Semantics for Variational Quantum Programming

XIAODONG JIA, Hunan University, China
ANDRE KORNELL, Tulane University, USA
BERT LINDENHOVIUS, Johannes Kepler Universität, Austria
MICHAEL MISLOVE, Tulane University, USA
VLADIMIR ZAMDZHIEV, Inria, France

We consider a programming language that can manipulate both classical and quantum information. Our language is type-safe and designed for variational quantum programming, which is a hybrid classical-quantum computational paradigm. The classical subsystem of the language is the Probabilistic FixPoint Calculus (PFPC), which is a lambda calculus with mixed-variance recursive types, term recursion and probabilistic choice. The quantum subsystem is a first-order linear type system that can manipulate quantum information. The two subsystems are related by mixed classical/quantum terms that specify how classical probabilistic effects are induced by quantum measurements, and conversely, how classical (probabilistic) programs can influence the quantum dynamics. We also describe a sound and computationally adequate denotational semantics for the language. Classical probabilistic effects are interpreted using a recently-described commutative probabilistic monad on DCPO. Quantum effects and resources are interpreted in a category of von Neumann algebras that we show is enriched over (continuous) domains. This strong sense of enrichment allows us to develop novel semantic methods that we use to interpret the relationship between the quantum and classical probabilistic effects. By doing so we provide a very detailed denotational analysis that relates domain-theoretic models of classical probabilistic programming to models of quantum programming.

CCS Concepts: · Theory of computation → Denotational semantics; Operational semantics.

Additional Key Words and Phrases: Variational Quantum Programming, Probabilistic Programming, Semantics

ACM Reference Format:
Xiaodong Jia, Andre Kornell, Bert Lindenhovius, Michael Mislove, and Vladimir Zamdzhiev. 2022. Semantics for Variational Quantum Programming. Proc. ACM Program. Lang. 6, POPL, Article 26 (January 2022), 31 pages. https://doi.org/10.1145/3498687

1 INTRODUCTION

Variational quantum algorithms [McClean et al. 2016; Peruzzo et al. 2014] are increasingly important in quantum computation. The main idea is to use hybrid classical-quantum algorithms that work in tandem to solve computational problems. The classical part of the computation is executed on a classical processor and the quantum part on a quantum device. During the overall computation, intermediary results produced by the quantum device occur with certain probabilities, and then are passed to the classical processor, which performs computations that are used to tune the parameters of the classical component of the algorithm, thereby influencing the quantum dynamics.

This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2022 Copyright held by the owner/author(s).
2473-1421/2022/1-ART26
https://doi.org/10.1145/3498687

Proc. ACM Program. Lang., Vol. 6, No. POPL, Article 26. Publication date: January 2022.
These kinds of hybrid classical-quantum algorithms pose interesting challenges for the design of suitable programming languages. Clearly, if we wish to understand how to program in such scenarios, we need to devise a type system equipped with an operational semantics that correctly models the manipulation of quantum resources. This includes accounting for the fact that quantum measurements induce probabilistic computational effects that are inherited by the classical side of the system. Moreover, quantum information behaves very differently from classical information. For instance, quantum information cannot be copied [Wootters and Zurek 1982]. In order to avoid potential runtime errors, a substructural typing discipline [Benton 1995; Benton and Wadler 1996; Girard 1987] where contraction is restricted is appropriate for the quantum subsystem. But, when manipulating classical information, such restrictions are unnecessary and often inconvenient. Therefore we wish to have a classical (non-linear) subsystem together with a quantum (linear) one that interact nicely with each other. Furthermore, separating the quantum and classical modes of operations has the added benefit that it makes it easier to extend existing classical programming languages with the necessary features for type-safe variational quantum programming.

The purpose of the present paper is to address this challenge by describing a type-safe programming language that combines classical (probabilistic) computation with quantum computation. Another one of our goals is to provide a denotational interpretation so that we may establish useful reasoning principles and therefore cement the design of our language.

Our Contributions. We describe a programming language that is suitable for hybrid classical-quantum computation that we call VQPL, the Variational Quantum Programming Language (§2). The language has two kinds of judgements: a classical (non-linear) judgement that represents classical programs, and a quantum (linear) judgement that represents quantum programs. Our type system also contains hybrid classical-quantum formation rules that explain how classical probabilistic and quantum computation interact with each other (see Figure 7).

From an operational perspective, VQPL supports both classical probabilistic and quantum effects. The quantum dynamics are modelled via a probabilistic reduction relation on quantum configurations (terms with quantum data embedded within them), where the probabilities of reduction are determined in accordance with the laws of quantum mechanics. The classical dynamics are modelled via a probabilistic reduction relation on terms, where the probabilities of reduction are induced by the quantum dynamics. We show that our system VQPL is type-safe (§2.3).

We also provide a denotational interpretation of our system. We use a recently-described commutative probabilistic monad on the category DCPO [Jia et al. 2021b] in order to interpret the classical probabilistic effects (we recall this construction in §3). We interpret quantum effects and resources in a category of hereditarily atomic von Neumann algebras (§4); von Neumann algebras are mathematical structures commonly used to study quantum foundations [Takesaki 2000]. We prove that this category is enriched over continuous domains (§4.3). This is a very strong sense of enrichment that allows us to develop novel semantic methods that we use to interpret the relationship between the quantum and classical probabilistic effects (§5). In particular, we show that the theory of Kegelspitzen [Keimel and Plotkin 2017] provides a crucial link between the two different ways that probability arises on the classical and quantum sides, respectively. This allows us to present all the relevant mathematical structure systematically within a categorical model (§6) that we show provides a sound and strongly adequate interpretation of VQPL (§7). Our paper presents a detailed mathematical and denotational analysis of the link between domain-theoretic models of classical probabilistic programming and quantum programming. We discuss related work in §8.

Remark 1.0.1. Because of space limitations, many proofs are omitted. An extended version of this paper [Jia et al. 2021a] contains most proofs of interest.
2 VQPL - THE VARIATIONAL QUANTUM PROGRAMMING LANGUAGE

In this section we describe the syntax and operational semantics for VQPL. The classical subsystem is the *Probabilistic FixPoint Calculus* (PFPC), the same language as in [Jia et al. 2021b]. PFPC is a call-by-value simply-typed lambda calculus with mixed-variance recursive types, (induced) term recursion and discrete probabilistic choice. The quantum fragment of the language is a first-order linear type system with inductive types and equipped with the usual primitives for manipulating quantum information. This fragment is most similar to [Péchoux et al. 2020b], however in the present paper we choose a Church-style syntax in order to more easily relate it to the classical subsystem. The distinguishing feature of our system is the mixed linear/non-linear and quantum/classical rules that allow the programmer to switch between the classical and quantum modes of operation. These features make our language suitable for programming variational quantum algorithms, where both classical and quantum computation work in synchrony in order to solve computational problems. Our mixed quantum/classical rules have some similarities with the QWIRE/EWIRE languages [Paykin et al. 2017; Rennela and Staton 2020], but both of these languages have some severe limitations that make them unsuitable for describing variational quantum algorithms, whereas these limitations do not apply to our language. This is discussed in more detail in §8.

This section is structured as follows: in §2.1 we describe the types of VQPL; in §2.2 we describe the terms of our language and how they should be understood, and we also describe the type system of VQPL; in §2.3 we prove that VQPL is type-safe; in §2.4 we discuss recursion and how to determine the probability that a term reduces to a given value; finally in §2.5 we present several illustrative program examples. Readers who are not interested in the formation conditions of terms may focus on the term language in §2.2 and the examples in §2.5 to get a feel of the syntax, but we recommend following the exposition in the presented order.

In order to make the paper easier to read, we use **bold** notation for the quantum types, contexts and terms, so that we can easily distinguish them from the classical primitives.

2.1 The Type Structure

We use $X, Y$ to range over classical type variables and we use $X, Y$ to range over quantum type variables. We use $\Theta$ and $\Theta$ to range over classical and quantum type contexts, respectively. Type variables and type contexts are used for the formation of recursive types, just as in FPC [Abadi and Fiore 1996; Fiore and Plotkin 1994]. We say that a classical type context $\Theta = X_1, \ldots, X_n$ is well-formed, written $\Theta \vdash$, whenever all type variables within it are distinct, and likewise for a quantum type context. The empty type context is written as "\cdot". The classical types of our language are ranged over by $P, R$, and the quantum types are ranged over by $A, B$. The grammars for our types are specified in Figure 1. We say that type $\cdot \vdash P$ is well-formed in type context $\Theta$ and write $\Theta \vdash P$ to indicate this, whenever the judgement is derivable from the formation rules presented in Figure 1. Likewise, we write $\Theta \vdash A$, to indicate that quantum type $A$ is well-formed in the quantum type context $\Theta$. Of course, as always, we are only interested in well-formed types and from now on we only deal with such types (inductive/recursive types that are not well-formed run into problems with variable capture). The closed classical types are those where $\cdot \vdash P$ and the closed quantum types are those where $\cdot \vdash A$, i.e., those types which do not contain any free type variables within them. In practice, we are primarily interested in closed types, so the formation rules for terms (§2.2) are described using closed types only. Notice that recursive types may be formed with no restrictions on the admissible logical polarities, just as in FPC.

We now explain how our types should be understood. On the quantum side: $I$ is the quantum unit type, i.e., the type which holds a single unique value; $\text{qbit}$ is the type of qubits (quantum bits); $A \oplus B$ represents quantum sum types, which are primarily used for conditional branching.
of programs; $A \otimes B$ represents quantum pair types, which are primarily used to group together values; $\mu X.A$ is used to form quantum inductive types, which are primarily used to introduce types that contain infinitely many values (see Example 2.1.1). All terms of quantum type obey a linear typing discipline and so these types should be viewed as being linear. On the classical side: $1$ is the classical unit type; $P + R$ is for classical sum types; $P \times R$ is for classical pair types; $P \to R$ is for classical (higher-order) function types; $\mu X.P$ is used to form classical recursive types. All terms of classical type follow a non-linear typing discipline (no restrictions on weakening and contraction), so they should be understood as being non-linear. Note that $Q(A,B)$, the type of first-order quantum lambda abstractions between quantum types $A$ and $B$, is classical (non-linear). This is because our quantum lambda abstractions are first-order and therefore they may be used any number of times (including zero). This type would correspond to $!(A \to B)$ in a call-by-value linear lambda calculus and may be informally thought of in this way.

**Example 2.1.1.** Some important (closed) types are defined as follows: $\textit{Booleans}$ as $\textit{Bool} \defeq 1 + 1$; \textit{Bits} as $\textit{bit} \defeq 1 \oplus 1$; $\textit{Natural numbers}$ as $\textit{Nat} \defeq \mu X.1 + X$; $\textit{Linear/Quantum natural numbers}$ as $\textit{QNat} \defeq \mu X.1 \oplus X$; $\textit{Lists of type } A$ as $\textit{List}(A) \defeq \mu X.1 + (A \times X)$; $\textit{Linear/Quantum lists of type } A$ as $\textit{List}(A) \defeq \mu X.1 \oplus (A \otimes X)$; $\textit{Classical Streams of type } A$ as $\textit{Stream}(A) \defeq \mu X.1 \to A \times X$.

A subset of our classical/quantum types are the observable classical/quantum types, which are defined in Figure 1. We use $O$ and $O$ to range over the observable classical/quantum types, respectively. These types play an important role for some of the mixed quantum-classical rules that we explain later. The observable quantum types may also be understood from a physical perspective because values of these types correspond to physically observable information. An example of a non-observable quantum type is $\textit{qbit}$. Indeed, observing a qubit in the physical sense is done via a quantum measurement, which destroys the qubit and produces a bit as output (note that $\textit{bit}$ is observable in our system). The observable classical types are exactly the ground types, i.e., types formed without any use of classical/quantum function space. The observable quantum types are in a 1-1 correspondence with the observable classical types. For each observable quantum type $O$, we write $|O|$ to indicate its observable classical counterpart. See Figure 1 for a precise definition of $|−|$.
Semantics for Variational Quantum Programming

Quantum Variables $x, y$ Classical Variables $x, y$ Quantum Configurations $C := [\vert \psi \rangle, \ell, q]$

Classical Terms $m, n$ := $x \mid \ell \mid (m, n) \mid \pi_1 m \mid \pi_2 m \mid \text{fold } m \mid \text{unfold } m \mid \lambda x.m \mid mn \mid in_1 m \mid in_2 m \mid \text{(case } m \text{ of in}_1 x \Rightarrow n_1 \mid \text{in}_2 y \Rightarrow n_2 \mid \lambda (x_1, \ldots, x_n).q \mid \text{new } \mid \text{meas } \mid U \mid \text{run } C$

Quantum Terms $q, r$ := $x \mid + \mid q \otimes r \mid \text{let } x \otimes y = q \text{ in } r \mid \text{fold } q \mid \text{unfold } q \mid \text{init } m \mid in_1 q \mid in_2 q \mid \text{(case } q \text{ of in}_1 x \Rightarrow r_1 \mid \text{in}_2 y \Rightarrow r_2 \mid \text{let } x = \text{lift } q \text{ in } r$

Classical Values $v, w$ := $x \mid \ell \mid (v, w) \mid in_1 v \mid in_2 v \mid \text{fold } v \mid \text{new } \mid \text{meas } \mid U$

Quantum Values $v, w$ := $x \mid + \mid v \otimes w \mid \text{in}_1 v \mid \text{in}_2 v \mid \text{fold } v$

Fig. 2. Grammars for terms, values and quantum configurations.

$$
\begin{align*}
\Phi, x : P & \vdash x : P \\
\Phi & \vdash () : 1 \\
\Phi & \vdash m : P \\
\Phi & \vdash (m, n) : P \times P \\
\Phi & \vdash m : P_1 \times P_2 \\
\Phi & \vdash \pi_i m : P_i \\
\Phi, x : P & \vdash m : P \rightarrow R \\
\Phi & \vdash n : P \\
\Phi & \vdash m : P_1 \rightarrow P_2 \\
\Phi & \vdash x : P_1 \rightarrow P_2 \\
\Phi, y : P_2 & \vdash n_1 : R \\
\Phi, y : P_2 & \vdash n_2 : R
\end{align*}
$$

Fig. 3. Formation rules for terms in the (classical) FPC subsystem.

$$
\begin{align*}
\Phi & \vdash \ast : I \\
\Phi, \Gamma_1 & \vdash q : I \\
\Phi, \Gamma_2 & \vdash r : A \\
\Phi, \Gamma_1 & \vdash q : A \\
\Phi, \Gamma_2 & \vdash r : B \\
\Phi, \Gamma_1 & \vdash q : A \oplus B \\
\Phi, \Gamma_2, x : A_1, y : A_2 & \vdash r : B \\
\Phi, \Gamma_1, \Gamma_2 & \vdash \text{let } x \otimes y = q \text{ in } r : B \\
\Phi, \Gamma_1, \Gamma_2 & \vdash \text{fold } q : \mu X.A \\
\Phi, \Gamma & \vdash \text{unfold } q : \mu X.A
\end{align*}
$$

Fig. 4. Formation rules for terms in the purely linear first-order subsystem.

$$
\begin{align*}
\Phi & \vdash \text{new } : Q(\text{bit}, q\text{bit}) \\
\Phi & \vdash \text{meas } : Q(q\text{bit}, \text{bit}) \\
\Phi & \vdash U : Q(q\text{bit}^n, q\text{bit}^n)
\end{align*}
$$

Fig. 5. Formation rules for term constants that manipulate quantum information.

$$
\begin{align*}
\ast & \overset{\text{def}}{=} () \\
v \otimes w & \overset{\text{def}}{=} (v \otimes w) \\
in_1 v & \overset{\text{def}}{=} in_1|v| \\
in_2 v & \overset{\text{def}}{=} in_2|v| \\
fold v & \overset{\text{def}}{=} \text{fold } v
\end{align*}
$$

Fig. 6. Translation between closed and observable quantum/classical values.

$$
\begin{align*}
\Phi, x_1, \ldots, x_n : A_n & \vdash q : B \\
\Phi, \lambda (x_1, \ldots, x_n).q : Q(A_1 \otimes \cdots \otimes A_n, B) & \vdash m : Q(A, B) \\
\Phi, \Gamma & \vdash q : A \\
\Phi & \vdash C : \mu; q\text{bit}^k \\
\Phi & \vdash \text{run } C : \vert O \rangle
\end{align*}
$$

Fig. 7. Formation rules for terms that mediate between the quantum and classical modes of operation.
2.2 The Term Language

For the formation of terms and term contexts, we implicitly assume that all types mentioned within are closed and well-formed. We use \( x, y \) to range over classical term variables and \( x, y \) to range over quantum term variables. Classical term contexts are ranged over by \( \Phi \) and quantum term contexts by \( \Gamma \). The (well-formed) term contexts are simply lists of (distinct) variables with their types.

