, Martin Roetteler, Alan Geller, Matthias Troyer, and Krysta Svore. 2020. Quantum programming languages. *Nature Reviews Physics* 2 (2020), 709–722.
[19] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. 2024. Quantum computing with Qiskit. arXiv:2405.08810 [quant-ph]
[20] Stacey Jeffery, Robin Kothari, and Frédéric Magniez. 2013. Nested quantum walks with quantum data structures. In *Proceedings of the 24th Annual ACM-SIAM symposium on Discrete algorithms (SODA).* 1474–1485.
[21] Ce Jin and Jakob Nogler. 2023. Quantum speed-ups for string synchronizing sets, longest common substring, and k-mismatch matching. In *Proceedings of the 34th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* 5090–5121.
[22] Emanuel Knill and Raymond Laflamme. 1997. Theory of quantum error-correcting codes. *Physical Review A* 55 (1997), 900–911.
[23] Emanuel H. Knill. 1996. *Conventions for quantum pseudo-code.* Technical Report. Los Alamos National Laboratory.
[24] Mark Koch, Alan Lawrence, Kartik Singhal, Seyon Sivarajah, and Ross Duncan. 2025. GUPPY: Pythonic quantum-classical programming. arXiv:2510.12582
[25] Barbara Liskov. 1987. Data abstraction and hierarchy. *ACM SIGPLAN Notices* 23, 5 (1987), 17–34.
[26] Barbara Liskov and Stephen Zilles. 1974. Programming with abstract data types. *ACM SIGPLAN Notices* 9, 4 (1974), 50–59.
[27] John McCarthy. 1962. Towards a mathematical science of computation. In *IFIP Congress 62.*
[28] Aran Nayebi, Scott Aaronson, Aleksandrs Belovs, and Luca Trevisan. 2015. Quantum lower bound for inverting a permutation with advice. *Quantum Information & Computation* 15 (2015), 901–913.
[29] Michael A. Nielsen and Isaac L. Chuang. 2000. *Quantum Computation and Quantum Information.* Cambridge University Press.
[30] Santiago Núñez-Corrales, Olivia Di Matteo, John Dumbell, Marcus Edwards, Edoardo Giusto, Scott Pakin, and Vlad Stirbu. 2025. Productive quantum programming needs better abstract machines. In *2025 IEEE International Conference on Quantum Computing and Engineering (QCE).* 816–826.
[31] Romain Péchoux, Simon Perdrix, Mathys Rennela, and Vladimir Zamdzhiev. 2020. Quantum programming with inductive datatypes: causality and affine type theory. In *Proceedings of the 23rd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS).* 562–581.
[32] Raphael Seidel, Sebastian Bock, René Zander, Matic Petrič, Niklas Steinmann, Nikolay Tcholtchev, and Manfred Hauswirth. 2024. Qrisp: a framework for compilable high-level programming of gate-based quantum computers. arXiv:2406.14792
[33] Peter W. Shor. 1995. Scheme for reducing decoherence in quantum computer memory. *Physical Review A* 52 (1995), R2493–R2496.
[34] Peter W. Shor. 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* 26 (1997), 1484–1509.
[35] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. 2018. Q#: enabling scalable quantum computing and development with a high-level DSL. In *Proceedings of the Real World Domain Specific Languages Workshop (RWSDL).* Article 7, 10 pages.
[36] Tamás Varga, Yaiza Aragonés-Soria, and Manuel Oriol. 2024. Quantum types: going beyond qubits and quantum gates. In *Proceedings of the 5th ACM/IEEE International Workshop on Quantum Software Engineering (Q-SE).* 49–52.
[37] Qisheng Wang, Zhean Xu, and Zhicheng Zhang. 2026. Quantum data structure for range minimum query. *J. Comput. System Sci.* 157, Article 103756 (2026).
[38] Niklaus Wirth. 1976. *Algorithms + Data Structures = Programs.* Prentice-Hall.
[39] Mingsheng Ying. 2024. *Foundations of Quantum Programming* (second ed.). Morgan Kaufmann.
[40] Charles Yuan and Michael Carbin. 2022. Tower: data structures in quantum superposition. *Proceedings of the ACM on Programming Languages* 6, OOPSLA2, Article 134 (2022), 259–288 pages.

A Deferred Proofs

All of the theorems and propositions in the main text of this paper were presented without proofs. In this appendix, we give all of these proofs. We first prove the following lemma, which is useful for proving Theorem 4.5.

Lemma A.1. *Let A and $\mathcal{A}$ be as in Theorem 4.5. Let t be a classical term involving classical variables $x_1 : s_1, \ldots, x_n : s_n$ and p be the principal variable of $\eta(t)$. We use $\overline{q} = q_1, \ldots, q_n$ to denote the designated variables $q_{x_1}, \ldots, q_{x_n}$ as in Definition 4.3. Denote $\overline{a} = a_1, \ldots, a_n \in A_{s_1}, \ldots, A_{s_n}$. Define $\sigma_{\overline{a}}$ to be an assignment of classical variables with $\sigma_{\overline{a}}(x_i) = a_i$ for $i = 1, \ldots, n$. For any quantum state*

$$|\varphi\rangle_{\overline{q}} = \sum_{\overline{a}} \beta_{\overline{a}} |\overline{a}\rangle_{\overline{q}},$$

we have

$$eval_{|\varphi\rangle}(\eta(t)) \downarrow p = \sum_{\overline{a}} |\beta_{\overline{a}}|^2 \, |eval_{\sigma_{\overline{a}}}(t)\rangle\langle eval_{\sigma_{\overline{a}}}(t)|_p \,,$$

where eval on the LHS is defined in the quantum algebra $\mathcal{A}$, and eval on the RHS is defined in the classical algebra A.

