Semantics of Higher-Order Quantum Computation via Geometry of Interaction

Ichiro Hasuo\textsuperscript{a,1,*}, Naohiko Hoshino\textsuperscript{b,2}

\textsuperscript{a}Department of Computer Science, The University of Tokyo, Hongo 7-3-1, Tokyo 113-8656, Japan
\textsuperscript{b}Research Institute for Mathematical Sciences, Kyoto University, Kitashirakawa-Oiwakecho, Kyoto 606-8502, Japan

Abstract

While much of the current study on quantum computation employs low-level formalisms such as quantum circuits, several high-level languages/calculi have been recently proposed aiming at structured quantum programming. The current work contributes to the semantical study of such languages by providing interaction-based semantics of a functional quantum programming language; the latter is, much like Selinger and Valiron’s, based on linear lambda calculus and equipped with features like the $!$ modality and recursion. The proposed denotational model is the first one that supports the full features of a quantum functional programming language; we prove adequacy of our semantics. The construction of our model is by a series of existing techniques taken from the semantics of classical computation as well as from process theory. The most notable among them is Girard’s Geometry of Interaction (GoI), categorically formulated by Abramsky, Haghverdi and Scott. The mathematical genericity of these techniques—largely due to their categorical formulation—is exploited for our move from classical to quantum.

Keywords: higher-order computation, quantum computation, programming language, geometry of interaction, denotational semantics, categorical semantics

\textsuperscript{*}An earlier version of this paper \cite{1} has been presented at the Twenty-Sixth Annual IEEE Symposium on Logic in Computer Science (LICS 2011), 21-24 June 2011, Toronto, Ontario, Canada.

\textsuperscript{*}Corresponding author

Email addresses: ichiro@is.s.u-tokyo.ac.jp (Ichiro Hasuo), naophiko@kurims.kyoto-u.ac.jp (Naohiko Hoshino)

URL: http://www.is.s.u-tokyo.ac.jp/~ichiro (Ichiro Hasuo), http://www.kurims.kyoto-u.ac.jp/~naophiko (Naohiko Hoshino)

\textsuperscript{1}Supported by Grants-in-Aid for Young Scientists (A) No. 24680001, JSPS, and by Aihara Innovative Mathematical Modelling Project, FIRST Program, JSPS/CSTP.

\textsuperscript{2}Supported by Grants-in-Aid for Young Scientists (B) No. 26730004, JSPS.
1. Introduction

1.1. Quantum Programming Languages

Computation and communication using quantum data has attracted growing attention. On the one hand, quantum computation provides a real breakthrough in computing power—at least for certain applications—as demonstrated by Shor’s algorithm. On the other hand, quantum communication realizes “unconditional security” e.g. via quantum key distribution. Quantum communication is being physically realized and put into use.

The extensive research efforts on this new paradigm have identified some challenges, too. On quantum computation, aside from a few striking ones such as Shor’s and quantum search algorithms, researchers are having a hard time finding new “useful” algorithms. On quantum communication, the counterintuitive nature of quantum data becomes an additional burden in the task of getting communication protocols right—which has proved extremely hard already with classical data.

Structured programming and mathematically formulated semantics are potentially useful tools against these difficulties. Structured programming often leads to discovery of ingenious algorithms; well-formulated semantics would provide a ground for proving a communication protocol correct.

In this direction, there have been proposed several high-level languages tailored for quantum computation (see [2] for an excellent survey). Among the first ones is QCL [3] that is imperative; the quantum IO monad [4] and its successor Quipper [5] are quantum extensions of Haskell that facilitate generation of quantum circuits. Closely related to the latter two is the one in [6], that is an (intuitionistic) $\lambda$-calculus with quantum stores.

Another important family—that is most strongly oriented towards mathematical semantics—is those of quantum $\lambda$-calculi that are very often based on linear $\lambda$-calculus. While $\lambda$-calculus is a prototype of functional programming languages and inherently supports higher-order computation, linearity in a type system provides a useful means of prohibiting duplication of quantum data (“no-cloning”). Examples of such languages are found in [7, 8, 9, 10, 11, 12, 13].