The term grammars of VQPL are specified in Figure 2. We write \( \Phi \vdash m : P \) to indicate that a classical term \( m \) is well-formed and has type \( P \) given classical context \( \Phi \). We write \( \Phi; \Gamma \vdash q : A \) to indicate that a quantum term \( q \) is well-formed and has type \( A \), given classical context \( \Phi \) and quantum context \( \Gamma \). A classical term \( m \) of type \( A \) is closed when \( \cdot \vdash m : A \) and in this case we also simply write \( m : A \). Likewise we write \( q : A \) when \( \vdash q : A \), and then we also say \( q \) is closed. We use \( v, w \) to range over classical values and \( v, w \) to range over quantum values (see Figure 2).

Example 2.2.1. Important closed values include: the (classical) false and true values given by \( \text{ff} \stackrel{\text{def}}{=} \text{in}_{1}() : \text{Bool} \) and \( \text{tt} \stackrel{\text{def}}{=} \text{in}_{2}() : \text{Bool} \); the false and true bits are defined by \( \text{ff} \stackrel{\text{def}}{=} \text{in}_{1} : \text{bit} \) and \( \text{tt} \stackrel{\text{def}}{=} \text{in}_{2} : \text{bit} \); the zero natural number \( \text{zero} \stackrel{\text{def}}{=} \text{fold in}_{1}() : \text{Nat} \) and the successor function \( \text{succ} \stackrel{\text{def}}{=} \lambda n_{\text{Nat}}. \text{fold in}_{2}n : \text{Nat} \rightarrow \text{Nat} \); quantum versions of zero and succ may also be defined.

Execution of (quantum) programs is described by the small-step call-by-value operational semantics in Figures 8–10. If \( m \) and \( n \) are classical terms, we write \( m \xrightarrow{P} n \) to indicate that \( m \) reduces to \( n \) with probability \( p \in [0, 1] \) in exactly one step. Note that the probabilistic behaviour of reduction in the classical subsystem is induced by quantum measurements from the quantum subsystem.

2.2.1 Quantum Configurations. Reduction for the quantum fragment is described, as usual, in terms of quantum configurations \( |\psi\rangle, \ell, q \), which may be seen as terms with embedded quantum information. We describe quantum configurations following [Pagani et al. 2014; Péchoux et al. 2020b]. Given \( n \in \mathbb{N} \), we write \( \text{qbit}^n \stackrel{\text{def}}{=} \text{qbit} \otimes \cdots \otimes \text{qbit} \) for the \( n \)-fold tensor product of \( \text{qbit} \).

Definition 2.2.2 (Quantum Configuration). A quantum configuration is a triple \( |\psi\rangle, \ell, q \), where: \( |\psi\rangle \) is a pure quantum state, i.e., a normalised vector in \( \mathbb{C}^2^n \); \( q \) is a quantum term; \( \ell : \text{QFV}(q) \rightarrow \{1, \ldots, \text{dim}(|\psi\rangle)\} \) is a function from the set of free quantum variables of \( q \) into the indicated set, where \( \text{dim}(|\psi\rangle) \) is the dimension of \( |\psi\rangle \). We refer to \( \ell \) as the linking function. A quantum configuration \( |\psi\rangle, \ell, q \) is well-formed in classical context \( \Phi \) with type \( A \) and \( k \) auxiliary qubits, which we write as \( \Phi \vdash |\psi\rangle, \ell, q : A; \text{qbit}^k \), whenever the following conditions are satisfied:

- \( \Phi ; x_1 : \text{qbit}, \ldots, x_m : \text{qbit} \vdash q : A \) is a well-formed quantum term.
- \( \text{dim}(|\psi\rangle) = m + k. \)
- The linking function \( \ell : \{x_1, \ldots, x_m\} \rightarrow \{1, \ldots, m + k\} \) is injective.

A configuration \( |\psi\rangle, \ell, q \) is total if \( \Phi \vdash |\psi\rangle, \ell, q : A; \text{qbit}^0 \), which we abbreviate by \( \Phi \vdash |\psi\rangle, \ell, q : A \). Thus, in a total configuration \( \ell \) defines a 1 - 1 correspondence between the qubits of \( |\psi\rangle \) and the free quantum variables of \( q \). A configuration is closed if \( \cdot \vdash |\psi\rangle, \ell, q : A; \text{qbit}^k \), for some \( k \in \mathbb{N} \).

We are primarily interested in well-formed configurations that are both total and closed. Nevertheless, the premises of the structural reduction rules in the operational semantics include non-total configurations that have some auxiliary qubits not used by the quantum term, so it is necessary also to consider non-total configurations (see [Pagani et al. 2014; Péchoux et al. 2020b] for more details). Otherwise, the configurations in the premises would not be typable, so it is necessary to allow auxiliary qubits as part of the formation conditions. Likewise, the denotational semantics includes configurations that are not closed, because the interpretation of closed terms may be defined using non-closed terms (e.g. lambda abstractions).
The linking function $\ell$ in a configuration $[[\psi]], \ell, q$ associates the free variables of $q$, each of type qbit, to specific qubits of the quantum state $|\psi\rangle$: $\ell$ is needed because some of the qubits in $|\psi\rangle$ may be entangled, in which case $|\psi\rangle$ cannot be broken down into smaller quantum states.

We use calligraphic letters $C, D$ to range over quantum configurations. Given quantum configurations $C$ and $D$ we write $C \xrightarrow{p} D$ to indicate that $C$ reduces to $D$ with probability $p$ in exactly one step. This is how we model the execution of quantum programs. A value configuration is a configuration $V = [[\psi], \ell, v]$, where $v$ is a quantum value. Reduction in the quantum subsystem terminates at value configurations, just as reduction in the classical system terminates at values.

2.2.2 The Subsystem FPC. We have organised the term formation rules and the associated reduction rules into several subsystems, which we now describe. Figure 3 specifies the formation rules for the classical terms that make up the subsystem FPC (which is well-known [Abadi and Fiore 1996; Fiore and Plotkin 1994; Fiore 1994]). The notation $P[\mu X.P/X]$ represents type substitution, which is defined as usual. The reduction rules for these terms are standard and are shown in Figure 8.

2.2.3 The Subsystem QPL. Figure 4 describes the formation rules for the quantum terms that make up a first-order linear subsystem with inductive types. These terms and their reduction rules are all standard, but they are now described on quantum configurations in Figure 8. For the structural reduction rules involving quantum evaluation contexts, the notation $\ell \cup \ell_0 = \emptyset$ indicates that the linking functions $\ell$ and $\ell_0$ have disjoint domains and the notation $\ell \cap \ell_0$ indicates the disjoint union of the two linking functions. The terms from Figure 4 do not directly modify the quantum state $|\psi\rangle$.

Figure 5 lists the formation rules for the term constants that we use to manipulate quantum information. Note that these constants are values of type $Q(A, B)$ and therefore are classical/non-linear (and may be used any number of times). The associated reduction rules for quantum function application using these constants are presented in Figure 9. The term $U(x_1 \otimes \cdots \otimes x_k)$ applies the unitary operation $U$ to the qubits identified by the variables $x_1, \ldots, x_k$ and modifies the quantum state in the configuration accordingly. The term new ff (resp. new tt) creates a new qubit in state $|0\rangle$ (resp. $|1\rangle$), creates a fresh new qubit variable that points to it and modifies the quantum configuration accordingly. The term meas y measures the qubit identified by variable y and produces bit tt or ff with probability given by the Born rule of quantum mechanics. This operation irreversibly modifies the quantum state and causes a probabilistic computational effect.

The terms in Figures 4 and 5 can be thought of as jointly making up a subsystem that we call QPL (it is roughly equivalent to QPL in [Péchoux et al. 2020b; Selinger 2004a]), which is a first-order language for quantum programming.

2.2.4 Mixed Classical/Quantum Terms. Both subsystems (P)FPC (Figure 3) and QPL have been studied previously (for very different purposes). The main distinguishing feature of VQPL is that it demonstrates how these subsystems can be combined and used simultaneously for variational quantum programming. The terms and formation rules that allow us to achieve this are presented in Figure 7, and we now describe them in greater detail.

The term $\lambda(x_1, \ldots, x_n).q$ is a value which represents a first-order quantum lambda abstraction. Note that this value is actually classical (non-linear). The term $mq$ represents quantum function application. In our view, this is the most interesting rule in VQPL. This is because its subterm $m: Q(A, B)$ represents a probabilistic classical program that eventually reduces to a quantum lambda abstraction. Because of this, the term $mq$ combines classical probabilistic computation with quantum computation (represented by the subterm $q$), and this is reflected in the associated reduction rules in Figure 10. Providing a semantic interpretation of this term requires considerable effort and the development of novel semantic methods, as we show later.

Proc. ACM Program. Lang., Vol. 6, No. POPL, Article 26. Publication date: January 2022.
\[\prod_1(v, w) \rightarrow v \quad \prod_2(v, w) \rightarrow w \quad \text{unfold fold } v \rightarrow v \quad (\lambda x.m)v \rightarrow m[v/x]\]

(case in1v of in1x \Rightarrow n1 | in2y \Rightarrow n2) \rightarrow n1[v/x] \quad (case in2v of in1x \Rightarrow n1 | in2y \Rightarrow n2) \rightarrow n2[v/y]

\[
[|\psi\rangle, \ell, \text{let } x \otimes y = v \otimes w \text{ in } r] \rightarrow [|\psi\rangle, \ell, r[v/x, w/y]]
\]

\[
[|\psi\rangle, \ell, \text{case in1v of } \text{in1x} \Rightarrow r_1 | \text{in2y} \Rightarrow r_2] \rightarrow [|\psi\rangle, \ell, r_1[v/x]]
\]

\[
[|\psi\rangle, \ell, \text{case in2v of } \text{in1x} \Rightarrow r_1 | \text{in2y} \Rightarrow r_2] \rightarrow [|\psi\rangle, \ell, r_2[v/y]]
\]

\[
[|\psi\rangle, \ell, \text{unfold fold } v] \rightarrow [|\psi\rangle, \ell, v]
\]

\[
[|\psi\rangle, \ell, s; r] \rightarrow [|\psi\rangle, \ell, r]
\]

\[
[|\psi\rangle, \ell, U(x_1 \otimes \cdots \otimes x_k)] \rightarrow [(\sigma \circ (U \otimes \text{id}) \circ \sigma^{-1})(|\psi\rangle, \ell, x_1 \otimes \cdots \otimes x_k)],
\]

for any permutation \(\sigma\), s.t. \(\sigma(i) = \ell(x_i), 1 \leq i \leq k\).

\[
\frac{m \xrightarrow{P} m'}{E[m] \xrightarrow{P} E[m']}
\]

\[
\frac{[|\psi\rangle, \ell, q]}{[|\psi\rangle, \ell, q, \text{new ff}]} \rightarrow [|\psi\rangle \otimes |0\rangle, \{x \mapsto \text{dim}(|\psi\rangle) + 1\}, x]
\]

\[
\frac{[|\psi\rangle, \ell, \text{new tt}]}{[|\psi\rangle, \ell, |1\rangle, \{x \mapsto \text{dim}(|\psi\rangle) + 1\}, x],}
\]

where \(x\) is chosen fresh.

\[
\left(\sum_i \alpha_i |b_i]\otimes [0]\otimes [b_i']\right) + \beta \left(\sum_i \beta_i |b_i] \otimes [1] \otimes [b_i']\right), \{y \mapsto j\}, \text{meas } y \rightarrow \left[\sum_i \alpha_i |b_i]\otimes [b_i']\right], \varnothing, \text{ff}
\]

\[
\left(\sum_i \alpha_i |b_i]\otimes [0] \otimes [b_i']\right) + \beta \left(\sum_i \beta_i |b_i] \otimes [1] \otimes [b_i']\right), \{y \mapsto j\}, \text{meas } y \rightarrow \left[\sum_i \beta_i |b_i]\otimes [b_i']\right], \varnothing, \text{tt}
\]

where \(\text{dim}(|b_i]) = j - 1\), so that the \(j\)-th qubit is measured.

Fig. 9. Rules for manipulating quantum information.

\[
[|\psi\rangle, \ell, (\lambda(x_1, \ldots, x_n). q)(v_1 \otimes \cdots \otimes v_n)] \rightarrow [|\psi\rangle, \ell, q[v_1/x_1, \ldots, v_n/x_n]]
\]

\[
\frac{\text{run } [|\psi\rangle, \varnothing, v]}{v}
\]

\[
\frac{m \xrightarrow{P} m'}{[|\psi\rangle, \ell, m] \xrightarrow{P} [|\psi\rangle, \ell, m']}
\]

\[
\frac{[|\psi\rangle, \ell, q]}{[|\psi\rangle, \ell, q'] \xrightarrow{P} \text{run } [|\psi\rangle, \ell', q']}
\]

\[
\frac{m \xrightarrow{P} m'}{[|\psi\rangle, \ell, q, \text{init } m] \xrightarrow{P} [|\psi\rangle, \varnothing, \text{init } m']}
\]

Fig. 10. Rules for quantum function application and extracting observable (quantum) information.
The observable quantum/classical values are simply quantum/classical values of observable types with observable context. The closed observable quantum values are in 1-1 correspondence with the closed observable classical values, which is made precise by the assignment $| - |$ from Figure 6. Therefore $| - |$ may be seen as a translation, not only between the observable types, but also between their closed values, and so we may think of them as carrying the same information. In the sequel, we see that this view extends to our denotational interpretation as well. Observable types and values are important, because they play a special role in the terms we introduce next.

Given any configuration $C$ of observable type, the term “run $C$” reduces the configuration $C$ to some value configuration $[[\psi], \emptyset, v]$, then extracts the observable quantum value $v$ from it and produces its classical counterpart $[v]$ as the result of the overall computation. Note that in this situation, the observable value $v$ does not depend on the quantum data, because it necessarily has an empty quantum context and thus empty linking function, and therefore the remaining quantum state $|\psi\rangle$ may be safely discarded (this is consistent with affine approaches to quantum programming, see [Clairambault et al. 2019; Péchoux et al. 2020b; Selinger 2004a]). The term “run $C$” is classical, and it allows us to execute quantum algorithms on a quantum computer and then extract the resulting observable information into our classical subsystem for further manipulation. The entire process is probabilistic. It is convenient to introduce some syntactic sugar. Given a closed quantum term $\cdot : \cdot \triangleright q : O$ of observable type, we can define run $q \triangleq$ run $[1, \emptyset, q]$. Users of the programming language are not expected to write the more general terms “run $C$” (which are useful for formalising the operational and denotational semantics), but only the sugarised terms “run $q$”. The term “init $m$” performs the reverse function of that of “run”, i.e., given a classical (probabilistic) process $m$ of observable type, the term “init $m$” prepares observable quantum information as indicated by the observable value that $m$ reduces to in the end.

Finally, the “let $x = \text{lift } q \text{ in } r$” term allows us to execute a quantum term $q$ of observable type and then promote the observable quantum information it produces to the classical world, so that we may use it any number of times within the continuation $r$ (see Figure 8). This term therefore implements what is often called “dynamic lifting” in the quantum programming literature. From a structural perspective, it is the only term that allows us to modify the non-linear context of quantum terms and as such may be compared to the corresponding rules of the LNL calculus [Benton 1995; Benton and Wadler 1996]. In practice, this term is useful for describing quantum processes where we measure a part of our quantum state and use the measurement outcome to influence the subsequent quantum dynamics. This is necessary for protocols like quantum teleportation.

### 2.3 Type Safety

The next two propositions show VQPL is type-safe. The first proposition shows type assignment is preserved by reduction, and as a consequence, totality of quantum configurations also is preserved.

**Proposition 2.3.1** (Type Preservation). If $\Phi \vdash m : P$ and $m \xrightarrow{p} n$, then $\Phi \vdash n : P$. Likewise, if $\Phi \vdash C : A; \text{qbit}^k$ and $C \xrightarrow{p} D$, then $\Phi \vdash D : A; \text{qbit}^k$. In both of these situations, if $p < 1$, then there exists a term $n'$ (resp. configuration $D'$), such that $m \xrightarrow{1-p} n'$ (resp. $C \xrightarrow{1-p} D'$).