Proof. We first consider the lemma for the computational-basis states; that is, $|\varphi\rangle = |\overline{a}\rangle$ for some $\overline{a}$. This case can be proved by induction on the length of t with the following induction hypothesis:

- **IH**: Let $\overline{o}$ denote the variables involved in the quantisation $\eta(t)$ but not in $p, \overline{q}$. For any classical assignment σ of $\overline{x}$, let $a_i = \sigma(x_i)$ and $\overline{a} = a_1, \ldots, a_n$. There exists a quantum state $|\gamma\rangle$ such that:
$$eval_{|\overline{a}\rangle}(\eta(t)) = |\overline{a}\rangle_{\overline{q}} \, |eval_\sigma(t)\rangle_p \, |\gamma\rangle_{\overline{o}} \,. \tag{43}$$

There are three cases:

- If $t = c$, then $\eta(t) = |\psi_c\rangle_p$ and $|\psi_c\rangle^{\mathcal{A}} = |c^A\rangle$. Thus, Equation (43) holds.
- If $t = x$, then $\eta(x) = SUM_s[q_x, q] \, |0\rangle_q$. Since $k_s(0_s^A) = 0$ and $a \boxplus_s 0_s^A = a$, we have
$$eval_{|\overline{a}\rangle}(\eta(x)) = SUM_s[q_x, q]^{\mathcal{A}} \, |a\rangle_{q_x} |0_s^A\rangle_q = |a\rangle_{q_x} |a\rangle_q \,.$$
Thus, Equation (43) holds.
- If $t = O(t_1, \ldots, t_m)$, then
$$\eta(t) = C_1; \ldots; C_m; pre_O[p_1, \ldots, p_m, \overline{r}]; U_O[p_1, \ldots, p_m, \overline{r}]; post_O[p_1, \ldots, p_m, \overline{r}] \, |\psi_1\rangle \ldots |\psi_m\rangle \, |\overline{0}\rangle_{\overline{r}} \,,$$
where each $\eta(t_i) = C_i \, |\psi_i\rangle$ has principal variable p_i. Note that in the in-place case, $p = p_i$ for the designated index i in Definition 4.1; in the out-of-place case, p is the first variable in $\overline{r}$. By the **IH** for $t_1, \ldots, t_m$, there exists a $|\Gamma\rangle$ such that
$$eval_{|\overline{a}\rangle}(C_1; \ldots; C_m \, |\psi_1\rangle \ldots |\psi_m\rangle) = |\overline{a}\rangle_{\overline{q}} \, |b_1, \ldots, b_m\rangle_{p_1,\ldots,p_m} \, |\Gamma\rangle_{\overline{o_1},\ldots,\overline{o_m}} \,, \tag{44}$$
where each $b_i = eval_\sigma(t_i)$, and $\overline{o_i}$ are additional variables involved in $\eta(t_i)$ but not in $p_i, \overline{q}$. By Equation (5), we further have
$$post_O^{\mathcal{A}} U_O^{\mathcal{A}} pre_O^{\mathcal{A}} \, |b_1, \ldots, b_m\rangle_{p_1,\ldots,p_m} \, |\overline{0}\rangle_{\overline{r}} = |O^A(b_1, \ldots, b_m)\rangle_p \, |\gamma'\rangle \tag{45}$$
for some $|\gamma'\rangle$. Combining Equations (44) and (45), we obtain
$$eval_{|\overline{a}\rangle}(\eta(t)) = |\overline{a}\rangle_{\overline{q}} \, |eval_\sigma(t)\rangle_p \, (|\Gamma\rangle \, |\gamma'\rangle)_{\overline{o}},$$
and Equation (43) holds.

Since all $|\overline{a}\rangle_{\overline{q}}$ are orthogonal, by linearity and the tracing out of $\overline{q}$, it is easy to prove the lemma for any quantum state $|\varphi\rangle$. □

Proof of Theorem 4.5. Since $\mathcal{A}$ is already a quantum algebra of *QSig*, it suffices to show that $\mathcal{A}$ satisfies every equation in $\eta(E)$. Consider any equation in E:

$$e = (\forall x_1 : s_1, \ldots, x_n : s_n)(t_1 = t_2).$$

We use $\overline{q} = q_1, \ldots, q_n$ to denote the designated variables $q_{x_1}, \ldots, q_{x_n}$ corresponding to $x_1, \ldots, x_n$, as in Definition 4.3.