1.2. Denotational Semantics of Quantum Programming Languages

Models of linear logic (and hence of linear $\lambda$-calculus) have been studied fairly well since 1990s; therefore denotational models for the last family of quantum programming languages may well be based on those well-studied models. Presence of quantum primitives—or more precisely coexistence of “quantum data, classical control”—poses unique challenges, however. It thus seems that denotational semantics for quantum $\lambda$-calculi has attracted research efforts, not only from those interested in quantum computation, but also from the semantics community in general, since it offers unique and interesting “exercises” to the semantical techniques developed over many years, many of which are formulated in categorical terms and hence are aimed at genericity.
Consider a quantum $\lambda$-calculus that is essentially a linear $\lambda$-calculus with quantum primitives. It is standard that compact closed categories provide models for the latter; so we are aiming a compact closed category 1) with a quantum flavor, and 2) that allows interpretation of the $!$ modality that is essential in duplicating classical data. This turns out to be not easy at all. For example, the requirement 1) makes one hope that the category $\text{fdHilb}$ of finite-dimensional Hilbert spaces and linear maps would work. This category however has no convenient “infinity” structure that can be exploited for the requirement 2). Moving to the category $\text{Hilb}$ of possibly infinite-dimensional Hilbert spaces does not work either, since it is not compact closed.

A few attempts have been made to address this difficulty. In [8] a categorical model is presented that is fully abstract for the $!$-free fragment of a quantum $\lambda$-calculus is presented. It relies on Selinger’s category $Q$ in [7]—it can be thought of as an extension of $\text{fdHilb}$ with non-duplicable classical information. The works [14, 15] essentially take “completions” of this model to accommodate the $!$-modality: the former [14] uses presheaves and thus results in a huge model; the construction in the latter [15] keeps a model in a tractable size by the general semantical technique called quantitative semantics [16, 17].

The difference between the two is comparable to the one between Girard’s normal functor semantics [18] (see also [19]) and quantitative semantics.

In this paper we take a different path towards a denotational model of a quantum $\lambda$-calculus. Instead of starting from $\text{fdHilb}$ (a purely quantum model) and completing it with structures suitable for classical data, we start from a general family of models of classical computation, and fix its parameter so that the resulting instance accommodates quantum data too. The family of models is the one given by Girard’s geometry of interaction (GoI) [20]—more specifically its categorical formulation by Abramsky, Haghverdi and Scott [21]. GoI, like game semantics [22, 23], is an interaction-based denotational semantics of (classical) computation that has a strong operational flavor, too. It thus possibly enables us to extract a compiler from a denotational model, which is the case with classical computation [24, 25, 26, 27, 28].

1.3. Contributions

In this paper we introduce a calculus Hoq and its denotational model that supports the full features (including the $!$ modality and recursion). The language Hoq is almost the same as Selinger and Valiron’s quantum $\lambda$-calculus [9]—in particular we share their principle of “quantum data, classical control”—but is modified for a better fit to our denotational model. We also define its operational semantics and prove adequacy.

For the construction of the denotational model we employ a series of existing techniques in theoretical computer science (Fig. 1). Namely: 1) a monad with an order structure for modeling branching, used in the coalgebraic study of state-based systems (e.g. in [29]); 2) Girard’s Geometry of Interaction (GoI) [20],

---

3 “Classical” as opposed to “quantum”; not as the opposite of “intuitionistic”. 

3
categorically formulated by Abramsky, Haghverdi and Scott [21], providing interaction-based, game-like semantics for linear logic and computation; 3) the realizability technique that turns an (untyped) combinatory algebra into a categorical model of a typed calculus (in our case a linear category [30, 31]; the linear realizability technique is used e.g. in [32]); and 4) the continuation-passing style (CPS) semantics. In each stage we benefit from the fact that the relevant technique is formulated in the language of category theory: the technique is originally for classical computation but its genericity makes it applicable to quantum settings.

Figure 1: The construction of the model

1.4. Organization of the Paper

In §2 we fix the notations for quantum computation and briefly review the semantical techniques used later. In §3 we introduce our target language Hoq and its operational semantics. The (subtle but important) differences from its predecessor are discussed, too. In §4 we introduce the quantum branching monad \( Q \) on \( \text{Sets} \); this is our choice for the monad \( B \) in Fig. 1. The resulting linear category \( \text{PER}_Q \) is described, too. In §5 we interpret Hoq in this category; finally in §6 we prove adequacy of the denotational model.

2. Preliminaries

We denote the syntactic equality by \( \equiv \).