**Proposition 2.3.2** (Progress). If $\cdot \vdash m : P$, then either $m$ is a value or there exists a classical term $n$, such that $m \xrightarrow{p} n$ for some $p \in (0, 1]$. Likewise, if $\cdot \vdash C : A; \text{qbit}^k$, then either $C$ is a value configuration or there exists a quantum configuration $D$, such that $C \xrightarrow{p} D$ for some $p \in (0, 1]$.

**Remark 2.3.3.** As usual, Progress holds for all closed terms/configurations, whereas Type Preservation holds for all well-formed terms/configurations, including the open ones. Here we note that the term formation rules are independent of the translation $| - |$ on values. This translation only
matters for three rules in our operational semantics and it is restricted to closed observable values (in Figure 6), because otherwise Type Preservation would fail for open terms/Configurations.

2.4 Recursion and Asymptotic Behaviour of Reduction

It is well-known that type recursion induces term recursion in FPC [Abadi and Fiore 1996; Fiore 1994; Harper 2016], and this also is true for VQPL. The call-by-value fixpoint operator

\[ \cdot + f x_{P \rightarrow R} : ((P \rightarrow R) \rightarrow P \rightarrow R) \rightarrow P \rightarrow R \]

may be derived at any function type \( P \rightarrow R \) (see [Abadi and Fiore 1996] and [Fiore 1994, §8] for more details). This fixpoint operator allows us to write recursive functions.

The probability that a term \( m \) reduces to a value \( v \) (in any number of steps) may be determined as in [Pagani et al. 2014]. The probability weight of a reduction path \( \pi = (m_1 \xrightarrow{p_1} \cdots \xrightarrow{p_n} m_n) \) is

\[ P(\pi) = \prod_{i=1}^{n} p_i. \]

The probability that \( m \) reduces to the value \( v \) in at most \( n \) steps is

\[ P(m \rightarrow_{\leq n} v) \stackrel{\text{def}}{=} \sum_{\pi \in \text{Paths}_{\leq n}(m,v)} P(\pi), \]

where \( \text{Paths}_{\leq n}(m,v) \) is the set of all reduction paths from \( m \) to \( v \) whose length is at most \( n \). The probability that \( m \) reduces to the value \( v \) in any number of steps is \( P(m \rightarrow_{\ast} v) \stackrel{\text{def}}{=} \sup_{n} P(m \rightarrow_{\leq n} v) \).

Similarly, the probability that a quantum configuration \( C \) reduces to a value configuration \( V \) (in any number of steps) is denoted \( P(C \rightarrow_{\ast} V) \); it is determined in exactly the same way as above by substituting the notion of term with that of configuration. Finally, the overall probability that a term \( m \) or configuration \( C \) terminates is given by:

\[ \text{Halt}(m) \stackrel{\text{def}}{=} \sum_{v \in \text{Val}(m)} P(m \rightarrow_{\ast} v) \quad \text{Halt}(C) \stackrel{\text{def}}{=} \sum_{V \in \text{Val}(C)} P(C \rightarrow_{\ast} V) \]

where \( \text{Val}(C) \stackrel{\text{def}}{=} \{ V \mid V \text{ is a value configuration and } P(C \rightarrow_{\ast} V) > 0 \} \) and \( \text{Val}(m) \stackrel{\text{def}}{=} \{ v \mid v \text{ is a value and } P(m \rightarrow_{\ast} v) > 0 \} \). Note that both sums may be countably infinite.

2.5 Examples

We now illustrate VQPL with some example programs.

Example 2.5.1. A fair coin toss can be defined by using some simple quantum resources: \( \text{coin}_{0.5} \stackrel{\text{def}}{=} \text{run } (\text{meas}(H(\text{new ff}))) : \text{Bool} \), where \( H \) represents the Hadamard unitary operation. More generally, by replacing \( H \) with a suitable unitary operation \( U_{p} \), a biased coin toss \( \text{coin}_{p} \) may be defined for \( p \in [0, 1] \) with reduction behaviour \( P(\text{coin}_{p} \rightarrow_{\ast} \text{ff}) = p \) and \( P(\text{coin}_{p} \rightarrow_{\ast} \text{tt}) = 1 - p \). Notice that \( \text{coin}_{p} : \text{Bool} \) is a classical term. This shows that classical discrete probabilistic choice is derivable and therefore the language PFPC [Jia et al. 2021b] is a subsystem of VQPL.

Example 2.5.2. Unitary operations controlled by a qubit (e.g. CNOT) may be implemented as usual, but we can also represent quantum operations that depend on classical data. For example:

\[ \text{if } \cdot H \stackrel{\text{def}}{=} \left( \lambda b : \text{ Bool} \cdot \text{if } (b) \text{ then } H \text{ else } \lambda (q).q \right) : \text{Bool} \rightarrow Q(\text{qbit}, \text{qbit}), \]

where we used the obvious syntactic sugar for the classical "if-then-else" statement. This is a mixed classical/quantum function which returns the Hadamard unitary operation if the input \( b \) is \text{tt} and returns the identity unitary operation otherwise. Such functions are often useful when performing unitary corrections that depend on the result of a quantum measurement (e.g. quantum teleportation [Bennett et al. 1993]).
Example 2.5.3. In VQPL we may also define recursive mixed classical/quantum functions. Therefore, this allows us to represent families of quantum operations that are parameterised by some classical data. Consider the function \( Un : \text{Nat} \rightarrow Q(\text{qbit}, \text{qbit}) \) defined by:

\[
Un \overset{\text{def}}{=} \text{fix}_{\text{Nat}} \rightarrow Q(\text{qbit}, \text{qbit}) \lambda f_{\text{Nat}} \rightarrow Q(\text{qbit}, \text{qbit}). \lambda n_{\text{Nat}}. \text{case } n \text{ of } 0 \Rightarrow \lambda (q). \ q \ | \ S(m) \Rightarrow \lambda (q). \ U((f m) q),
\]

where \( U : \mathbb{C}^2 \rightarrow \mathbb{C}^2 \) is some fixed single-qubit unitary operation and where we use some obvious syntactic sugar again. The recursive call is performed by the "\( f m \)" expression. This function represents the quantum unitary operation \( U^n : \mathbb{C}^2 \rightarrow \mathbb{C}^2 \). More precisely, given a value \( k : \text{Nat} \) and a qubit variable \( q \) together with a quantum configuration \( [\ket{\psi}, \ell, (Un k) q] \), with \( \ell(q) = 1 \), then the reduction is completely deterministic and this configuration will reduce to configuration \( [U^k|\psi\rangle, \ell, q] \) with probability 1, i.e.:

\[
P([|\psi\rangle, \ell, (Un k) q] \rightarrow, [U^k|\psi\rangle, \ell, q]) = 1.
\]

This shows that classical recursion induces recursion on the quantum subsystem.

Example 2.5.4. The fixpoint operator \( \text{fix} \) also allows us to write quantum recursive functions. The function \( Ts \) below applies the unitary \( T \) to each qubit in a list. Defining, for brevity, \( Fqs \overset{\text{def}}{=} Q(\text{List}(\text{qbit}), \text{List}(\text{qbit})) \), let:

\[
Ts \overset{\text{def}}{=} (\text{fix}_{\text{Nat}} \rightarrow Fqs . \lambda f_{\text{Nat}} \rightarrow Fqs . \lambda x_{\text{Nat}}. \lambda (qs). \text{case}(qs) \text{ of } \text{nil} \Rightarrow \text{nil} | q :: qs' \Rightarrow Tq :: (fx)qs') : 1 \rightarrow Fqs,
\]

where we used some (hopefully obvious) syntactic sugar for pattern matching of (linear) lists. The recursive call is performed by the \( fx \) expression, which is of type \( Fqs \). Setting \( Ts \overset{\text{def}}{=} Ts'() : Q(\text{List}(\text{qbit}), \text{List}(\text{qbit})) \), we get the desired function. An example quantum execution is given by \( P([[[111]], \ell, Ts(x_1 :: x_2 :: x_3 :: \text{nil})] \rightarrow [(T \otimes T \otimes T)[111], \ell, x_1 :: x_2 :: x_3 :: \text{nil}]) = 1 \), where \( \ell \) is the linking function defined by \( \ell(x_i) = i \).

Example 2.5.5. Repeat-until-success (RUS) \cite{Paetz} is a technique for implementing quantum unitary operators by using quantum measurements in a careful way. The main benefit in using RUS over other techniques that synthesise unitary operators, is that RUS may require fewer applications of the \( T \) gate, which is considered expensive in terms of resources needed for error correction.

For brevity, we consider the simplest case, where we wish to realise a fixed single-qubit unitary operator \( U : \mathbb{C}^2 \rightarrow \mathbb{C}^2 \). The RUS algorithm is described as follows. Given an input qubit \( \ket{\psi} \), then: (1) prepare a new qubit in state \( |0\rangle \), so the overall state is \( |0\rangle \otimes |\psi\rangle \); (2) apply a two-qubit unitary operator \( U' \) that has been previously determined based on \( U \); (3) perform a measurement in the computational basis on the first qubit only; (4) if the measurement outcome is 0 (with some probability \( p \)), then the final state is \( U|\psi\rangle \) and the algorithm terminates; otherwise the resulting state is \( E|\psi\rangle \), where \( E \) is some other unitary operator (chosen in advance), so we apply \( E^\dagger \) to this state and we go back to step (1). The unitary operators \( U' \) and \( E \) must be chosen (in advance) so that the probability \( p \) of success (i.e., termination in step (4)) is high. One can prove this program always terminates in state \( U|\psi\rangle \), as required, but the expected number of loops depends on \( p \).

We can implement this in VQPL by taking a term \( U_{\text{RUS}}' : 1 \rightarrow Q(\text{qbit}, \text{qbit}) \) which is defined as:

\[
U_{\text{RUS}}' \overset{\text{def}}{=} \text{fix}_{x_1 \rightarrow Q(\text{qbit}, \text{qbit})} \lambda f_{\text{Nat}} \rightarrow Q(\text{qbit}, \text{qbit}). \lambda m. \lambda (q). \text{case}(m) \text{ of } \text{ff} \Rightarrow q_2 | \text{tt} \Rightarrow (fx)(E^\dagger q_2),
\]

where we used some obvious syntactic sugar for pattern matching of (linear) bits. The recursive call is again performed by the \( fx \) expression. Setting \( U_{\text{RUS}} \overset{\text{def}}{=} U_{\text{RUS}}'() : Q(\text{qbit}, \text{qbit}) \), we get the desired quantum function.
Example 2.5.6. In this example we consider the Quantum Approximate Optimisation Algorithm (QAOA) [Farhi et al. 2014], which is a variational quantum algorithm. In particular, we consider QAOA when applied to the MAXCUT problem. The optimisation problem is the following: given a simple undirected graph $G$ (no parallel edges, no self-loops), find a maximum cut of $G$. This problem is NP-hard and the goal of the QAOA algorithm (applied to the MAXCUT instance) is to identify a cut of the graph whose size is as close to the largest one as possible. It is possible to implement the entire algorithm in our language, but due to space restrictions, we only provide a sketch of the crucial steps and focus on the quantum-classical interactions.

We assume we have defined a type $\text{Graph}$, which has sufficient structure to represent simple undirected graphs. QAOA is parameterised by a choice of natural number $p \in \mathbb{N}$ and two $p$-tuples of rotation angles $(\alpha_1, \ldots, \alpha_p) \in [0, 2\pi)^p$ and $(\beta_1, \ldots, \beta_p) \in [0, 2\pi)^p$ that are used to determine some of the unitary operators used by QAOA. We can represent these tuples as lists of length $p$ of type $\mathbb{R}$, where we assume some type $\mathbb{R}$ to approximate real numbers (e.g. take the rational numbers with the obvious representations as pairs of $\mathbb{N}$). Given an input graph $G$ and these rotation parameters, we have to construct a unitary operator acting on $n$ qubits, where $n$ is the number of vertices of the graph $G$. This can be achieved by implementing a (recursive) function

$$\text{unitary} : \text{Graph} \rightarrow \text{List}(\mathbb{R}) \rightarrow \text{List}(\mathbb{R}) \rightarrow Q(\text{List}(\text{qbit}), \text{List}(\text{qbit}))$$

A single run of the QAOA algorithm will simply apply this unitary operator to the initial state $|+\rangle^\otimes n$ and then measures all qubits at the end. The measurement outcome is then a list of type $\text{bit}$ of length $n$ that determines a cut of the graph $G$ (each value of type $\text{bit}$ identifies to which partition of the cut the vertex is assigned). Using the function unitary, this is easy to program and we obtain the function

$$\text{qaoa} : \text{Graph} \rightarrow \text{List}(\mathbb{R}) \rightarrow \text{List}(\mathbb{R}) \rightarrow Q(I, \text{List}(\text{bit}))$$

which describes the entire quantum part of the algorithm. However, QAOA is a variational quantum algorithm and the hardest part of the algorithm is to select appropriate rotation parameters that would produce a near-optimal cut with high probability. This is achieved by repeatedly executing the quantum part of the algorithm (which we already covered) and then using a classical optimisation procedure that determines the set of rotation parameters to use for the next iteration of the algorithm. This process is repeated until some satisfactory termination condition is satisfied. To represent this, we can define

$$\text{runQAOA} \overset{\text{def}}{=} \lambda g. \lambda l. \lambda l'. \text{run} ((\text{qaoa } g \ l \ l')^*) : \text{Graph} \rightarrow \text{List}(\mathbb{R}) \rightarrow \text{List}(\mathbb{R}) \rightarrow \text{List}(\text{Bool})$$

to represent a single execution of the quantum part of the algorithm. Note that the result, after using the "run" term, is now a classical list of type $\text{Bool}$, which identifies a cut of the input graph. Finally, a classical function

$$\text{solve} : \text{Graph} \rightarrow \text{List}(\text{Bool})$$

may be defined which: (1) selects the initial values of the rotation parameters; (2) repeatedly calls runQAOA and uses the observed measurement outcomes and the previously selected rotation parameters in order to compute new rotation parameters for the next recursive call; and (3) this recursive process terminates when the desired termination condition is met.

This short sketch shows that our language has sufficient structure to represent variational quantum algorithms, where there is a clear separation between the quantum and classical parts of the computation, which is also clearly reflected in $\text{VQPL}$.
3 PROBABILISTIC EFFECTS AND (COMMUTATIVE) VALUATIONS MONADS

As we already explained, PFPC is a subsystem of our language. In this section we recall the construction of the commutative monad \( M : \text{DCPO} \rightarrow \text{DCPO} \) of [Jia et al. 2021b], which we also use in our denotational semantics. Our classical judgements are interpreted in the Kleisli category \( \text{DCPO}_M \) of \( M \), which provides a sound and (strongly) adequate model of PFPC [Jia et al. 2021b].

3.1 Domain-Theoretic and Topological Preliminaries

If \( D \) is a partially ordered set (poset), a nonempty subset \( A \) of \( D \) is called directed if each pair of elements in \( A \) has an upper bound in \( A \). Then \( D \) is a directed complete poset (dcpo, for short) if each of its directed subsets has a supremum. For example, the unit interval \([0, 1]\) is a dcpo in the usual ordering. A Scott-continuous map \( f : D \rightarrow E \) between (posets) dcpo’s is a function that is monotone and preserves (existing) suprema of directed subsets. Pointed posets have least elements, usually denoted by \( \bot_D \) if \( D \) is the ambient poset. A Scott-continuous function \( f : D \rightarrow E \) between pointed dcpo’s is strict if \( f \) preserves the least element, that is, \( f(\bot_D) = \bot_E \).

The category \( \text{DCPO} \) of dcpo’s and Scott-continuous functions is complete, cocomplete and Cartesian closed [Abramsky and Jung 1994]. The categorical (co)product of the dcpo’s \( A_1 \) and \( A_2 \) is denoted \( A_1 \times A_2 \) (\( A_1 + A_2 \)), with \( \pi_1, \pi_2 \) (\( in_1, in_2 \)) the associated (co)projections. Initial and terminal objects of \( \text{DCPO} \) are denoted by \( \emptyset \) and \( 1 \), which are the empty dcpo and the singleton dcpo, respectively. For dcpo’s \( A \) and \( B \), the internal hom of \( A \) and \( B \) in \( \text{DCPO} \) is \([A \rightarrow B]\), the space of all Scott-continuous functions \( f : A \rightarrow B \) ordered pointwise.