Let p be the common principal variable of $\eta(t_1)$ and $\eta(t_2)$, as in Definition 4.4. Consider an arbitrary quantum state

$$|\varphi\rangle_{\overline{q}} = \sum_{\overline{a}} \beta_{\overline{a}} |\overline{a}\rangle_{\overline{q}}.$$

Since A is an algebra of *Spec* and $e \in E$, we have $A \models e$. Therefore, for every $\overline{a} \in A_{s_1} \times \cdots \times A_{s_n}$,

$$eval_{\sigma_{\overline{a}}}(t_1) = eval_{\sigma_{\overline{a}}}(t_2). \tag{46}$$

Applying Lemma A.1 to t_1 and t_2, and using Equation (46), we obtain

$$eval_{|\varphi\rangle}(\eta(t_1)) \downarrow p = eval_{|\varphi\rangle}(\eta(t_2)) \downarrow p,$$

which is essentially $\mathcal{A} \models \eta(e)$ by the validity of equations (see Definition 3.6). The conclusion then follows. □

Proof of Proposition 5.1. Note that in Equation (20), all variables are principal variables. To show that the equation is valid in algebra $\mathcal{A}$, by linearity, it suffices to consider any computational-basis state of variables q_a, q_i, q_v, q_u:

$$|\varphi\rangle = |a\rangle_{q_a} |j\rangle_{q_i} |v\rangle_{q_v} |u\rangle_{q_u}.$$

Let a' be an array with $a'[i] = a[i]$ for $i \neq j$ and $a'[j] = u$. By evaluating the LHS of Equation (20), we have

$$\begin{aligned}
&eval_{|\varphi\rangle}(qwrite[q_a, q_i, q_u]; qread[q_a, q_i, q_v]) \\
&\quad = qread[q_a, q_i, q_v]\Big(qwrite[q_a, q_i, q_u] |a\rangle_{q_a} |j\rangle_{q_i} |v\rangle_{q_v} |u\rangle_{q_u}\Big) \\
&\quad = qread[q_a, q_i, q_v]\Big(|a'\rangle_{q_a} |j\rangle_{q_i} |v\rangle_{q_v} |a[j]\rangle_{q_u}\Big) \\
&\quad = |a'\rangle_{q_a} |j\rangle_{q_i} |v \boxplus u\rangle_{q_v} |a[j]\rangle_{q_u}.
\end{aligned}$$

By evaluating the RHS of Equation (20), we have

$$\begin{aligned}
&eval_{|\varphi\rangle}(SUM[q_u, q_v]; qwrite[q_a, q_i, q_u]) \\
&\quad = qwrite[q_a, q_i, q_u]\Big(SUM[q_u, q_v] |a\rangle_{q_a} |j\rangle_{q_i} |v\rangle_{q_v} |u\rangle_{q_u}\Big) \\
&\quad = qwrite[q_a, q_i, q_u]\Big(|a\rangle_{q_a} |j\rangle_{q_i} |v \boxplus u\rangle_{q_v} |u\rangle_{q_u}\Big) \\
&\quad = |a'\rangle_{q_a} |j\rangle_{q_i} |v \boxplus u\rangle_{q_v} |a[j]\rangle_{q_u}.
\end{aligned}$$

By linearity, we have

$$eval_{|\varphi\rangle}(qwrite[q_a, q_i, q_u]; qread[q_a, q_i, q_v]) = eval_{|\varphi\rangle}(SUM[q_u, q_v]; qwrite[q_a, q_i, q_u])$$

for any quantum state $|\varphi\rangle$, and the conclusion follows.

□

Proof of Proposition 6.1. Note that in Equations (23) and (24), q_a is the only principal variable. By Equations (21) and (22) and the definition of *qcorr*, for every $x \in \{0, 1\}$ and $i \in \{0, 1, 2\}$, we have

$$\begin{aligned}&qcorr[q_a, q_s]\ |enc^A(x)\rangle_{q_a} |\bot\rangle_{q_s} \\ &\quad = |corr^A(enc^A(x))\rangle_{q_a} |syn(enc^A(x))\rangle_{q_s} = |enc^A(x)\rangle_{q_a} |\bot\rangle_{q_s},\end{aligned} \tag{47}$$

$$\begin{aligned}&qcorr[q_a, q_s]\ |flip^A(enc^A(x), i)\rangle_{q_a} |\bot\rangle_{q_s} \\ &\quad = |corr^A(flip^A(enc^A(x), i))\rangle_{q_a} |syn(flip^A(enc^A(x), i))\rangle_{q_s} = |enc^A(x)\rangle_{q_a} |i\rangle_{q_s}.\end{aligned} \tag{48}$$

To prove that Equation (23) is valid in $\mathcal{A}$, consider an arbitrary state of the variable q_l:

$$|\varphi\rangle_{q_l} = \sum_{x\in\{0,1\}} \alpha_x |x\rangle_{q_l}.$$

By evaluating the LHS and using Equation (47), we have

$$eval_{|\varphi\rangle}\Big(qenc[q_l, q_a]; qcorr[q_a, q_s]\ |0\rangle_{q_a} |\bot\rangle_{q_s}\Big) = |0\rangle_{q_l} \sum_x \alpha_x |x, x, x\rangle_{q_a} |\bot\rangle_{q_s}.$$

By evaluating the RHS, we have

$$eval_{|\varphi\rangle}\Big(qenc[q_l, q_a]\ |0\rangle_{q_a} |\bot\rangle_{q_s}\Big) = |0\rangle_{q_l} \sum_x \alpha_x |x, x, x\rangle_{q_a} |\bot\rangle_{q_s}.$$

Both evaluations give the same state, so Equation (23) is valid.