2.1. Quantum Computation

We follow Kraus’ formulation [36] of quantum mechanics, which is by now standard and is used in e.g. [37, 7]. For proofs and more detailed explanation, our principal reference is the standard textbook [37, Chap. 2 & Chap. 8].

Notation 2.1. \( I_m \) denotes the \( m \times m \) identity matrix; \( A^\dagger \) denotes a matrix \( A \)’s adjoint (i.e. conjugate transpose).
2.1.1. Density Matrices

We motivate the formalism of density matrices as one that generalizes the state vector formalism. See [37, §2.4] for more details; for our developments later, it is crucial that we allow density matrices \( \rho \) such that \( \text{tr}(\rho) \) is possibly less than 1.

A mathematical representation of a state of a quantum mechanical system is standardly given by a normalized vector \( |v\rangle \) with \( \| |v\rangle \| = 1 \) in some Hilbert space \( \mathcal{H} \). As is usual in the context of quantum information and quantum computation, we will be working exclusively with finite-dimensional systems \( \mathcal{H} \cong \mathbb{C}^n \) for some \( n \in \mathbb{N} \). As an example let us consider the following Bell state:

\[
|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|0_10_2\rangle + |1_11_2\rangle) .
\] (1)

The vector \( |\Phi^+\rangle \in \mathbb{C}^4 \) is a state of a 2-qubit system; we shall sometimes use explicit subscripts \( 1, 2 \) as in \( 0_2 \) above to designate which of the two qubits we are referring to.

We now consider the measurement of the first qubit with respect to the basis consisting of \( |0\rangle \) and \( |1\rangle \). The outcome is \( |0\rangle \) or \( |1\rangle \) with the same probabilities \( 1/2 \); in each case the state vector gets reduced and becomes \( |0_10_2\rangle \) or \( |1_11_2\rangle \), respectively. In other words, the result of the measurement is a probability distribution

\[
\left[ |0_10_2\rangle \mapsto \frac{1}{2} , |1_11_2\rangle \mapsto \frac{1}{2} \right]
\]

over state vectors. Such is called an ensemble.

Density matrices generalize state vectors and also encompass ensembles; in other words, they represent both pure and mixed states. Given an ensemble

\[
\left[ |v_i\rangle \mapsto p_i \right] \quad \text{with} \quad p_i \in \mathbb{R}_{\geq 0} \quad \text{and} \quad \sum_i p_i \leq 1 ,
\]

the corresponding density matrix is defined to be

\[
\sum_i p_i |v_i\rangle \langle v_i| ,
\]

where \( \langle v_i| = |v_i\rangle^\dagger \) as usual. For example, the Bell state \( |\Phi^+\rangle \) is represented by the density matrix

\[
|\Phi^+\rangle \langle \Phi^+| = \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} ;
\] (2)

the ensemble \( \left[ |0_10_2\rangle \mapsto \frac{1}{2} , |1_11_2\rangle \mapsto \frac{1}{2} \right] \) that results from the measurement above is represented by

\[
\frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} .
\] (3)

5
Notation 2.2. In (2–3) we followed the common lexicographic indexing convention: the matrices are with respect to the basis vectors $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$ in this order. See e.g. [7, §3.2]. This convention will be used in the rest of the paper too.

Here is an “axiomatic” definition of density matrix. Every density matrix arises in the way described above from some ensemble; see [37, Thm. 2.5].

Definition 2.3 (Density matrix). An $m$-dimensional density matrix is an $m \times m$ matrix $\rho \in \mathbb{C}^{m \times m}$ which is positive and satisfies $\text{tr}(\rho) \in [0, 1]$. Here $[0,1]$ denotes the unit interval. The set of all $m$-dimensional density matrices is denoted by $\text{DM}_m$.

Note that we allow density matrices with trace less than 1. This will be the case typically when “some probability is missing,” such as when the original ensemble $[|v_i\rangle \mapsto p_i]_{i \in I}$ is such that $\sum_i p_i < 1$. This generality turns out to be very useful later when we model classical control structures that depend on the outcome of measurements.

We note that if a quantum system consists of $N$ qubits, then the system is $2^N$-dimensional (we can take a basis that consists of $|00\rangle, |01\rangle, \ldots, |11\rangle$).

In this case a density matrix that represents a (pure or mixed) state will be $2^N \times 2^N$.