The category \( \text{DCPO}_\perp \) of pointed dcpo’s and strict Scott-continuous functions is symmetric monoidal closed (with respect to the strict Scott-continuous function space), and it also is complete and cocomplete [Abramsky and Jung 1994].

The Scott topology \( \sigma D \) on a dcpo \( D \) consists of the upper subsets \( U = \uparrow U \coloneqq \{ x \in D \mid (\exists u \in U) u \leq x \} \) that are inaccessible by directed suprema: i.e., if \( A \subseteq D \) is directed and \( \sup A \in U \), then \( A \cap U \neq \emptyset \). The topological space \( (D, \sigma D) \) is also written as \( \Sigma D \). Scott-continuous functions between dcpo’s \( D \) and \( E \) are exactly the continuous functions between \( \Sigma D \) and \( \Sigma E \) [Gierz et al. 2003, Proposition II-2.1]. We always equip \([0, 1]\) with the Scott topology unless stated otherwise.

If \( B \subseteq D \) and \( D \) is a dcpo, then \( B \) is a sub-dcpo if every directed subset \( A \subseteq B \) satisfies \( \sup_D A \in B \), where \( \sup_D A \) denotes the supremum of \( A \) in \( D \). In this case, \( B \) is a dcpo in the induced order from \( D \) and \( \sup_D A = \sup_B A \) for each directed subset \( A \) of \( B \).

3.2 The Monad \( M \)

If \( X \) is a topological space, then the open set lattice \( OX \) is a complete lattice in the inclusion order, hence a dcpo. A subprobability valuation on \( X \) is a Scott-continuous function \( \nu : OX \rightarrow [0, 1] \) that is strict (\( \nu(\emptyset) = 0 \)) and modular (\( \nu(U) + \nu(V) = \nu(U \cup V) + \nu(U \cap V) \)). The set \( \mathcal{V}X \) of subprobability valuations on \( X \) is a dcpo in the stochastic order defined by: \( \nu_1 \leq \nu_2 \) if and only if \( \nu_1(U) \leq \nu_2(U) \) for all \( U \in OX \), for \( \nu_1, \nu_2 \in \mathcal{V}X \), and the supremum of a directed family of valuations \( \{\nu_i\}_{i \in I} \) is computed pointwise: \( (\sup_{i \in I} \nu_i)(U) \coloneqq \sup_{i \in I} \nu_i(U) \), for all \( U \in OX \). The least element of \( \mathcal{V}X \) is the constantly zero valuation \( \delta_X \). The dcpo \( \mathcal{V}X \) also enjoys a convex structure: if \( \nu_i \in \mathcal{V}X \) and \( r_i \geq 0 \) for \( i = 1, \ldots, n \) and \( \sum_{i=1}^n r_i \leq 1 \), then the convex sum \( \sum_{i=1}^n r_i \nu_i \), defined by \( (\sum_{i=1}^n r_i \nu_i)(U) \coloneqq \sum_{i=1}^n r_i \nu_i(U) \) for \( U \in OX \), also is in \( \mathcal{V}X \).

The Dirac valuations \( \delta_x \), for \( x \in X \), are defined by \( \delta_x(U) = 1 \) if \( x \in U \) and \( \delta_x(U) = 0 \) otherwise. These are canonical examples of subprobability valuations, as are their convex sums, which we call simple valuations: they have form \( \sum_{i=1}^n r_i \delta_{x_i} \), where \( x_i \in X \), \( r_i \geq 0 \), and \( \sum_{i=1}^n r_i \leq 1 \). The simple valuations are denoted \( SX \), and \( SX \subseteq \mathcal{V}X \), but \( SX \) is not a dcpo in general.
If \( f : X \to [0,1] \) is a continuous function and \( v \in \mathcal{V}X \), the integral of \( f \) against \( v \) is given by the Choquet formula
\[
\int_{x \in X} f(x) dv \overset{\text{def}}{=} \int_0^1 v(f^{-1}((t,1])) dt,
\]
where the right side is a Riemann integral of the bounded antitone function \( t \mapsto v(f^{-1}((t,1])) \).
Since \( f \) is continuous, and \( (t,1] \subseteq [0,1] \) is Scott open for each \( t \in [0,1] \), \( f^{-1}((t,1]) \) is open in \( X \), so \( v(f^{-1}((t,1])) \) is well defined. Thus the integral makes sense. If no confusion can occur, we simply write \( \int_{x \in X} f(x) dv \) as \( \int f dv \). Note that if \( f : X \to [0,1] \) is fixed, then the map \( v \mapsto \int f dv : \mathcal{V}X \to [0,1] \) is Scott-continuous. Other basic properties of this integral can be found in [Jones 1990].

If \( D \) is a dcpo, then \( \mathcal{V}D \overset{\text{def}}{=} \mathcal{V} \Sigma D = \mathcal{V}(D, \sigma(D)) \) is well-defined, and [Jones 1990] proved that \( \mathcal{V} \) extends to a monad on DCPO.

3.2.1 Monad Structure. The unit of \( \mathcal{V} \) at \( D \) is \( \eta^X_D : D \to \mathcal{V}D : x \mapsto \delta_x \). The Kleisli extension \( f^\dagger \) of a Scott-continuous map \( f : D \to \mathcal{V}E \) maps \( v \in \mathcal{V}D \) to \( f^\dagger(v) \in \mathcal{V}E \), where for \( U \in \sigma E \), \( f^\dagger(v)(U) \overset{\text{def}}{=} \int_{x \in D} f(x)(U) dv \). The multiplication \( \mu^X_D : \mathcal{V} \mathcal{V}D \to \mathcal{V}D \) is given by \( \mu \overset{\text{def}}{=} \text{id} \circ \mathcal{V} \sigma D \). Thus, \( \mathcal{V} \) defines an endofunctor on DCPO that sends a dcpo \( D \) to \( \mathcal{V}D \), and a Scott-continuous map \( h : D \to E \) to \( \mathcal{V}(h) \overset{\text{def}}{=} (\eta^E \circ h)^\dagger \), so \( \mathcal{V}(h)(v)(U) = v(h^{-1}(U)) \) for \( v \in \mathcal{V}D \) and \( U \in \sigma E \). The valuation \( h_v(v) = \mathcal{V}(h)(v) \) is called the push forward of \( v \) by \( h \).

In fact, \( \mathcal{V} \) defines a strong monad on DCPO [Jones 1990]: the strength at \( (D,E) \) is
\[
\tau^\mathcal{V}_{DE} : D \times \mathcal{V}E \to \mathcal{V}(D \times E) : (x,v) \mapsto \lambda U. \int_{y \in E} \chi_U(x,y) dv,
\]
where \( \chi_U \) is the characteristic function of \( U \in \sigma(D \times E) \). However, it is unknown whether \( \mathcal{V} \) is a commutative monad on DCPO. This is equivalent to showing the Fubini-style equation
\[
\int_{x \in D} \int_{y \in E} \chi_U(x,y) d\xi dv = \int_{y \in E} \int_{x \in D} \chi_U(x,y) dv d\xi,
\]
holds for dcpo’s \( D \) and \( E \), where \( U \in \sigma(D \times E) \) and \( v \in \mathcal{V}D, \xi \in \mathcal{V}E \) [Jones and Plotkin 1989]. This problem is addressed in [Jia et al. 2021b], where we define a subclass of valuations that simultaneously validates (3.1), contains all simple valuations, and forms a dcpo in the stochastic order. We showed that this construction defines a commutative valuations monad \( M \) on DCPO and we devote the rest of this subsection to summarising the construction of the monad.

Definition 3.2.1. For a dcpo \( D \), \( MD \) is the intersection of all sub-dcpo’s of \( \mathcal{V}D \) containing \( SD \).

We call the valuations in \( MD \) the \( M \)-valuations on \( D \). In fact, \( MD \) is the smallest sub-dcpo of \( \mathcal{V}D \) containing \( SD \). It follows that the \( M \)-valuations on \( D \), directed suprema of simple valuations on \( D \), directed suprema of directed suprema of simple valuations on \( D \) and so forth, transfinitely. It is straightforward to show that (3.1) holds when \( \xi \) and \( v \) are simple valuations, and because the nested integral operations are Scott-continuous in the valuations components, it follows (3.1) holds for \( M \)-valuations. This is the idea behind the proof of the following theorem.

Theorem 3.2.2 ([Jia et al. 2021b, Theorem 8]). \( M \) has the structure of a commutative monad on DCPO when equipped with the (co)restricted monad operations of \( \mathcal{V} \).

Since the inclusions \( MD \subseteq \mathcal{V}D \) form a strong map of monads, we are justified in viewing \( M \) as a submonad of \( \mathcal{V} \) and we use the same notation for the monad operations of \( M \) and \( \mathcal{V} \).
4 QUANTUM EFFECTS AND HEREDITARILY ATOMIC VON NEUMANN ALGEBRAS

We now turn our attention to the model for the quantum subsystem of VQPL. We begin with a short review of operator algebras, which can be used to study quantum foundations. The standard references are [Blackadar 2006], [Takesaki 2000] and [Kadison and Ringrose 1997]. For quantum computing, it is sufficient to consider a special class of operator algebras that is known as the hereditarily atomic von Neumann algebras. This class consists of (possibly infinite) products of finite-dimensional matrix algebras. These algebras are studied in [Kornell 2020]. Our main result in this section proves this category of hereditarily atomic von Neumann algebras is enriched over continuous domains (§4.3), which is crucial for providing a semantic interpretation of the "mq" term.

4.1 Definition of von Neumann Algebras

If $H$ is a Hilbert space, a linear map $x : H \to H$ is called an operator, and $x$ is bounded if it is continuous with respect to the norm $\| \cdot \|$ induced by the inner product $\langle \cdot, \cdot \rangle$ on $H$. The space $B(H)$ of all bounded operators on $H$ forms an algebra over $\mathbb{C}$ with composition as multiplication. Moreover, $B(H)$ has an involution $x \mapsto x^*$, where $x^*$ is the unique bounded operator satisfying $\langle x^* k, h \rangle = \langle k, xh \rangle$ for $h, k \in H$. A subalgebra $A \subseteq B(H)$ that is closed under the involution is called a $*$-subalgebra. If, in addition, $xy = yx$ for each $x, y \in A$, we call $A$ commutative. The commutant of a subset $S \subseteq B(H)$ is $S' = \{ y \in B(H) \mid xy = yx \ (\forall x \in S) \}$.

Definition 4.1.1. Let $H$ be a Hilbert space. A von Neumann algebra on $H$ is a $*$-subalgebra $M$ of $B(H)$ such that $M'' = M$. If $K$ is another Hilbert space, and $N$ is a von Neumann algebra on $K$, a linear map $\varphi : M \to N$ that preserves the multiplication and the involution is a $*$-homomorphism. If, in addition, $\varphi$ is bijective, it is a $*$-isomorphism.

The commutant of any non-empty set in $B(H)$ always contains $1_H$, so $1_H \in M$ for any von Neumann algebra $M \subseteq B(H)$. We sometimes write $1_M$ as $1_H$ to emphasize it is the unit of $M$.

Example 4.1.2. $B(H)$ itself is a von Neumann algebra, and if $H$ is $n$-dimensional, then $B(H)$ is $*$-isomorphic to $M_n(\mathbb{C})$, the algebra of $n \times n$-complex valued matrices. This example plays an important role in the definition of hereditarily atomic von Neumann algebras below.

Example 4.1.3. If $X$ is a set, then $\ell^2(X) \overset{\text{def}}{=} \{ f : X \to \mathbb{C} \mid \sum_{x \in X} |f(x)|^2 < \infty \}$ is a Hilbert space with inner product $\langle f, g \rangle \overset{\text{def}}{=} \sum_{x \in X} \overline{f(x)} g(x)$. The space $\ell^\infty(X) \overset{\text{def}}{=} \{ f : X \to \mathbb{C} \mid \sup_{x \in X} |f(x)| < \infty \}$ equipped with the norm $\| f \| \overset{\text{def}}{=} \sup_{x \in X} |f(x)|$ can be embedded isometrically into $B(\ell^2(X))$ via the maps $f \mapsto m_f$, where $m_f : \ell^2(X) \to \ell^2(X)$ is the left multiplication $g \mapsto fg$ [Landsman 2017, Proposition B.73]. Thus, identifying $\ell^\infty(X)$ with its image in $B(\ell^2(X))$ shows it is a commutative von Neumann algebra on $\ell^2(X)$ [Landsman 2017, Proposition B.108].

Given Hilbert spaces $(H_\alpha)_{\alpha \in A}$, the sum $\bigoplus_{\alpha \in A} H_\alpha \overset{\text{def}}{=} \{ (h_\alpha)_{\alpha \in A} \in \prod_{\alpha \in A} H_\alpha \mid \sum_{\alpha \in A} |h_\alpha|^2 < \infty \}$ is a Hilbert space with inner product $\langle k, h \rangle \overset{\text{def}}{=} \sum_{\alpha \in A} \langle k_\alpha, h_\alpha \rangle$ for $k = (k_\alpha)_{\alpha \in A}$ and $h = (h_\alpha)_{\alpha \in A}$.

Proposition 4.1.4. [Takesaki 2000, Proposition II.3.3] Let $M_\alpha$ be a von Neumann algebra on a Hilbert space $H_\alpha$ for each $\alpha \in A$. Then $\prod_{\alpha \in A} M_\alpha \overset{\text{def}}{=} \{ (x_\alpha)_{\alpha \in A} \mid \sup_{\alpha \in A} |x_\alpha| < \infty \}$ is a von Neumann algebra on $H = \bigoplus_{\alpha \in A} H_\alpha$, where $xh \overset{\text{def}}{=} (x_\alpha h_\alpha)_{\alpha \in A} \in H$ for $x = (x_\alpha)_{\alpha \in A} \in \prod_{\alpha \in A} M_\alpha$ and $h = (h_\alpha)_{\alpha \in A} \in H$.

Definition 4.1.5. We call a von Neumann algebra $M$ hereditarily atomic, or simply an HA-algebra, if $M$ is isomorphic to $\prod_{\alpha \in A} M_\alpha$, where each $M_\alpha$ is $*$-isomorphic to some matrix algebra.
In order to make the correspondence between HA-algebras and the types of our language clearer, we overload notation and often write $\bigoplus_{a \in A} M_a \overset{\text{def}}{=} \prod_{a \in A} M_a$.

**Example 4.1.6.** All of the following are HA-algebras and we indicate to which type they correspond. The algebra $\mathbb{C}$ of complex numbers corresponds to type $1$; the algebra $\bigoplus_{n \in \mathbb{N}} \mathbb{C}$ corresponds to type $\text{qbit}$; the algebra $\bigoplus_{n \in \mathbb{N}} M_2^n(\mathbb{C})$ corresponds to type $\text{List(qbit)}$ $\equiv \mu \text{X.} X \otimes (\text{qbit} \otimes X)$. Moreover, if $X$ is a set, then $\ell^\infty(X) \equiv \bigoplus_{x \in X} \mathbb{C}$ is an HA-algebra and we will see it corresponds to observable quantum types, in general.

### 4.2 Quantum Computation with Hereditarily Atomic von Neumann Algebras

In this subsection, we define the appropriate notion of morphism that is computationally relevant.

If $M$ is a von Neumann algebra on a Hilbert space $H$, we say $x \in M$ is self adjoint if $x^* = x$, and positive if $x = y^* y$ for some $y \in M$; equivalently, $(h, x h) \geq 0$ for each $h \in H$ [Kadison and Ringrose 1997]. Given self-adjoint elements $x$ and $y$ in $M$, we write $x \leq y$ iff $y - x$ is positive. The relation $\leq$ is a partial order on the set $M_{sa}$ of self-adjoint elements in $M$ under which $M_{sa}$ is bounded directed complete: if a directed subset of $M_{sa}$ has an upper bound in $M_{sa}$, then it has a least upper bound in $M_{sa}$. The partial order $\leq$ is often called the Löwner order.