To prove that Equation (24) is valid in $\mathcal{A}$, consider an arbitrary state of variables q_l, q_i:

$$|\varphi\rangle_{q_l q_i} = \sum_{\substack{x\in\{0,1\}\\ i\in\{0,1,2\}}} \alpha_{x,i} |x\rangle_{q_l} |i\rangle_{q_i}.$$

By evaluating the LHS and using Equation (48), we have

$$eval_{|\varphi\rangle}\Big(qenc[q_l, q_a]; qflip[q_a, q_i]; qcorr[q_a, q_s]\ |0\rangle_{q_a} |\bot\rangle_{q_s}\Big) = |0\rangle_{q_l} \sum_{x,i} \alpha_{x,i} |x, x, x\rangle_{q_a} |i\rangle_{q_i} |i\rangle_{q_s}.$$

By evaluating the RHS, we have

$$eval_{|\varphi\rangle}\Big(qenc[q_l, q_a]\ |0\rangle_{q_a} |\bot\rangle_{q_s}\Big) = |0\rangle_{q_l} \sum_{x,i} \alpha_{x,i} |x, x, x\rangle_{q_a} |i\rangle_{q_i} |\bot\rangle_{q_s}.$$

Since q_a is the principal variable, we trace out q_l, q_i, q_s. Both sides of Equation (24) give

$$\sum_{i=0}^{2} \sum_{x,x'\in\{0,1\}} \alpha_{x,i}\overline{\alpha_{x',i}}\, |x, x, x\rangle\langle x', x', x'|_{q_a}.$$

Therefore, Equation (24) is valid. The conclusion immediately follows. □

Lemma A.2. *There exists an orthonormal basis* $\left\{|\Phi_{x,s}\rangle : x \in \{0, 1\}, s \in \{0, 1, 2, \bot\}^4\right\}$ *that satisfies Equation* (28).

Proof. We first consider the following orthonormal basis of $\left(\mathcal{H}^{\mathcal{B}}_{\mathtt{pqbit}}\right)^{\otimes 3}$:

$$\left\{\frac{|000\rangle \pm |111\rangle}{\sqrt{2}}, \frac{|100\rangle \pm |011\rangle}{\sqrt{2}}, \frac{|010\rangle \pm |101\rangle}{\sqrt{2}}, \frac{|001\rangle \pm |110\rangle}{\sqrt{2}}\right\}$$

Let $i = 3r + t$, where $r, t \in \{0, 1, 2\}$. For $P \in \{\mathbb{1}, X, Y, Z\}$, applying P_i to the state in Equation (26) gives

$$P_i \left|\Psi_x\right\rangle = \left(\frac{\left|000\right\rangle + (-1)^x \left|111\right\rangle}{\sqrt{2}}\right)^{\otimes r} \otimes P_t\left(\frac{\left|000\right\rangle + (-1)^x \left|111\right\rangle}{\sqrt{2}}\right) \otimes \left(\frac{\left|000\right\rangle + (-1)^x \left|111\right\rangle}{\sqrt{2}}\right)^{\otimes(2-r)}, \quad (49)$$

where P_t applies P on the qubit indexed by t within a group of three qubits. In particular, for different P_t, we have

$$\begin{aligned}
X_t\left(\frac{\left|000\right\rangle + (-1)^x \left|111\right\rangle}{\sqrt{2}}\right) &= \begin{cases} \frac{\left|100\right\rangle + (-1)^x \left|011\right\rangle}{\sqrt{2}}, & t = 0, \\ \frac{\left|010\right\rangle + (-1)^x \left|101\right\rangle}{\sqrt{2}}, & t = 1, \\ \frac{\left|001\right\rangle + (-1)^x \left|110\right\rangle}{\sqrt{2}}, & t = 2, \end{cases} \\
Z_t\left(\frac{\left|000\right\rangle + (-1)^x \left|111\right\rangle}{\sqrt{2}}\right) &= \frac{\left|000\right\rangle - (-1)^x \left|111\right\rangle}{\sqrt{2}}, \\
Y_t\left(\frac{\left|000\right\rangle + (-1)^x \left|111\right\rangle}{\sqrt{2}}\right) &= \mathrm{i}\begin{cases} \frac{\left|100\right\rangle - (-1)^x \left|011\right\rangle}{\sqrt{2}}, & t = 0, \\ \frac{\left|010\right\rangle - (-1)^x \left|101\right\rangle}{\sqrt{2}}, & t = 1, \\ \frac{\left|001\right\rangle - (-1)^x \left|110\right\rangle}{\sqrt{2}}, & t = 2. \end{cases}
\end{aligned} \quad (50)$$

From Equation (27), $\sigma(\mathbb{1}, i) = (\perp, \perp, \perp, \perp)$ for every $i \in B_{\mathtt{idx}}$, and $\mathbb{1}_i \left|\Psi_x\right\rangle = \left|\Psi_x\right\rangle$. Apart from the case $P = P' = \mathbb{1}$, $\sigma(P, i) = \sigma(P', i')$ for distinct pairs $(P, i) \neq (P', i')$ only if $P = P' = Z$ and $i, i' \in \{3r, 3r + 1, 3r + 2\}$ for some $r \in \{0, 1, 2\}$. Moreover, for every $x \in B_{\mathtt{1bit}}$ and $r \in \{0, 1, 2\}$, we have

$$Z_{3r} \left|\Psi_x\right\rangle = Z_{3r+1} \left|\Psi_x\right\rangle = Z_{3r+2} \left|\Psi_x\right\rangle.$$