In fact, if $D \subseteq M_{sa}$ is directed and $x = \sup D$, then $x$ is the limit of $D$ in the strong operator topology on $M$, i.e., $\lim_{d \in D} dh = x h$ for each $h \in H$; this also implies convergence with respect to the weak operator topology on $M$, i.e., $\lim_{d \in D} (k, dh) = (k, x h)$ for each $h, k \in H$ [Blackadar 2006, Proposition I.3.2.5 & Corollary I.3.2.6]. As a consequence, the unit interval of $M$, $[0, 1]_M = \{x \in M_{sa} \mid 0 \leq x \leq 1 \}$ is a dcpo.

A linear function $\varphi : M \to N$ between von Neumann algebras is unital if $\varphi(1_M) = 1_N$, and subunital if $\varphi(1_M) \leq 1_N$; $\varphi$ is positive if it preserves positive elements, equivalently, if $\varphi$ is monotone with respect to $\leq$. A positive and subunital $\varphi : M \to N$ restricts to a monotone map $[0, 1]_M \to [0, 1]_N$ (which by linearity completely determines $\varphi$) and we call $\varphi$ normal if this restriction is Scott continuous.

We denote by $M_n(M)$ the von Neumann algebra of all $n \times n$-matrices with entries in $M$.

### Definition 4.2.1.

**Definition 4.2.1.** We denote the category of von Neumann algebras and NCPSU maps by $\mathbf{vN}$. Its full-on-objects subcategory having normal unital $*$-homomorphisms as morphisms is denoted by $\mathbf{vN}_*$. The category of HA-algebras and NCPSU maps is denoted by $\mathbf{HA}$ and we denote the full-on-objects subcategory of $\mathbf{HA}$ with normal unital $*$-homomorphisms by $\mathbf{HA}_*$. The categories relevant for our semantics are their formal duals given by $Q = \text{def} (\mathbf{HA})^{\text{op}}$ and $Q_* = \text{def} (\mathbf{HA}_*)^{\text{op}}$.

**Remark 4.2.2.** When working with von Neumann algebras, it is customary to adopt the Heisenberg picture of quantum mechanics, rather than the Schrödinger one. This corresponds to working in the category $\mathbf{Q}$ that is the formal dual of $\mathbf{HA}$. In fact, the program of non-commutative geometry is based on dualities between categories of operator algebras and “formal dual” categories. Furthermore, this approach has been applied in quantum programming semantics [Cho and Westerbaan 2016; Péchoux et al. 2020b] and is necessary for the appropriate categorical structure (§6.2) of our model.

---

1The definitions of $\lim_{d \in D} dh$ and of $\lim_{d \in D} (k, dh)$ implicitly use Moore-Smith Convergence; cf. [Kelley 1975].

2In this case the underlying Hilbert space of $M_n(M)$ is the $n$-fold direct sum of the underlying Hilbert space of $M$.
tr : M₁(C) → C  |  stateₚ : C → M₂(C)  |  meas : M₂(C) → C ⊗ C  |  unitary_U : M₂(C) → M₂(C)
tr : x ↔ ∑ᵢ xᵢ,i  |  stateₚ : α ↔ αₚ  |  meas : (α β) → (a β)  |  unitary_U : x ↔ UxU∗
tr⁺ : C → M₁(C)  |  stateₚ⁺ : M₂(C) → C  |  meas⁺ : C ⊗ C → M₂(C)  |  unitary_U⁺ : M₂(C) → M₂(C)
tr⁺ : α ↔ α₁M₁(C)  |  stateₚ⁺ : x ↔ tr(αp)  |  meas⁺ : (a β) → (α β)  |  unitary_U⁺ : x ↔ U∗xU

Fig. 11. Maps in the Schrödinger picture (φ : M → N) and their Hermitian adjoints (φ⁺ : N → M).

From now on, all *-homomorphisms we work with are implicitly assumed to be unital and normal. We interpret quantum values in Q⁺ and quantum terms in Q. Next, we describe maps between HA-algebras that are crucial for quantum computation and that we use in our semantics.

The maps in the upper half of Figure 11 describe NCPSU maps between HA-algebras that are well-known in the quantum computing literature. The map "tr" computes the trace of a matrix; the map "stateₚ" prepares a new (mixed) quantum state that is described by the density matrix ρ; the map "meas" performs a destructive quantum measurement on a qubit and returns a bit as outcome; the map "unitary_U" applies the unitary matrix U of arity n to an n-dimensional quantum state. These are the appropriate maps to take in the Schrödinger picture of quantum mechanics, but as explained above, the Heisenberg picture is more appropriate for our denotational semantics, so it is the Hermitian adjoints of these maps (bottom half of Figure 11) that are relevant to us. By writing φ⁺ = (φ∗)op for φ ∈ {tr, stateₚ, meas, unitary_U}, these maps are then morphisms of Q. In particular, meas⁺ : M₂(C) → C ⊗ C is the morphism of Q which represents quantum measurement.

We also define a morphism new⁺ = measop ∈ Q(C ⊗ C, M₂(C)). The way to think of this map (in Q) is that given a bit i ∈ {0, 1}, the map would prepare the density matrix |i⟩⟨i|. Indeed, notice that (in Q) we have meas⁺ = (meas ∗ meas⁺)op = id_{C⊗C}, as expected.

### 4.3 Continuous Domain Enrichment of Q

The category vN is enriched over DCPO_{⊥i} [Cho 2016], where φ ≤ ψ iff ψ − φ is completely positive, for φ, ψ ∈ vN(M, N). The DCPO_{⊥i}-enrichment of vN immediately implies that of HA and Q.

While DCPO_{⊥i}-enrichment is important, it is insufficient for our purposes. In particular, the crucial connection between the semantics of the quantum and classical probabilistic effects in our language requires Q(A, B) to be an M-algebra for A, B objects of Q (Theorem 5.3.5). We can show this is the case when Q is enriched in a much stronger sense, namely, when Q is enriched over continuous dcpo’s (see Theorem 5.3.3). Explaining all this is our next goal.

We begin with continuous dcpo’s. For x, y in a dcpo D, x is way-below y (in symbols, x ≪ y) if and only if for every directed set A with y ≤ sup A, there is some a ∈ A such that x ≤ a. A dcpo D is continuous, or simply a domain, if every element x ∈ D is the supremum of a directed set of elements that are way-below x. We use DOM to denote the category of domains and Scott-continuous maps.

It was noted in [Selinger 2004b, Example 2.7] that the unit interval of M(Cⁿ) is a continuous dcpo, from which it is easy to show [0, 1]ᵣ is a domain for every HA-algebra A. We conclude this section with a stronger result.

**Theorem 4.3.1.** The category Q is enriched over DOM.

**Proof.** The proof (see [Jia et al. 2021a]) starts with the fact that [0, 1]ᵣ is a domain for each HA-algebra A and then makes extensive use of the representation theory of von Neumann algebras. □

**Remark 4.3.2.** The hereditarily atomic property of von Neumann algebras is necessary for proving Theorem 4.3.1. By a result of Furber [Furber 2019], if the unit interval of a von Neumann algebra A is
a domain (not just dcpo), then $A$ is hereditarily atomic. Furthermore, its unit interval is isomorphic to the hom-dcpo $Q(A, C)$, so this condition is necessary for the domain enrichment of $Q$.

5 PROBABILISTIC EFFECTS, QUANTUM EFFECTS AND KEGELSPITZEN

Our language shows that quantum effects induce probabilistic effects on the classical side (via the "run" term) and, vice-versa, probabilistic effects on the classical side can also influence the quantum dynamics (via the "init" and "mq" terms). In this section, we describe the mathematical structure we use to interpret this correspondence.

In particular, we show there is a strong relationship between the Kleisli category $DCPO_M$ of $M$ (where we interpret classical programs) and our category $Q$ of hereditarily atomic von Neumann algebras (where we interpret quantum programs). The link between the two categories is provided by the theory of (continuous) Kegelspitzen [Keimel and Plotkin 2017]. The relationship we identify is crucial for the interpretation of the mixed classical/quantum judgements of Figure 7, and it links the classical theory of valuations monads to the quantum theory of von Neumann algebras.

5.1 (Continuous) Kegelspitzen

We begin by recalling the definition of Kegelspitzen [Keimel and Plotkin 2017].

**Definition 5.1.1.** A Kegelspitze is a dcpo equipped with a convex structure. More precisely:

- A barycentric algebra is a set $A$ endowed with binary operations $(a, b) \mapsto a +, b: A \times A \to A$ indexed by $r \in [0, 1]$ such that for all $a, b, c \in A$ and $r, p \in [0, 1]$, the following equations hold:

  $a +_1 b = a; \quad a +, b = b +_1 -a; \quad a + r a = a; \quad (a + r b) + r c = a +_{pr} (b +_{1-pr} c)$ provided $r, p < 1$.

- A pointed barycentric algebra is a barycentric algebra $A$ with a distinguished element $\perp$. For $a \in A$, we define scalar multiplication $r \cdot a \overset{def}{=} a + r \perp$. A map $f: A \to B$ between pointed barycentric algebras is linear if $f(\perp_A) = \perp_B$ and $f(a + r b) = f(a) + r f(b)$ for all $a, b \in A, r \in [0, 1]$.

- A Kegelspitze is a pointed barycentric algebra $K$ equipped with a directed-complete partial order such that scalar multiplication $(r, a) \mapsto r \cdot a: [0, 1] \times K \to K$ and the binary operation $(a, b) \mapsto a +, b: K \times K \to K$, for $r \in [0, 1]$, are Scott-continuous (in both arguments). A continuous Kegelspitze is a Kegelspitze that is a domain in the equipped order.

**Example 5.1.2.** For each dcpo $D$, $MD$ is a Kegelspitze: for $v_i \in MD$ and $r_i \geq 0, i = 1, \ldots, n$ with $\sum_{i=1}^n r_i \leq 1$, the convex sum $\sum_{i=1}^n r_i v_i$ is again in $MD$. Then, if $v_1, v_2 \in MD$ and $r \in [0, 1]$, we define $v_1 +_r v_2 \overset{def}{=} rv_1 + (1-r)v_2$. The zero valuation $0_D$ is the distinguished least element. If, in addition, $D$ is a domain, then $MD = \mathcal{V}D$ is a continuous Kegelspitze [Jia et al. 2021b]. For each Scott-continuous map $f: D \to E$, the map $M(f): MD \to ME$ is Scott-continuous and linear.

**Example 5.1.3.** The real unit interval $[0, 1]$ is obviously a continuous Kegelspitze. More generally, the unit interval $[0, 1]_A$ of any von Neumann algebra $A$ is a Kegelspitze. If $A$ also is hereditarily atomic, then $[0, 1]_A$ is a continuous Kegelspitze by [Selinger 2004b, Example 2.7]. Moreover, any NCPSU map $f: A \to B$ between von Neumann algebras $A$ and $B$, is Scott-continuous and linear when (co)restricted to the unit intervals of $A$ and $B$.

5.2 Correspondence between Observable Quantum/Probabilistic Effects

Our next result describes a bijective correspondence between observable quantum/probabilistic effects that allows us to interpret the terms dealing with observable primitives. A semantic observation (which we make precise later) shows that: any quantum observable type $O$ is interpreted as a commutative HA-algebra that is $*$-isomorphic to $\ell^\infty(X)$ for some set $X$; its classical observable
counterpart \(|\mathcal{O}|\) is interpreted as the \emph{discrete domain} with underlying set \(X\). Moreover, quantum values correspond to \(*\)-homomorphisms and classical values to Dirac valuations.

**Theorem 5.2.1.** Let \(X\) be an arbitrary set. Then, there exists an isomorphism of (continuous) Kegelspitzen \(r_X: \mathcal{Q}(\mathbb{C}, f^\omega(X)) \cong M(X, \sqsubseteq): r_X^{-1}\), where \(\sqsubseteq\) is the discrete order on \(X\). Furthermore, this isomorphism restricts to a 1-1 correspondence between the \(*\)-homomorphisms of \(\mathcal{Q}(\mathbb{C}, f^\omega(X))\) and the Dirac valuations of \(M(X, \sqsubseteq)\).

Combined with the above semantic observation, this theorem shows there is a 1-1 correspondence between the quantum and classical probabilistic states of \emph{observable} types, and also a 1-1 correspondence between the interpretations of quantum and classical observable values. This isomorphism is used for the interpretations of the "run" and "init" terms.

Next, we construct an isomorphism that we use for the interpretation of dynamic lifting (the "lift" term). This is similar to a construction first reported in [Rennela and Staton 2020].

**Proposition 5.2.2.** Given a dcpo \(X\), HA-algebras \(A, B\), and a discrete dcpo \(Y\), there exists a Scott-continuous and linear bijection \((-): \text{DCPO}(X \times Y, Q(A, B)) \cong \text{DCPO}(X, Q(f^\omega(Y) \otimes A, B))\), natural in all components.

### 5.3 Combining Probabilistic and Quantum Effects

In the previous subsection we considered \emph{observable} effects. In the present subsection, we show how to combine \emph{arbitrary} quantum and probabilistic effects \emph{into} quantum ones.

We begin by noting that on any Kegelspitze, the binary operations \(a \lor b\) generalize to convex sums. We then use these convex sums in order to define \emph{barycentre} maps.

**Definition 5.3.1.** In each pointed barycentric algebra \(K\), given \(a_i \in K, r_i \in [0, 1], i = 1, \ldots, n\) with \(\sum_{i=1}^n r_i \leq 1\), we inductively define the convex sum by

\[
\sum_{i=1}^n r_i a_i \defeq \begin{cases} 
  a_1, & \text{if } r_1 = 1, \\
  a_1 \lor (\sum_{i=2}^n r_i a_i), & \text{if } r_1 < 1.
\end{cases}
\]

This sum is invariant under index-permutation: for \(\pi\) a permutation of \(\{1, \ldots, n\}\), \(\sum_{i=1}^n r_i a_i = \sum_{i=1}^n r_{\pi(i)} a_{\pi(i)}\) [Jones 1990, Lemma 5.6]. If \(K\) is a Kegelspitze, then the expression \(\sum_{i=1}^n r_i a_i\) is Scott-continuous in each \(r_i\) and \(a_i\). A countable convex sum also can be defined: if \(a_i \in K\) and \(r_i \in [0, 1]\), for \(i \in \mathbb{N}\), with \(\sum_{i \in \mathbb{N}} r_i \leq 1\), define \(\sum_{i \in \mathbb{N}} r_i a_i \defeq \sup\{\sum_{j \in J} r_j a_j \mid J \subseteq \mathbb{N} \text{ and } J\text{ is finite}\}\).

**Definition 5.3.2.** Let \(K\) be a Kegelspitze and \(s = \sum_{i=1}^n r_i x_i\) be a simple valuation on \(K\). The \emph{barycentre} of \(s\) is defined as \(\beta_s(s) = \sum_{i=1}^n r_i x_i\). Furthermore, if \(K\) is a continuous Kegelspitze and \(v \in MK\), the barycentre of \(v\) is defined as \(\beta(v) \defeq \sup\{\beta_s(s) \mid s \in SK\text{ and } s \ll v\}\).

When \(K\) is a continuous Kegelspitze, the barycentre map \(\beta: MK \rightarrow K:: v \mapsto \beta(v)\) is well-defined, unique, Scott-continuous and linear [Jia et al. 2021b]. We emphasise that continuity is crucial for establishing this and it is unclear if this holds otherwise. Moreover, the following also is true.

**Theorem 5.3.3 ([Jia et al. 2021b]).** The Eilenberg-Moore category \(\text{DOM}^M\) of \(M\) over \(\text{DOM}\) is isomorphic to the category of continuous Kegelspitzen and Scott-continuous linear maps. In particular:

1. Each continuous Kegelspitze \(K\) admits a linear barycentre map \(\beta: MK \rightarrow K\) (as in Definition 5.3.2) for which the pair \((K, \beta)\) is an Eilenberg-Moore algebra of \(M\) over \(\text{DOM}\).
2. Conversely, on each \(M\)-algebra \((K, \beta)\) on \(\text{DOM}\), define \(a \lor b \defeq \beta(\delta_a \lor \delta_b)\) for \(a, b \in K\) and \(r \in [0, 1]\). Then with the operations \(\lor\), \(K\) is a continuous Kegelspitze and \(\beta: MK \rightarrow K\) is linear.
In §4 we saw that $Q$ is enriched over $\text{DOM}$. We now further strengthen that result.

**Theorem 5.3.4.** The category $Q$ is enriched over continuous Kegelspitzen in the following sense: for all objects $A, B$ in $Q$, the homset $Q(A, B)$ is a continuous Kegelspitz, and for any morphism $\varphi : A \to B$ in $Q$ and any object $C$ in $Q$, the following maps are Scott-continuous and linear:

\[
Q(C, \varphi) : Q(C, A) \to Q(C, B) :: \psi \mapsto \varphi \circ \psi \quad \text{and} \quad Q(\varphi, C) : Q(B, C) \to Q(A, C) :: \psi \mapsto \psi \circ \varphi.
\]

Combining these two theorems gives the main result of this section.

**Theorem 5.3.5.** For any HA-algebras $A$ and $B$, there exists a (unique) Scott-continuous and linear barycentre map $\beta : M Q(A, B) \to Q(A, B)$ that is also an Eilenberg-Moore algebra of $M$.

The above properties of $\beta$ are exactly what is needed to interpret the "mq" term from Figure 7, which allows us to combine classical probabilistic computation with quantum computation.

## 6 CATEGORICAL MODEL

In this section we organise the relevant mathematical data into several categories that we later use to describe our denotational semantics. A diagrammatic summary is provided in Figure 12 (§7).

### 6.1 The Kleisli Category of $M$

This subsection provides a summary of the development in [Jia et al. 2021b] of the Kleisli category of the monad $M : \text{DCPO} \to \text{DCPO}$, which we denote $\text{DCPO}_M$. In order to distinguish between the categorical primitives of $\text{DCPO}$ and $\text{DCPO}_M$, we adopt the notation of [Jia et al. 2021b], indicating the morphisms of $\text{DCPO}_M$ by $f : A \to B$, and using $f \circ g \defeq \mu \circ M(f) \circ g$ to denote the Kleisli composition of morphisms in $\text{DCPO}_M$ (where $\mu$ is the multiplication of $M$). We write $\text{id}_A : A \to A$ with $\text{id}_A = \eta_A : A \to MA$ for the identity morphisms in $\text{DCPO}_M$. The adjunction $J \dashv U : \text{DCPO}_M \to \text{DCPO}$ that factorises $M$ is determined by the assignments:

\[
J A \defeq A, \quad J f \defeq \eta \circ f, \quad UA \defeq MA, \quad Uf \defeq \mu \circ Mf.
\]

#### 6.1.1 Coproducts

$\text{DCPO}_M$ inherits (small) coproducts from $\text{DCPO}$ in the standard way [Jacobs 2016, pp. 264] and we write $A_1 + A_2 \defeq A_1 + A_2$ for the induced (binary) coproduct. The induced coprojections are given by $J (\text{id}_1) : A_1 \to A_1 + A_2$ and $J (\text{id}_2) : A_2 \to A_1 + A_2$. Then for $f : A \to C$ and $g : B \to D$, $f + g = [M(\text{id}_C) \circ f, M(\text{id}_D) \circ g]$ and the functor $J$ strictly preserves coproducts.

#### 6.1.2 Symmetric Monoidal Structure

Because $M$ is commutative, it induces a canonical symmetric monoidal structure on $\text{DCPO}_M$ making $J$ a strict monoidal functor [Power and Robinson 1997]. The induced tensor product is $A \times B \defeq A \times B$ with Kleisli projections $J (\pi_A) : A \times B \to A$ and $J (\pi_B) : A \times B \to B$, but these projections do not satisfy the universal property of a product. The tensor product of $f : A \to C$ and $g : B \to D$ is denoted by $f \times g$ and is defined as usual. It follows that Kleisli products distribute over Kleisli coproducts and we write $d_{A, B, C} : A \times (B + C) \equiv (A \times B) + (A \times C)$ for this natural isomorphism.

#### 6.1.3 Kleisli Exponential

The adjunction $J \dashv U$ also contains the structure of a Kleisli-exponential. Following [Moggi 1991], we use this to interpret higher-order function types.

For each dcpo $B$, we use $[\mathsf{B} \to -] \defeq [\mathsf{B} \to U(-)] : \text{DCPO}_M \to \text{DCPO}$ to denote the right adjoint of the functor $J (-) \times B : \text{DCPO} \to \text{DCPO}_M$. Therefore, on objects, $[\mathsf{B} \to C] = [\mathsf{B} \to MC]$. This determines a family of Scott-continuous bijections $\lambda : \text{DCPO}_M(J A \times B, C) \equiv \text{DCPO}(A, [\mathsf{B} \to C])$, natural in $A$ and $C$, often called currying. We also denote the counit of these adjunctions by $e : J [\mathsf{B} \to -] \times B \Rightarrow \text{Id}$, which is often called evaluation. Since this family of adjunctions...
is parameterised by objects \( B \) of \( \mathsf{DCPO}_M \), standard categorical results [Mac Lane 1998, §IV.7] imply the assignment \( [B \rightarrow -] : \mathsf{DCPO}_M \rightarrow \mathsf{DCPO} \) can be extended uniquely to a bifunctor 
\[ [- \rightarrow -] : \mathsf{DCPO}_M \times \mathsf{DCPO}_M \rightarrow \mathsf{DCPO}, \]
such that \( \lambda \) is natural in all three components.

6.1.2 Enrichment Structure. The Kleisli category \( \mathsf{DCPO}_M \) is enriched over \( \mathsf{DCPO}_{\downarrow!} \) for dcpo’s \( A, B \) and \( C \), the Kleisli exponential \( [A \rightarrow B] = [A \rightarrow MB] = \mathsf{DCPO}_M(A, B) \) is a pointed dcpo and the Kleisli composition \( \circ : [A \rightarrow B] \times [B \rightarrow C] \rightarrow [A \rightarrow C] \) is strict and Scott continuous. Furthermore, the adjunction \( \mathcal{F} \vdash \mathcal{U} \) also is \( \mathsf{DCPO} \)-enriched, as are the bifunctors \( (- \times -), (- + -) \) and \( [- \rightarrow -] \).

The category \( \mathsf{DCPO}_M \) also has a convex structure: for each dcpo \( B, MB \) is a Kegelspitze in the stochastic order by Example 5.1.2, from which it follows that \( [A \rightarrow B] = \mathsf{DCPO}_M(A, B) \) also is a Kegelspitze in the pointwise order. This convex structure is preserved by Kleisli composition \( \circ \), Kleisli coproduct + and Kleisli product \( \times \) [Jia et al. 2021b, Lemma 38].

6.1.5 The Subcategories \( \mathsf{TD} \) and \( \mathsf{PD} \). We identify two important subcategories of \( \mathsf{DCPO}_M \): one for the interpretation of classical values (TD) and one for solving recursive domain equations (PD).

**Definition 6.1.1.** The subcategory TD of deterministic total maps is the full-on-objects subcategory of \( \mathsf{DCPO}_M \) whose morphisms \( f : X \rightarrow Y \) admit a factorisation \( f = \mathcal{F}(f') = \eta_Y \circ f' \), for some \( f' \).

Each map \( f : X \rightarrow Y \) in TD satisfies \( f(x) = \delta_y \) for some \( y \in Y \), by definition. We call such maps deterministic because they carry no interesting convex structure, and they are total in that they map all inputs \( x \in X \) to non-zero valuations. TD is important because all classical values of our language are interpreted in TD. In fact, \( \mathsf{DCPO} \equiv \mathsf{TD} \) [Jia et al. 2021b, Proposition 40].

The canonical copy map at an object \( A \) in our model is given by the map \( \mathcal{F}(\text{id}_A, \text{id}_A) : A \rightarrow A \times A \); likewise, the canonical discarding map at \( A \) is the map \( \mathcal{F}(1_A) : A \rightarrow 1 \), where \( 1_A : A \rightarrow 1 \) is the terminal map of \( \mathsf{DCPO} \). Because maps in TD are in the image of \( \mathcal{F} \), they are compatible with the copy and discard maps, and hence also with weakening and contraction [Benton 1995].

**Definition 6.1.2.** The subcategory of deterministic partial maps, denoted PD, is the full-on-objects subcategory of \( \mathsf{DCPO}_M \) each of whose morphisms \( f : X \rightarrow Y \) admits a factorisation \( f = \left( X \xrightarrow{f} Y \xrightarrow{\delta_y} M \right) \), where \( Y_\perp \) is the dcpo obtained from \( Y \) by freely adding a least element \( \perp \), and where \( \delta_y \) is the map \( \delta_y : Y_\perp \rightarrow MY : y \mapsto \begin{cases} 0_Y, & \text{if } y = \perp \\ \delta_y, & \text{if } y \neq \perp. \end{cases} \)

These maps are partial because they map some inputs to 0; they also are deterministic, because the convex structure is trivial in both cases. This is justified by the fact that \( \mathsf{PD} \equiv \mathsf{DCPO}_\tau \equiv \mathsf{DCPO}_{\downarrow!} \), where \( \mathsf{DCPO}_\tau \) is the Kleisli category of the lift monad \( \mathcal{T} : \mathsf{DCPO} \rightarrow \mathsf{DCPO} \) [Jia et al. 2021b].

6.1.6 Solving Recursive Domain Equations. The standard method for interpreting recursive types is to construct parameterised initial algebras [Fiore and Plotkin 1994; Fiore 1994]. We employ this approach in PD using the limit-colimit coincidence theorem [Smyth and Plotkin 1982].

**Definition 6.1.3** (see [Fiore 1994, §6.1]). Given a category \( \mathcal{C} \) and a functor \( \mathcal{T} : \mathcal{C}^{n+1} \rightarrow \mathcal{C} \), a parameterised initial algebra for \( \mathcal{T} \) is a pair \( (\mathcal{T}^\#, i^\mathcal{T}) \), such that:

- \( \mathcal{T}^\# : \mathcal{C}^n \rightarrow \mathcal{C} \) is a functor;
- \( i^\mathcal{T} : \mathcal{T} \circ (\text{id}_\mathcal{C}, \mathcal{T}^\#) \Rightarrow \mathcal{T}^\# : \mathcal{C}^n \rightarrow \mathcal{C} \) is a natural transformation;
- For every \( \mathcal{C} \in \text{Ob}(\mathcal{C}^n) \), the pair \( (\mathcal{T}^\#, i^\mathcal{T}_\mathcal{C}) \) is an initial \( \mathcal{T}(\mathcal{C}, -) \)-algebra.

The usual notion of an initial algebra arises in the case that \( n = 1 \).
**Proposition 6.1.4** (see [Lindenhovius et al. 2021, §4.3]). Let \( C \) be a category with an initial object and all \( \omega \)-colimits, and let \( \mathcal{T} : C^{n+1} \to C \) be an \( \omega \)-cocontinuous functor. Then \( \mathcal{T} \) has a parameterised initial algebra \((\mathcal{T}\mathcal{A}, \mathcal{I})\) and the functor \( \mathcal{T}\mathcal{A} : C^n \to C \) is also \( \omega \)-cocontinuous.

In fact, the subcategory \( \mathsf{PD} \) has sufficient structure to solve recursive domain equations, because it is DCPO-algebraically compact [Jia et al. 2021b]. Therefore, every DCPO-enriched covariant functor on \( \mathsf{DCPO}_M \) that restricts to \( \mathsf{PD} \) has a parameterised initial algebra (whose inverse is a parameterised final coalgebra). Solving equations involving mixed-variance functors (induced by function types) can be done using the limit-colimit coincidence theorem [Smyth and Plotkin 1982].

An important observation made in [Smyth and Plotkin 1982] is that all type expressions (including function spaces) can be interpreted as covariant functors on subcategories of embeddings. For more details on this, see [Lindenhovius et al. 2019, 2021]; here we also follow this approach.

**Definition 6.1.5.** If \( C \) is a DCPO-enriched category, a morphism \( e : X \to Y \) is an embedding if there exists a (necessarily unique) projection \( e^0 : Y \to X \), i.e., a morphism satisfying \( e^0 \circ e = \text{id}_X \) and \( e \circ e^0 \leq \text{id}_Y \). \( C_e \) denotes the full-on-objects subcategory of \( C \) whose morphisms are the embeddings.

**Proposition 6.1.6** ([Jia et al. 2021b, Proposition 47]). The category \( \mathsf{PD}_e \) has an initial object and all \( \omega \)-colimits, and the assignments:

\[
\begin{align*}
\times_e : \mathsf{PD}_e \times \mathsf{PD}_e &\to \mathsf{PD}_e \quad \text{defined by } X \times_e Y \overset{\text{def}}{=} X \times Y \quad \text{and } e_1 \times_e e_2 \overset{\text{def}}{=} e_1 \times e_2 \nonumber \\
+e : \mathsf{PD}_e \times \mathsf{PD}_e &\to \mathsf{PD}_e \quad \text{defined by } X +_e Y \overset{\text{def}}{=} X + Y \quad \text{and } e_1 +_e e_2 \overset{\text{def}}{=} e_1 + e_2 \nonumber \\
[\to]_e : \mathsf{PD}_e \times \mathsf{PD}_e &\to \mathsf{PD}_e \quad \text{defined by } [X \to Y]_e \overset{\text{def}}{=} \mathcal{J}[X \to Y] \quad \text{and } [e_1 \to e_2]_e \overset{\text{def}}{=} \mathcal{J}[e_1^0 \to e_2] \nonumber
\end{align*}
\]

are covariant \( \omega \)-cocontinuous bifunctors on \( \mathsf{PD}_e \).

Thus Propositions 6.1.4 and 6.1.6 show we can solve recursive domain equations induced by all well-formed type expressions within \( \mathsf{PD}_e \), notably with no restrictions on the admissible logical polarities of the types. However, our classical judgements support weakening and contraction, so we have an extra proof obligation: proving each isomorphism that is a solution to a recursive domain equation can be copied and discarded. This is true, because every isomorphism of \( \mathsf{PD} \) (and \( \mathsf{PD}_e \)) also are isomorphisms of \( \mathsf{TD} \) [Jia et al. 2021b, Proposition 48].

### 6.2 The Quantum Category \( \mathbb{Q} \)

We now describe the categorical structure of \( \mathbb{Q} \) and its subcategory \( \mathbb{Q}_e \). We interpret quantum terms in \( \mathbb{Q} \) and quantum values in \( \mathbb{Q}_e \).

#### 6.2.1 Coproducts

Proposition 4.1.4 describes the categorical product on \( \mathsf{vN} \), which restricts to a categorical product on \( \mathsf{HA} \), since the product of hereditarily atomic von Neumann algebras clearly is hereditarily atomic. Moreover, the product on \( \mathsf{HA} \) extends to a product on \( \mathsf{HA} \). As a consequence, \( \mathbb{Q} \) and \( \mathbb{Q}_e \) have small coproducts and we write \( A \oplus B \) to denote the coproduct in both categories. We write the coprojections as \( \text{in}_1 : A \to A \oplus B \) and \( \text{in}_2 : B \to A \oplus B \). Note that the initial object \( \mathbf{0} \) of \( \mathbb{Q}_e \), given by the 1 element HA-algebra, is a zero object in \( \mathbb{Q} \) (but not in \( \mathbb{Q}_e \)).

#### 6.2.2 Symmetric Monoidal Structure

Given two von Neumann algebras \( M \) and \( N \) on Hilbert spaces \( H \) and \( K \), respectively, the algebraic tensor product \( M \otimes N \) acts in a natural way on the Hilbert space tensor product \( H \otimes K \). The weak operator closure of \( M \otimes N \) in \( B(H \otimes K) \) is a von Neumann algebra, usually denoted \( M \bar{\otimes} N \), and called the spatial tensor product of \( M \) and \( N \). The construction in [Blackadar 2006, III.2.2.5] shows the spatial tensor product of von Neumann algebras induces a

\[3\]The same notation is used for Kleisli composition in \( \mathsf{DCPO}_M \), but hopefully this should not lead to confusion.
6.2.3 Adjunctions. The subcategory inclusion \( I : Q_\ast \to Q \) corresponds to an embedding \( HA_\ast \to HA \) that is shown to have a left adjoint in [Westerbaan 2019, Section 4.3.4]. Therefore \( I \) has a right adjoint. Moreover, the adjunction between \( Q_\ast \) and \( Q \) is Kleislian [Westerbaan 2019] and therefore there exists a natural isomorphism \( d_{A,B,C} : A \otimes (B \oplus C) \cong (A \otimes B) \oplus (A \otimes C) \).