Thus, for every $x \in B_{\mathtt{1bit}}$, $P, P' \in \{\mathbb{1}, X, Y, Z\}$, and $i, i' \in B_{\mathtt{idx}}$,

$$\sigma(P, i) = \sigma(P', i') \Rightarrow P_i \left|\Psi_x\right\rangle = P'_{i'} \left|\Psi_x\right\rangle.$$

By Equations (27), (49) and (50), for every $x, x' \in B_{\mathtt{1bit}}$, $P, P' \in \{\mathbb{1}, X, Y, Z\}$, and $i, i' \in B_{\mathtt{idx}}$, it is easy to verify that

$$\left\langle\Psi_x\right| P_i^\dagger P'_{i'} \left|\Psi_{x'}\right\rangle = \begin{cases} 1, & x = x' \text{ and } \sigma(P, i) = \sigma(P', i'), \\ 0, & o.w. \end{cases}$$

Therefore, Equation (28) is well-defined and these $\left|\Phi_{x,\sigma(P,i)}\right\rangle$ can be extended to an orthonormal basis of $\mathcal{H}^{\mathcal{B}}_{\mathtt{qarray}}$. □

Proof of Proposition 6.2. Note that in Equations (29) to (32), q_a is the only principal variable. By Equations (27) and (28) and the definition of *qcorr*, for every $x \in B_{\mathtt{1bit}}$, we have

$$\begin{aligned}
&qcorr[q_a, q_s] \left|\Psi_x\right\rangle_{q_a} \left|\perp, \perp, \perp, \perp\right\rangle_{q_s} \\
&\quad = qcorr[q_a, q_s] \left|\Phi_{x,(\perp,\perp,\perp,\perp)}\right\rangle_{q_a} \left|\perp, \perp, \perp, \perp\right\rangle_{q_s} = \left|\Psi_x\right\rangle_{q_a} \left|\perp, \perp, \perp, \perp\right\rangle_{q_s}.
\end{aligned} \quad (51)$$

Similarly, for every $x \in B_{\mathtt{1bit}}$, $P \in \{X, Y, Z\}$, and $i \in B_{\mathtt{idx}}$,

$$\begin{aligned}
&qcorr[q_a, q_s]\, P_i \left|\Psi_x\right\rangle_{q_a} \left|\perp, \perp, \perp, \perp\right\rangle_{q_s} \\
&\quad = qcorr[q_a, q_s] \left|\Phi_{x,\sigma(P,i)}\right\rangle_{q_a} \left|\perp, \perp, \perp, \perp\right\rangle_{q_s} = \left|\Psi_x\right\rangle_{q_a} \left|\sigma(P, i)\right\rangle_{q_s}.
\end{aligned} \quad (52)$$

To prove that Equation (29) is valid in $\mathcal{B}$, consider an arbitrary state of the variable q_l:

$$|\varphi\rangle_{q_l} = \sum_{x\in\{0,1\}} \alpha_x |x\rangle_{q_l}.$$

By evaluating the LHS and using Equation (51), we have

$$eval_{|\varphi\rangle}\Big(qenc[q_l,q_a]; qcorr[q_a,q_s]\; |0\rangle_{q_a} |\bot,\bot,\bot,\bot\rangle_{q_s}\Big) = |0\rangle_{q_l} \sum_x \alpha_x |\Psi_x\rangle_{q_a} |\bot,\bot,\bot,\bot\rangle_{q_s}.$$

By evaluating the RHS, we have

$$eval_{|\varphi\rangle}\Big(qenc[q_l,q_a]\; |0\rangle_{q_a} |\bot,\bot,\bot,\bot\rangle_{q_s}\Big) = |0\rangle_{q_l} \sum_x \alpha_x |\Psi_x\rangle_{q_a} |\bot,\bot,\bot,\bot\rangle_{q_s}.$$

Both evaluations give the same state, so Equation (29) is valid.

To prove that Equations (30) to (32) are valid in $\mathcal{B}$, consider an arbitrary state of variables q_l, q_i:

$$|\varphi\rangle_{q_lq_i} = \sum_{\substack{x\in\{0,1\}\\ i\in\{0,1,\dots,8\}}} \alpha_{x,i} |x\rangle_{q_l} |i\rangle_{q_i}.$$