6.2.4 Affine Structure. The monoidal unit \( \mathbb{C} \) is initial in \( HA_\ast \) and therefore it is terminal in \( Q_\ast \), but the same is not true for \( HA \) and \( Q \). The terminal map of \( Q_\ast \) at \( M_n(\mathbb{C}) \) is actually \( tr^\dagger \) (see §4.2). This allows us to define suitable discarding maps. The map \( drop_k^\dagger \) should be thought of as discarding \( k \) auxiliary qubits; this map is used for the interpretation of the "run" term when we execute a non-total quantum configuration that has \( k \) auxiliary qubits (which may be safely discarded at the end of the computation). Indeed, notice that in the category \( Q \), we have \( drop_k^\dagger \circ (id_A \otimes state^\dagger_\rho) = id_A \) for any density matrix \( \rho \) in \( M_{2^k}(\mathbb{C}) \), as one would expect (for brevity, we implicitly suppress the isomorphism \( A \otimes \mathbb{C} \cong A \)).

6.2.5 Solving Recursive Domain Equations. We now show that the category \( Q_\ast \) has sufficient structure to construct parameterised initial algebras for polynomial functors. On the quantum side, this covers all recursive domain equations that have to be solved.

**Proposition 6.2.1.** The category \( Q_\ast \) is cocomplete and the functors \( \otimes : Q_\ast \times Q_\ast \to Q_\ast \) and \( \oplus : Q_\ast \times Q_\ast \to Q_\ast \) are cocontinuous.

**Proof.** Cocompleteness of \( Q_\ast \) is shown in [Kornell 2020, Proposition 8.6]; the coproduct bifunctor \( \oplus \) is obviously cocontinuous; the functor \( \otimes \) is cocontinuous because \( Q_\ast \) is monoidal closed.

7 DENOTATIONAL SEMANTICS

We now give the denotational semantics of our language. In Figure 12, we summarise the overall structure of the interpretation. The blue arrows show where the indicated programming primitives can be interpreted. Every such primitive may also be interpreted in a category above it by following the corresponding left adjoint. All depicted categories are symmetric monoidal and so are the

![Diagram](image-url)
adjunctions between them. The left adjoints $L$ and $\ell^{\infty}$ are both fully faithful and this allows us to interpret the \textit{observable} values in the bottom three categories and also to coherently relate these interpretations.

\subsection*{7.1 Interpretation of Types}

We begin with the interpretation of (open) types, which is described in Figure 13. Open quantum types are interpreted as functors $[\Theta \vdash A] : Q^{[\Theta]} \to Q_*$ and open classical types are interpreted as functors $[\Theta \vdash P] : PD_e^{[\Theta]} \to PD_e$. Given closed types $\cdot \vdash A$ and $\cdot \vdash P$, we write $[A] \defeq \llbracket \cdot \vdash A \rrbracket (\ast) \in \text{Ob}(Q_*) = \text{Ob}(Q)$ and $[P] \defeq \llbracket \cdot \vdash P \rrbracket (\ast) \in \text{Ob}(PD_e) = \text{Ob}(\text{DCPO})$.

\textbf{Proposition 7.1.1.} The assignments $[\Theta \vdash A] : Q^{[\Theta]} \to Q_*$ and $[\Theta \vdash P] : PD_e^{[\Theta]} \to PD_e$ are well-defined $\omega$-cocontinuous functors.

\textbf{Proof.} By induction using Propositions 6.1.4, 6.1.6 and 6.2.1. \hfill $\square$

\textbf{Lemma 7.1.2 (Substitution).} Given quantum types $\Theta, X \vdash A$ and $\Theta \vdash B$ and classical types $\Theta, X \vdash P$ and $\Theta \vdash R$, then:

\[ [\Theta \vdash A[B/X]] = [\Theta, X \vdash A] \circ \langle \text{Id}, [\Theta \vdash B] \rangle \quad \text{and} \quad [\Theta \vdash P[R/X]] = [\Theta, X \vdash P] \circ \langle \text{Id}, [\Theta \vdash R] \rangle. \]

For closed recursive types, the folding/unfolding isomorphisms are defined in Figure 18, where the equalities are from Lemma 7.1.2 and the unnamed isomorphisms are the initial algebras. Note that $\text{fold}_{\mu X} : PD_e$ and $\text{fold}_{\mu X A} : Q_*$ are isomorphisms in $\text{TD}$ and $Q_*$, respectively. Now, the derived equations in Figure 14 follow immediately.

\textbf{Relationship between Observable Types.} Quantum/classical observable types play a special role in our language and they also satisfy a special denotational relationship that we now describe.

\textbf{Proposition 7.1.3.} Let $O$ be a closed quantum observable type with $[O]$ its classical counterpart. Then there exists a canonical set $[O]$, defined by induction on the derivation of $\cdot \vdash O$, such that: (1) $[O] \equiv \ell^{\infty}(\llbracket O \rrbracket)$ in $Q_*$, and (2) $[O] = L([O])$, where $L : \text{Set} \to \text{DCPO}$ is the obvious left adjoint functor that equips a set $X$ with the discrete order.

\textbf{Proof.} Both of these statements follow as special cases of the abstract categorical semantics in [Lindenhovius et al. 2019, 2021, Section 6], where the source category is $\text{Set}$ and the target one is $Q_*$ and $\text{TD}$, respectively (note that $\text{TD} \cong \text{DCPO}$, so $\text{DCPO}$ can also be taken). \hfill $\square$

\textbf{Remark 7.1.4.} The coherence conditions outlined in [Lindenhovius et al. 2019, 2021] are very strong and the functor $\ell^{\infty} : \text{Set} \to Q_*$ is a fully faithful strong symmetric monoidal left adjoint (just like $L$). To avoid notational overhead, we treat the $\ast$-isomorphism in (1) above as an equality.

Proposition 7.1.3 shows that the interpretation of any classical observable type is a \textit{discrete domain} and the set $[\langle O \rangle]$ is simply its underlying set. We may safely extend the action of the functor $\ell^{\infty}$ to discrete dcpo’s and then by the above remark it follows that observable types are related by the following strong relationship:

\[ [O] = \ell^{\infty}([\langle O \rangle]). \] (7.1)

This shows the interpretation $[\langle O \rangle]$ of a quantum observable type $O$ is a \textit{commutative} HA-algebra.
Fig. 13. Interpretation of types. $K_X$ is the constant-X-functor.

Fig. 14. Derived equations for closed types.

Fig. 15. Interpretation of classical term judgements.

Fig. 16. Interpretation of quantum term judgements.

Fig. 17. Interpretation of quantum configurations.

fold$_{\mu X.A} : [A[\mu X.A/X]] = [X[ \mu X.A ] ] ≅ [\mu X.A ] : \text{unfold}_{\mu X.A}$

fold$_{\mu X.P} : [P[\mu X.P/X]] = [X[ \mu X.P ] ] ≅ [\mu X.P ] : \text{unfold}_{\mu X.P}$

Fig. 18. Definition of the folding/unfolding isomorphisms.
The interpretation of classical/quantum term judgements and quantum configurations is defined by mutual induction in Figures 15, 16 and 17. Next, we explain some of the notation used therein.

The interpretation of a classical term judgement $\Phi \vdash m : P$ is a morphism $[\Phi \vdash m : P] : [\Phi] \to [P]$ in $\mathbf{DCPO}$ that we often abbreviate by writing $[m]$. Likewise, a quantum term judgement $\Phi; \Gamma \vdash q : A$ is interpreted as a morphism $[\Phi; \Gamma \vdash q : A] : [\Phi] \to Q([\Gamma], [A])$ in $\mathbf{DCPO}$ that we often abbreviate by writing $[q]$. For an element $x \in [\Phi]$, we also write $[q]_x$ and $[m]_x$ as a shorthand for $[q](x)$ and $[m](x)$, respectively. In the special case that $\Phi = \cdot$, we can also regard $[q]$ as a morphism $[q] : [\Gamma] \to [A]$ in $Q$. The unnamed isomorphism in Figure 16 is the left monoidal unitor $\mathbb{C} \otimes A \cong A$ and "swap" is the monoidal symmetry in $Q$.

The interpretation of a configuration $\Phi \vdash ([]\psi, \ell, q) : A; \text{qbit}^k$ is given by a morphism in $\mathbf{DCPO}$ of type $[\Phi \vdash ([]\psi, \ell, q) : A; \text{qbit}^k] : [\Phi] \to Q(\mathbb{C}, [A] \otimes [\text{qbit}^k])$ defined in Figure 17. The notation $\sigma_\ell$ used there denotes any permutation $\sigma_\ell : [\text{qbit}^n] \to [\text{qbit}^n]$ that maps the $i$-th component to $\ell(x_i)$, defined in full analogy to $\sigma$ from Figure 9, and where $n = \dim([\psi])$. In the special case that $C = [\psi], \ell, q$ is total, its interpretation can be seen as a morphism $[C] : [\Phi] \to Q(\mathbb{C}, [A])$ in $\mathbf{DCPO}$, given by $[C] = (x \mapsto [q]_x \circ \sigma_\ell \circ \text{state}_x^\psi(\psi))$. If the linking function $\ell$ is given by $\ell(x_i) = i$ (which we may usually assume), then its interpretation is equivalently given by $[C] = (x \mapsto [q]_x \circ \text{state}_x^\psi(\psi))$. If, in addition, $C$ is closed, i.e., $\Phi = \cdot$, then we can regard it as a map $[C] : C \to A$ in $Q$, which is just a state in $Q$ (and also a state in the operator-algebraic sense).

We now comment on the terms that are of primary interest to us. The interpretation of the "new", "meas" and "U" terms is determined by the constant function on the appropriate $Q$-morphism from §4.2 that is then injected via $\mathcal{J}$ into TD. For quantum lambda abstractions, $[q]$ is, by construction, a Scott-continuous function $[q] : [\Phi] \to Q([\Gamma], [A])$, so we may see it as a morphism of TD via the $\mathcal{J} : \mathbf{DCPO} \cong \mathbf{TD}$ isomorphism. For the interpretation of the "run" term, let us consider the special case when $C$ is total. Then the semantics is equivalently given by the Kleisli morphism $[\text{run}]C = (x \mapsto r(\Gamma, A))$, where $r$ is the isomorphism from Theorem 5.2.1. When $C$ is not total, the drop$_k$ morphism is used to get rid of the remaining auxiliary qubits in accordance with affine principles. The interpretation of the "init" term is done by simply taking the inverse isomorphism $r^{-1}$. Dynamic lifting is interpreted using the natural bijection (−) from Proposition 5.2.2.

Finally, the interpretation of the "mq" term, representing quantum function application, makes use of the barycentre maps from Theorem 5.3.5. In our view, this is the term of highest interest discussed here and that required the most effort to interpret. Notice that its interpretation is unique in that it combines two different Kegelspitzen structures living in two different categories.

### 7.3 Interpretation of (Observable) Values

The interpretation of values in our language enjoys additional structural properties, as usual.

**Proposition 7.3.1.** For any classical value $\Phi \vdash v : P$ and quantum value $\Phi; \Gamma \vdash v : A$, we have:

1. $[v] : [\Phi] \to [P]$ also is a morphism of TD. Equivalently, it is in the image of $\mathcal{J}$.
2. $[v] : [\Phi] \to Q([\Gamma], [A])$ corestricts to $Q((\Gamma, [A])).$ That is, $\forall x \in [\Phi].[v]_x \in Q((\Gamma, [A])).$

This means $[v]_x$ is a Dirac valuation and $[v]_x$ is a $*$-homomorphism, for any $x \in [\Phi]$. Note that the functor $\mathcal{J}$ restricts to an isomorphism of categories $\mathcal{J} : \mathbf{DCPO} \cong \mathbf{TD}$ [Jia et al. 2021b], so for a classical value $v$, we can define $[v] \overset{\text{def}}{=} \mathcal{J}^{-1}[v] : [\Phi] \to [P]$ in $\mathbf{DCPO}$.

The interpretation of classical/quantum observable values enjoys even stronger structural properties and they also are strongly related to each other, as we show next. If $\Phi \vdash v : P$ is an observable value, then $[\Phi]$ and $[P]$ are discrete, so we can safely regard $[v] : [\Phi] \to [P]$ as a morphism in $\mathbf{Set}$. For the next proposition, we identify $\mathbb{C}$ with $l^\infty(1)$ (see Remark 7.1.4).
Proposition 7.3.2. Let \( \vdash \cdot \colon \mathbf{v} \colon \mathbf{O} \) be an observable quantum value and let \( \vdash \cdot : |\mathbf{v}| : |\mathbf{O}| \) be its classical counterpart. Then \( [\mathbf{v}] = \ell^\infty(|\mathbf{v}|) \) and \( ||\mathbf{v}|| = \mathcal{J}(|\mathbf{v}|) \). Furthermore, \( r([\mathbf{v}]) = ||\mathbf{v}|| \).

Proof. By combining Proposition 7.1.3 and [Lindenhovius et al. 2021, Proposition 6.15]. The final statement follows by Theorem 5.2.1 and by construction of the isomorphism \( r \), which relies on the fact that the functors \( \ell^\infty \) and \( \mathcal{J} \) (and implicitly \( \mathcal{L} \)) are fully faithful.

The above proposition is used for establishing soundness for the “run” and “init” terms. The natural isomorphism \( \sim \) from Proposition 5.2.2 satisfies an additional coherence condition (see [Jia et al. 2021a]) w.r.t \( \ell^\infty \), which, combined with Proposition 7.3.2, is used for the soundness proof of the “lift” term. Also, by Remark 2.3.3, it suffices to establish Proposition 7.3.2 for closed observable values, as we have done.

7.4 Soundness and Computational Adequacy

Our final contribution is to show our semantic interpretation is sound and (strongly) adequate.

Lemma 7.4.1 (Substitution). Let \( \Phi \vdash v : P \) be a classical value and \( \Phi, \Sigma \vdash v : A \) a quantum value.

1. If \( \Phi, x : P \vdash m : R \), then \( [m \backslash v / x] = [m] \circ (\text{id}_{|\Sigma|} \otimes \{v\}) \circ \mathcal{J}(\text{id}_{|\Phi|}, \text{id}_{|\Sigma|}) \).
2. If \( \Phi, \Sigma, y : A \vdash q : B \) and \( x \in [\Phi] \), then \( [q \backslash v / y]_x = [q]_x \circ (\text{id}_{|\Sigma|} \otimes \{v\}_x) \).
3. If \( \Phi, \Sigma, z : P, \Sigma \vdash q : B \) and \( x \in [\Phi] \), then \( [q \backslash v / z](x) = [q]_x(\{v\}(x)) \).

Soundness is the statement that our interpretation is invariant under single-step reduction in a probabilistic sense. In both equations sums of morphisms are defined pointwise using the convex structure of the codomain (which is a Kegelspitze).

Theorem 7.4.2 (Soundness). For any classical term \( \Phi \vdash m : P \) and configuration \( \Phi \vdash C : A ; \text{qbit}^k \):

\[
[m] = \sum_{m \rightarrow m'} p[m'] \quad \quad [C] = \sum_{\mathcal{L} \rightarrow \mathcal{C}'} r[C']
\]

assuming \( m \mathcal{P} m' \) and \( C \mathcal{L} C' \) for some rules from the operational semantics (§2) and where the convex sums range over all such rules.

In the above theorem, both sums have at most two summands. Our next theorem is much stronger, because it involves reductions spanning an arbitrary number of steps, and the convex sums can be countably infinite (these can be defined in any Kegelspitze by Definition 5.3.1).

Theorem 7.4.3 (Strong Adequacy). Let \( \vdash \cdot : m : P \) be a closed classical term and \( \vdash \cdot : C : A ; \text{qbit}^k \) a closed quantum configuration. Then:

\[
[m] = \sum_{v \in \text{Val}(m)} P(m \rightarrow_* v) \{v\} \quad \quad [C] = \sum_{\mathcal{V} \in \text{Val}(C)} P(C \rightarrow_* \mathcal{V}) \{\mathcal{V}\}.
\]

Corollary 7.4.4 (Adequacy). Let \( \vdash \cdot : 1 \) be a term and \( \vdash \cdot : C : 1 \) a quantum configuration. Then:

\[
[m]_1(\{\ast\}) = \text{Halt}(m) \quad \quad [C](1) = \text{Halt}(C).
\]

Remark 7.4.5. For our semantic model we chose to use the monad \( \mathcal{M} \), which is commutative, instead of \( \mathcal{V} \), which is not known to be commutative. It is possible to describe the semantics with the standard powerdomain \( \mathcal{V} \) and adequacy would still hold. However, in this case, the induced Kleisli category would be only premonoidal, whereas ours is monoidal, which simplifies the semantics. Furthermore, again assuming an adequate semantics, a commutative monad is more useful, because it can be used to establish a greater range of contextually equivalent programs. Finally, our monad \( \mathcal{M} \) is smaller than \( \mathcal{V} \), so our model has fewer elements that are not denotable.
8 RELATED WORK

In this section we discuss some related work that we view as most relevant to ours.

8.1 QWIRE/EWire

The work closest to ours is QWIRE [Paykin et al. 2017] and EWire [Rennela and Staton 2020]. They are related languages that have a classical and non-linear host language together with a separate small quantum circuit language. However, neither language is suitable for variational quantum programming and our language, VQPL, makes several important additions compared to QWIRE/EWire which address their limitations, as we now explain.

A major difference in focus between QWIRE/EWire and VQPL is that the former is concerned with quantum circuits (as one can see from their names), whereas VQPL is concerned with quantum and probabilistic effects and how they interact. In QWIRE/EWire, the quantum types \( A \) consist of tensor products of atomic types, but they are not closed under formation of quantum sum types or quantum inductive types\(^4\). Because of this, the type \( \text{Circ}(A, B) \) in QWIRE/EWire may be thought of as the type of quantum circuits from type \( A \) into type \( B \). This type corresponds to \( Q(A, B) \) in VQPL, but as we allow quantum sum types and quantum inductive types, we should not think of it as a type of circuits. Furthermore, in our system, quantum terms represent effectful quantum computation and reductions induced by quantum measurement are probabilistic (on the quantum side), whereas in QWIRE/EWire, the operational semantics on the quantum side is deterministic (no probabilistic effects are allowed), which is another reason not to think of \( Q(A, B) \) as representing quantum circuits in VQPL. This is why we emphasised that the appropriate way to view the type \( Q(A, B) \) is as a first-order quantum lambda abstraction, which is similar to [Péchoux et al. 2020a,b].

In QWIRE/EWire probabilistic effects are only allowed on the classical side, but not on the quantum side. This is indeed reflected by their operational semantics, as we mentioned in the preceding paragraph. Furthermore, “unboxing” in QWIRE/EWire is restricted to pure and deterministic (i.e. non-probabilistic) programs [Rennela and Staton 2020, pp. 7]. In VQPL, “unboxing” corresponds to “quantum function application” and there are no such restrictions. This operation is not the same as monadic sequencing in Moggi’s metalanguage, because the typing judgement is quantum, not classical. This is one of the major additions that VQPL makes over QWIRE/EWire and it allows us to freely mix effectful probabilistic and quantum computation. This rule is not derivable within QWIRE/EWire. Modelling this denotationally is one of the main contributions of our paper, relying on Theorems 4.3.1, 5.3.4 and 5.3.5. Interpreting this rule requires domain-enrichment and not just dcpo-enrichment in our semantics (we do not know how to achieve this otherwise), which further highlights the differences between the languages and their denotational and operational semantics. In fact, this rule is one of the main reasons why we refer to "quantum effects" and "quantum function application", rather than "quantum circuits" and "unboxing", which we view as too restrictive in the context of effectful probabilistic/quantum computation.

Other additions included in VQPL are inductive quantum types (only informally discussed in EWire) and recursive classical types. We also include a precise formulation of the operational semantics on both the classical and quantum sides, whereas QWIRE/EWire do not formally specify an operational semantics on the classical side. Furthermore, both languages have only limited support for recursion and neither of these papers proves an adequacy result. The only soundness statements that are proven are with respect to strongly-normalising fragments of the languages and the soundness statements cover reductions on the quantum side only. Overall, our denotational and operational treatment is considerably more comprehensive.

\(^4\)Quantum inductive types are only informally discussed in [Rennela and Staton 2020].
8.2 Quantum Lambda Calculi

Other related work includes adequate [Tsukada et al. 2018] and even fully-abstract semantics for the quantum lambda calculus [Clairambault and de Visme 2020; Clairambault et al. 2019; Pagani et al. 2014]. These models are inspired by quantitative models of linear logic and we will collectively refer to them as the quantitative models. In this version of the quantum lambda calculus, classical information is represented by types of the form $!(A \rightarrow B)$, so by using the well-known (call-by-value) translation of the non-linear (intuitionistic) function space in linear logic, one can also reason about classical functions. In our type system, we do not have an explicit $!$-modality and we have two essentially separate languages (one classical, one quantum) together with some terms that mediate between the classical/quantum modes of operation. In particular, our approach shows how one can extend an existing classical programming language with features for quantum programming while keeping the quantum fragment relatively small and simple.

In fact, the quantum part of our model does support a $!$-modality with base category given by $\textbf{Set}$ (see Figure 12) that can be modelled in the same way as in [Cho and Westerbaan 2016], where the authors consider a different version of the quantum lambda calculus with subtyping, but without recursion. In fact, the model in [Cho and Westerbaan 2016] is also based on von Neumann algebras and it contains the quantum part of our model. The problem with this approach to interpreting the $!$-modality is that the induced $!$-endofunctor on $\mathcal{Q}$ is not DCPO-enriched, so it does not behave well under (type) recursion.

Another major difference is that our language has (only) first-order quantum lambda abstractions (represented by the type $\mathcal{Q}(A, B)$), and this is why we do not need a $!$-modality in our language. While the quantitative models support higher-order quantum lambda abstractions, we cannot easily interpret them (together with recursion), because there is no known order on the category $\mathcal{Q}$, that makes the adjunction $\mathcal{Q} \leftarrow \mathcal{Q} \rightarrow \text{DCPO}$-enriched. Even though $\mathcal{Q}$, is monoidal closed, the induced Kleisli exponential is not DCPO-enriched, so it does not behave well under (type) recursion. On the other hand, the category $\mathcal{Q}$ is enriched in a very strong sense (over domains), which is crucial for establishing the strong correspondence between classical probabilistic and quantum effects that we presented. Resolving this conundrum is left for future work.

The quantitative models do not investigate all inductive/recursive types, but only natural numbers and lists. Our paper shows how to interpret all classical recursive types and all inductive quantum types. How to model all recursive quantum types (i.e. those where $\rightarrow$ and $!$ are admissible in type recursion) is still an open problem.

Another difference between the quantitative models and our model is that our model is more amenable to continuous probability distributions. In the present paper we only consider discrete probabilistic choice, but we believe that our model does not require extensive changes to support continuous probabilistic choice. We conjecture that by replacing the submonad $\mathcal{M}$ with another suitable submonad of $\mathcal{V}$ (e.g. the central valuations monad [Jia et al. 2021c]) or even taking $\mathcal{V}$ itself, one can recover a model that supports continuous probability distributions. However, it seems it would be very difficult to modify the quantitative models to support continuous probability.

ACKNOWLEDGMENTS

We thank the anonymous POPL reviewers for their feedback. XJ acknowledges the support of NSFC (No. 12001181); AK and MM acknowledge financial support from the AFOSR under the MURI grant FA9550-16-1-0082 entitled, "Semantics, Formal Reasoning, and Tool Support for Quantum Programming"; VZ acknowledges support by the STIC-AmSud project QAPLA: "Quantum aspects of programming languages."
REFERENCES

M. Abadi and M. P. Fiore. 1996. Syntactic Considerations on Recursive Types. In Proceedings 11th Annual IEEE Symposium on Logic in Computer Science. 242–252. https://doi.org/10.1109/LICS.1996.561324

S. Abramsky and A. Jung. 1994. Domain Theory. In Handbook of Logic in Computer Science (Vol. 3). Oxford University Press, Oxford, UK, 1–168. http://dl.acm.org/citation.cfm?id=218742.218744

Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. 1993. Teleporting an Unknown Quantum State via Dual Classical and EPR Channels.

P.N. Benton. 1995. A mixed linear and non-linear logic: Proofs, terms and models. In Computer Science Logic: 8th Workshop, CSL ’94, Selected Papers. https://doi.org/10.1007/BFb0022251

P. N. Benton and P. Wadler. 1996. Linear Logic. Monads and the Lambda Calculus. In LICS 1996.

B. Blackadar. 2006. Operator Algebras: Theory of C*-algebras and von Neumann algebras. Springer-Verlag.

K. Cho. 2016. Semantics for a Quantum Programming Language by Operator Algebras. New Generation Comput. 34, 1-2 (2016), 25–68. https://doi.org/10.1007/s00354-016-0204-3

Kenta Cho and Abraham Westerbaan. 2016. Von Neumann Algebras form a Model for the Quantum Lambda Calculus. CoRR abs/1603.02133 (2016). arXiv:1603.02133 http://arxiv.org/abs/1603.02133

Pierre Clairambault and Marc de Visme. 2020. Full abstraction for the quantum lambda-calculus. Proc. ACM Program. Lang. 4, POPL (2020), 63:1–63:28. https://doi.org/10.1145/3371138

Pierre Clairambault, Marc de Visme, and Glynn Winskel. 2019. Game semantics for quantum programming. Proc. ACM Program. Lang. 3, POPL (2019), 32:1–32:29. https://doi.org/10.1145/3290345

Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 [quant-ph]

Marcelo Fiore and Gordon Plotkin. 1994. An Axiomatization of Computationally Adequate Domain Theoretic Models of FPC. In LICS. IEEE Computer Society, 92–102.

M. P. Fiore. 1994. Axiomatic domain theory in categories of partial maps. Ph.D. Dissertation. University of Edinburgh, UK.

R. Furber. 2019. Continuous Dcpos in Quantum Computing. preprint (2019). http://people.cs.aau.dk/~furber/papers/contawconf.pdf

G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. W. Mislove, and D. S. Scott. 2003. Continuous Lattices and Domains. Cambridge University Press.

J-Y. Girard. 1987. Linear Logic. Theoretical Computer Science 50 (1987), 1 – 101.

Robert Harper. 2016. Practical Foundations for Programming Languages (2nd Ed.). Cambridge University Press. https://www.cs.cmu.edu/~rwh/pfpl/index.html

Bart Jacobs. 2016. Introduction to Coalgebra: Towards Mathematics of States and Observation. Cambridge Tracts in Theoretical Computer Science, Vol. 59. Cambridge University Press. https://doi.org/10.1017/CBO9781107064643

Xiaodong Jia, Andre Kornell, Bert Lindenhovius, Michael W. Mislove, and Vladimir Zamzhiiev. 2021a. Semantics for Variational Quantum Programming. CoRR abs/2107.13347 (2021). arXiv:2107.13347 https://arxiv.org/abs/2107.13347

Extended version of this POPL paper.

Xiaodong Jia, Bert Lindenhovius, Michael Mislove, and Vladimir Zamzhiiev. 2021b. Commutative Monads for Probabilistic Programming Languages. In Logic in Computer Science (LICS 2021). arXiv:2102.00510 [cs.LO]

Xiaodong Jia, Michael Mislove, and Vladimir Zamzhiev. 2021c. The Central Valuations Monad. CoRR abs/2111.10873 (2021). arXiv:2111.10873 https://arxiv.org/abs/2111.10873

Claire Jones. 1990. Probabilistic Non-determinism. Ph.D. Dissertation. University of Edinburgh, UK. http://hdl.handle.net/1842/413

C. Jones and Gordon D. Plotkin. 1989. A Probabilistic Powerdomain of Evaluations. In Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS ’89), Pacific Grove, California, USA, June 5-8, 1989. IEEE Computer Society, 186–195. https://doi.org/10.1109/LICS.1989.39173

R.V. Kadison and J.R. Ringrose. 1997. Fundamentals of the Theory of Operator Algebras, Volume I: Elementary Theory. American Mathematical Society.

Klaus Keimel and Gordon D. Plotkin. 2017. Mixed powerdomains for probability and nondeterminism. Logical Methods in Computer Science 13, Issue 1 (Jan. 2017). https://doi.org/10.23638/LMCS-13(1:2)2017

John L. Kelley. 1975. General Topology. Number 27 in Graduate Texts in Mathematics. Springer-Verlag.

A. Kornell. 2020. Quantum Sets. J. Math. Phys. 61 (2020), 102202. https://doi.org/10.1063/1.5054128

Klaas Landsman. 2017. Foundations of Quantum Theory - From Classical Concepts to Operator Algebras. Springer Open.

Bert Lindenhovius, Michael Mislove, and Vladimir Zamzhiev. 2019. Mixed Linear and Non-linear Recursive Types. Proc. ACM Program. Lang. 3, ICFP, Article 111 (Aug. 2019), 29 pages. https://doi.org/10.1145/3341715

Bert Lindenhovius, Michael Mislove, and Vladimir Zamzhiev. 2021. LNL-FPC: The Linear/Non-linear Fixpoint Calculus. Logical Methods in Computer Science Volume 17, Issue 2 (April 2021). https://lmcs.episciences.org/7390

Saunders Mac Lane. 1998. Categories for the Working Mathematician (2nd ed.). Springer.

Proc. ACM Program. Lang., Vol. 6, No. POPL, Article 26. Publication date: January 2022.
Semantics for Variational Quantum Programming

Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. 2016. The theory of variational hybrid quantum-classical algorithms. New Journal of Physics 18, 2 (2016), 023023.

Eugenio Moggi. 1991. Notions of Computation and Monads. Inf. Comput. 93, 1 (1991), 55–92. https://doi.org/10.1016/0890-5401(91)90052-4

Adam Paetznick and Krysta M. Svore. 2014. Repeat-until-Success: Non-Deterministic Decomposition of Single-Qubit Unitaries. Quantum Inf. Comput. 14, 15–16 (Nov. 2014), 1277–1301.

Michele Pagani, Peter Selinger, and Benoît Valiron. 2014. Applying quantitative semantics to higher-order quantum computing. In POPL ’14. ACM, 647–658. https://doi.org/10.1145/2535838.2535879

J. Paykin, R. Rand, and S. Zdancewic. 2017. QWIRE: a core language for quantum circuits. In POPL. ACM, 846–858.

Romain Péchoux, Simon Perdrix, Mathys Rennela, and Vladimir Zamdzhiev. 2020a. Quantum Programming with Inductive Datatypes. https://homepages.loria.fr/VZamdzhiev/papers/qpl-inductive.pdf Preprint.

Romain Péchoux, Simon Perdrix, Mathys Rennela, and Vladimir Zamdzhiev. 2020b. Quantum Programming with Inductive Datatypes: Causality and Affine Type Theory. In Foundations of Software Science and Computation Structures - 23rd International Conference, FOSSACS 2020 (Lecture Notes in Computer Science, Vol. 12077). Springer, 562–581. https://doi.org/10.1007/978-3-030-45231-5_29

Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O’brien. 2014. A variational eigenvalue solver on a photonic quantum processor. Nature communications 5, 1 (2014), 1–7.

John Power and Edmund Robinson. 1997. Premonoidal Categories and Notions of Computation. Math. Struct. Comput. Sci. 7, 5 (1997), 453–468. https://doi.org/10.1017/S0960129597002375

Mathys Rennela and Sam Staton. 2020. Classical Control, Quantum Circuits and Linear Logic in Enriched Category Theory. Log. Methods Comput. Sci. 16, 1 (2020). https://doi.org/10.23638/LMCS-16(1:30)2020

P. Selinger. 2004a. Towards a quantum programming language. Mathematical Structures in Computer Science 14, 4 (2004), 527–586.

Peter Selinger. 2004b. Towards a semantics for higher-order quantum computation. Proceedings of the 2nd International Workshop on Quantum Programming Languages, 127–143.

M.B. Smyth and G.D. Plotkin. 1982. The Category-theoretic Solution of Recursive Domain Equations. Siam J. Comput. (1982).

M. Takesaki. 2000. Theory of Operator Algebra I. Springer.

Takeshi Tsukada, Kazuyuki Asada, and C.-H. Luke Ong. 2018. Species, Profunctors and Taylor Expansion Weighted by SMCC: A Unified Framework for Modelling Nondeterministic, Probabilistic and Quantum Programs. In Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018, Anuj Dawar and Erich Grädel (Eds.). ACM, 889–898. https://doi.org/10.1145/3209108.3209157

Abraham Westerbaan. 2019. The Category of von Neumann algebras, PhD Thesis. arXiv:arxiv:1804.02203

William K Wootters and Wojciech H Zurek. 1982. A single quantum cannot be cloned. Nature 299, 5886 (1982), 802–803.