By evaluating the LHSs and using Equation (52), we have

$$\begin{aligned}
&eval_{|\varphi\rangle}\Big(qenc[q_l,q_a]; qflipX[q_a,q_i]; qcorr[q_a,q_s]\; |0\rangle_{q_a} |\bot,\bot,\bot,\bot\rangle_{q_s}\Big)\\
&\quad= |0\rangle_{q_l} \sum_{x,i} \alpha_{x,i} |\Psi_x\rangle_{q_a} |i\rangle_{q_i} |\sigma(X,i)\rangle_{q_s},\\
&eval_{|\varphi\rangle}\Big(qenc[q_l,q_a]; qflipZ[q_a,q_i]; qcorr[q_a,q_s]\; |0\rangle_{q_a} |\bot,\bot,\bot,\bot\rangle_{q_s}\Big)\\
&\quad= |0\rangle_{q_l} \sum_{x,i} \alpha_{x,i} |\Psi_x\rangle_{q_a} |i\rangle_{q_i} |\sigma(Z,i)\rangle_{q_s},\\
&eval_{|\varphi\rangle}\Big(qenc[q_l,q_a]; qflipX[q_a,q_i]; qflipZ[q_a,q_i]; qcorr[q_a,q_s]\; |0\rangle_{q_a} |\bot,\bot,\bot,\bot\rangle_{q_s}\Big)\\
&\quad= \mathrm{i}\, |0\rangle_{q_l} \sum_{x,i} \alpha_{x,i} |\Psi_x\rangle_{q_a} |i\rangle_{q_i} |\sigma(Y,i)\rangle_{q_s}.
\end{aligned}$$

By evaluating the RHS, we have

$$eval_{|\varphi\rangle}\Big(qenc[q_l,q_a]\; |0\rangle_{q_a} |\bot,\bot,\bot,\bot\rangle_{q_s}\Big) = |0\rangle_{q_l} \sum_{x,i} \alpha_{x,i} |\Psi_x\rangle_{q_a} |i\rangle_{q_i} |\bot,\bot,\bot,\bot\rangle_{q_s}.$$

Since q_a is the principal variable, we trace out q_l, q_i, q_s. Both sides of Equations (30) to (32) give

$$\sum_{i=0}^{8} \sum_{x,x'\in\{0,1\}} \alpha_{x,i}\overline{\alpha_{x',i}}\, |\Psi_x\rangle\langle\Psi_{x'}|_{q_a}.$$

Therefore, Equations (30) to (32) are valid. The conclusion immediately follows. □

Proof of Proposition 7.3. As observed after Equation (35), every G_{s_q} is an isometric embedding. It suffices to verify the two conditions in Definition 7.2.

(1) Let $|\psi\rangle \in \mathcal{K}$ be a quantum state symbol. Then, there is a constant symbol $c :\longrightarrow s$ such that $|\psi\rangle = \eta(c)$. Since g is a homomorphism, $g_s(c^A) = c^{A'}$, which means

$$G_{s_q} |\psi\rangle^{\mathcal{A}} = G_{s_q} |c^A\rangle = |c^{A'}\rangle = |\psi\rangle^{\mathcal{A}'}.$$

Consequently, Equation (33) holds.

(2) Let $U \in \mathcal{U}$ be a quantum gate symbol. It suffices to consider the following cases.

- Suppose $U = pre_O$ or $U = post_O$ for some nonconstant operation symbol O. This case is trivial because $pre^{\mathcal{B}}_O = post^{\mathcal{B}}_O = \mathbb{1}$ for $\mathcal{B} \in \{\mathcal{A}, \mathcal{A}'\}$.
- Suppose $U = U_O$ for some nonconstant operation symbol $O : s_1, \ldots, s_n \to s$. Write $s_{iq} = \eta(s_i)$ and $s_q = \eta(s)$. For $a_i \in A_{s_i}$ and $a \in A_s$, we have

$$\begin{aligned}
&\left(G_{s_{1q}} \otimes \ldots \otimes G_{s_{nq}} \otimes G_{s_q}\right) U^{\mathcal{A}}_O \left|a_1\right\rangle \cdots \left|a_n\right\rangle \left|a\right\rangle \\
&\quad = \left|g_{s_1}(a_1)\right\rangle \cdots \left|g_{s_n}(a_n)\right\rangle \left|g_s(a \boxplus^A_s O^A(a_1, \ldots, a_n))\right\rangle \\
&\quad = \left|g_{s_1}(a_1)\right\rangle \cdots \left|g_{s_n}(a_n)\right\rangle \left|g_s(a) \boxplus^{A'}_s O^{A'}(g_{s_1}(a_1), \ldots, g_{s_n}(a_n))\right\rangle \\
&\quad = U^{\mathcal{A}'}_O \left(G_{s_{1q}} \otimes \ldots \otimes G_{s_{nq}} \otimes G_{s_q}\right) \left|a_1\right\rangle \cdots \left|a_n\right\rangle \left|a\right\rangle .
\end{aligned}$$

  The second equality follows from the condition about $\boxplus^A_s$ and $\boxplus^{A'}_s$ and the fact that g is a homomorphism.

By linearity, Equation (34) holds.

Thus, G satisfies Definition 7.2 and is an embedding from $\mathcal{A}$ to $\mathcal{A}'$. □

Proof of Proposition 7.4. As observed after Equation (35), every G_{s_q} is unitary. It suffices to verify the two conditions in Definition 7.2. For each $s \in S$, let $N_s = |A_s| = \left|A'_s\right|$ and $\omega_s = e^{2\pi i/N_s}$.

(1) Let $\left|\psi\right\rangle \in \mathcal{K}$ be a quantum state symbol. Similar to the proof of Proposition 7.3, $G_{s_q} \left|\psi\right\rangle^{\mathcal{A}} = \left|\psi\right\rangle^{\mathcal{A}'}$ and therefore Equation (33) holds.

(2) Let $U \in \mathcal{U}$ be a quantum gate symbol. It suffices to consider the following cases.
- Suppose $U = pre_O$ or $post_O$ for some nonconstant operation symbol $O : s_1, \ldots, s_n \to s$. For any $a \in A_s$, since g_s is bijective,

$$\begin{aligned}
G_{s_q} QFT_{|A_s|} \left|a\right\rangle &= \frac{1}{\sqrt{|A_s|}} \sum_{b \in A_s} \omega_s^{k^A_s(a) k^A_s(b)} \left|g_s(b)\right\rangle \\
&= \frac{1}{\sqrt{|A_s|}} \sum_{g_s(b) \in A'_s} \omega_s^{k^{A'}_s(g_s(a)) k^{A'}_s(g_s(b))} \left|g_s(b)\right\rangle \\
&= QFT_{|A_s|} G_{s_q} \left|a\right\rangle .
\end{aligned} \tag{53}$$

  Here, the second equality uses $k^A_s = k^{A'}_s \circ g_s$. Tensoring Equation (53) with $\bigotimes_{i=1}^n G_{s_{iq}}$ followed by linearity leads to Equation (34) for $U = pre_O$. Taking the adjoints proves it for $U = post_O$.
- Suppose $U = U_O$ for some nonconstant operation symbol $O : s_1, \ldots, s_n \to s$. For $a_i \in A_{s_i}$ and $a \in A_s$, Equation (9) gives

$$\begin{aligned}
&(G_{s_{1q}} \otimes \ldots \otimes G_{s_{nq}} \otimes G_{s_q}) U^{\mathcal{A}}_O \left|a_1, \ldots, a_n\right\rangle \left|a\right\rangle \\
&= \omega_s^{k^A_s(a) k^A_s(O^A(a_1, \ldots, a_n))} \left|g_{s_1}(a_1), \ldots, g_{s_n}(a_n)\right\rangle \left|g_s(a)\right\rangle \\
&= \omega_s^{k^{A'}_s(g_s(a)) k^{A'}_s(O^{A'}(g_{s_1}(a_1), \ldots, g_{s_n}(a_n)))} \left|g_{s_1}(a_1), \ldots, g_{s_n}(a_n)\right\rangle \left|g_s(a)\right\rangle \\
&= U^{\mathcal{A}'}_O (G_{s_{1q}} \otimes \ldots \otimes G_{s_{nq}} \otimes G_{s_q}) \left|a_1, \ldots, a_n\right\rangle \left|a\right\rangle .
\end{aligned}$$

  The second equality follows from $k^{A'}_s \circ g_s = k^A_s$ and the fact that g is a homomorphism. By linearity, Equation (34) holds.

Thus, G satisfies Definition 7.2 and is an isomorphism from $\mathcal{A}$ to $\mathcal{A}'$. □

Proof of Proposition 8.3. As observed after Equation (40), every J_{s_q} is unitary. It suffices to verify the two conditions in Definition 7.2.

(1) Let $|\psi\rangle \in \mathcal{K}$ be a quantum state symbol, so $|\psi\rangle = \eta(c)$ for a constant symbol $c :\longrightarrow s$. Since $|\psi\rangle$ has the single sort s_q, $T_{s_q} = \mathbb{1}$. Then

$$J_{s_q} |\psi\rangle^C = J_{s_q} |(c^A, c^B)\rangle = |c^A\rangle \otimes |c^B\rangle = |\psi\rangle^{\mathcal{A}\otimes\mathcal{B}}.$$

Consequently, Equation (33) holds.

(2) Let $U \in \mathcal{U}$ be a quantum gate symbol. It suffices to consider the following cases.
- Suppose $U = pre_O$ or $U = post_O$ for some nonconstant operation symbol O. This case is trivial because $pre_O^{\mathcal{D}} = post_O^{\mathcal{D}} = \mathbb{1}$ for $\mathcal{D} \in \{\mathcal{A}, \mathcal{B}, C\}$, and Equation (38) further implies that $pre_O^{\mathcal{A}\otimes\mathcal{B}} = post_O^{\mathcal{A}\otimes\mathcal{B}} = \mathbb{1}$. Therefore, Equation (34) holds in both cases.
- Suppose $U = U_O$ for some nonconstant operation symbol $O : s_1, \ldots, s_n \longrightarrow s$. Let us denote $\overline{a} = a_1, \ldots, a_n$, $\overline{b} = b_1, \ldots, b_n$, and $\overline{(a,b)} = (a_1, b_1), \ldots, (a_n, b_n)$. For $a_i \in A_{s_i}$, $b_i \in B_{s_i}$, $a \in A_s$, and $b \in B_s$, we have

$$\begin{aligned}
&(J_{s_{1q}} \otimes \ldots \otimes J_{s_{nq}} \otimes J_{s_q}) U_O^C \, |\overline{(a,b)}\rangle \, |(a,b)\rangle \\
&= (J_{s_{1q}} \otimes \ldots \otimes J_{s_{nq}} \otimes J_{s_q}) \, |\overline{(a,b)}\rangle \, |(a,b) \boxplus_s^C O^C(\overline{(a,b)})\rangle \\
&= \bigotimes_{i=1}^{n} (|a_i\rangle \otimes |b_i\rangle) \otimes \Big(|a \boxplus_s^A O^A(\overline{a})\rangle \otimes |b \boxplus_s^B O^B(\overline{b})\rangle \Big) \\
&= T^{-1}_{s_{1q},\ldots,s_{nq},s_q} \Big(U_O^{\mathcal{A}} \, |\overline{a}\rangle \, |a\rangle \otimes U_O^{\mathcal{B}} \, |\overline{b}\rangle \, |b\rangle \Big) \\
&= U_O^{\mathcal{A}\otimes\mathcal{B}} (J_{s_{1q}} \otimes \ldots \otimes J_{s_{nq}} \otimes J_{s_q}) \, |\overline{(a,b)}\rangle \, |(a,b)\rangle .
\end{aligned}$$

  Here, the first equality follows from Equation (7); the second follows from Equations (36), (40) and (41); the third follows from Equations (7) and (37); and the last follows from Equations (37) and (38).

By linearity, Equation (34) holds.

Thus, J satisfies Definition 7.2 and is an isomorphism from C to $\mathcal{A} \otimes \mathcal{B}$. □

Proof of Proposition 8.4. As shown in the proof of Proposition 8.3, Equation (33) holds. It remains to verify Equation (34) for every $U \in \mathcal{U}$ through the following cases.

- Suppose that $U = pre_O$ or $U = post_O$ for some nonconstant operation symbol $O : s_1, \ldots, s_n \longrightarrow s$. Equation (42) implies that for each sort $s \in S$,

$$J_{s_q} QFT_{|C_s|} = \Big(QFT_{|A_s|} \otimes QFT_{|B_s|} \Big) J_{s_q}. \tag{54}$$

  Consequently,

$$\begin{aligned}
&(J_{s_{1q}} \otimes \ldots \otimes J_{s_{nq}} \otimes J_{s_q}) pre_O^C \\
&= (J_{s_{1q}} \otimes \ldots \otimes J_{s_{nq}} \otimes J_{s_q}) (\mathbb{1}_{s_{1q},\ldots,s_{nq}} \otimes QFT_{|C_s|}) \\
&= (J_{s_{1q}} \otimes \ldots \otimes J_{s_{nq}}) \otimes \Big((QFT_{|A_s|} \otimes QFT_{|B_s|}) J_{s_q} \Big) \\
&= T^{-1}_{s_{1q},\ldots,s_{nq},s_q} (pre_O^{\mathcal{A}} \otimes pre_O^{\mathcal{B}}) T_{s_{1q},\ldots,s_{nq},s_q} (J_{s_{1q}} \otimes \ldots \otimes J_{s_{nq}} \otimes J_{s_q}) \\
&= pre_O^{\mathcal{A}\otimes\mathcal{B}} (J_{s_{1q}} \otimes \ldots \otimes J_{s_{nq}} \otimes J_{s_q}),
\end{aligned}$$

  where the second equality uses Equation (54), the third follows from Equation (37), and the final follows from Equation (38). Combining the above together, Equation (34) holds for $U = pre_O$. Taking adjoints proves it for $U = post_O$.

- Suppose $U = U_O$ for some nonconstant operation symbol $O : s_1, \ldots, s_n \longrightarrow s$. Write $\overline{a} = a_1, \ldots, a_n$, $\overline{b} = b_1, \ldots, b_n$, and $\overline{(a,b)} = (a_1, b_1), \ldots, (a_n, b_n)$. For $a_i \in A_{s_i}$, $b_i \in B_{s_i}$, $a \in A_s$, and $b \in B_s$, let $a' = O^A(\overline{a})$ and $b' = O^B(\overline{b})$. By Equation (36), $O^C(\overline{(a,b)}) = (a', b')$. Then we have

$$
\begin{aligned}
&(J_{s_{1q}} \otimes \ldots \otimes J_{s_{nq}} \otimes J_{s_q}) U_O^C \left|\overline{(a,b)}\right\rangle \left|(a,b)\right\rangle \\
&= \exp\left(\frac{2\pi i\, k_s^C((a,b)) k_s^C((a',b'))}{|C_s|}\right) (J_{s_{1q}} \otimes \ldots \otimes J_{s_{nq}} \otimes J_{s_q}) \left|\overline{(a,b)}\right\rangle \left|(a,b)\right\rangle \\
&= \exp\left(\frac{2\pi i\, k_s^A(a) k_s^A(a')}{|A_s|}\right) \exp\left(\frac{2\pi i\, k_s^B(b) k_s^B(b')}{|B_s|}\right) (J_{s_{1q}} \otimes \ldots \otimes J_{s_{nq}} \otimes J_{s_q}) \left|\overline{(a,b)}\right\rangle \left|(a,b)\right\rangle \\
&= U_O^{\mathcal{A} \otimes \mathcal{B}} (J_{s_{1q}} \otimes \ldots \otimes J_{s_{nq}} \otimes J_{s_q}) \left|\overline{(a,b)}\right\rangle \left|(a,b)\right\rangle .
\end{aligned}
$$

  Here, the first equality follows from Equation (9); the second follows from Equation (42); and the last follows from Equations (9) and (38). By linearity, Equation (34) holds.

Thus, J satisfies Definition 7.2 and is an isomorphism from C to $\mathcal{A} \otimes \mathcal{B}$. □