The following order is standard and used e.g. in [37, 7].

Definition 2.4 (Löwner partial order). The order $\sqsubseteq$ on the set $\text{DM}_m$ of density matrices is defined by: $\rho \sqsubseteq \sigma$ if and only if $\sigma - \rho$ is a positive matrix.

A prototypical situation in which we have $\rho \sqsubseteq \sigma$ is when

- $\rho$ arises from an ensemble $[|v_i\rangle \mapsto p_i]_{i \in I}$;
- $\sigma$ arises from an ensemble $[|v_i\rangle \mapsto q_i]_{i \in I}$; and
- $p_i \leq q_i$ for each $i \in I$.

That is, when, in comparing $\rho$ and $\sigma$ thought of as ensembles, $\rho$ has some components missing.

The following fact is crucial in this work. It is proved in [7, Prop. 3.6] using a translation into quadratic forms; in Appendix AppendixA we present another proof using matrix norms.

Lemma 2.5. The relation $\sqsubseteq$ in Def. 2.4 is indeed a partial order. Moreover it is an $\omega$-CPO: any increasing $\omega$-chain $\rho_0 \sqsubseteq \rho_1 \sqsubseteq \cdots$ in $\text{DM}_m$ has the least upper bound. \[\square\]

2.1.2. Quantum Operations

Built on top of the density matrix formalism, the notion of quantum operation captures the general concept of “what we can do to quantum systems,” unifying preparation, unitary transformation and measurement. See [37, Chap. 8] for details.
Definition 2.6 (Quantum operation, QO). A quantum operation (QO) from an $m$-dimensional system to an $n$-dimensional system is a mapping $\mathcal{E} : \text{DM}_m \rightarrow \text{DM}_n$ subject to the following axioms.

1. (Trace condition) \[
\frac{\text{tr}(\mathcal{E}(\rho))}{\text{tr}(\rho)} \in [0, 1] \text{ for any } \rho \in \text{DM}_m.
\]

2. (Convex linearity) Let $(\rho_i)_{i \in I}$ be a family of $m$-dimensional density matrices; and $(p_i)_{i \in I}$ be a probability subdistribution (meaning $p_i \in \mathbb{R}_{\geq 0}$ and \[\sum_{i} p_i \leq 1\]). Then:
\[
\mathcal{E}\left(\sum_{i \in I} p_i \rho_i\right) = \sum_{i \in I} p_i \mathcal{E}(\rho_i).
\]

3. (Complete positivity) An arbitrary “extension” of $\mathcal{E}$ of the form $I_k \otimes \mathcal{E} : M_k \otimes \text{DM}_m \rightarrow M_k \otimes \text{DM}_n$ carries a positive matrix to a positive one. In particular, so does $\mathcal{E}$ itself.

The set of QOs of the type $\text{DM}_m \rightarrow \text{DM}_n$ shall be denoted by $\text{QO}_{m,n}$.

The definition slightly differs from the one in [37, §8.2.4]. This difference—which is technically minor but conceptually important—is because we allow density matrices with trace less than 1.

QO has two alternative definitions other than the above “axiomatic” one. One is by the operator-sum representation $\sum_i A_i(\_\_) A_i^\dagger$ and useful in concrete calculations. This is presented below. The other is “physical” and describes a QO as a certain succession of operations to a system, namely: combining with an auxiliary quantum state; a unitary transformation; and measurement. See [37, §8.2] for further details.

Proposition 2.7 (Operator-sum representation). A mapping $\mathcal{E} : \text{DM}_m \rightarrow \text{DM}_n$ is a QO if and only if it can be represented in the form

$$
\mathcal{E}(\rho) = \sum_{i \in I} E^{(i)}(\rho(E^{(i)}))^\dagger,
$$

(4)

where $I$ is a finite index set, $E^{(i)}$ is an $n \times m$ matrix for each $i$, and

$$
\sum_{i \in I} E^{(i)^\dagger} E^{(i)} \subseteq \mathcal{I}_m.
$$

Here the order $\subseteq$ refers to the one in Def. 2.4.

Proof. See [37, §8.2.4]. \qed

We call the right-hand side of (4) an operator-sum representation of a QO $\mathcal{E}$. Given a QO $\mathcal{E}$, its operator-sum representation is not uniquely determined. However:
Definition 2.8 (The matrix $M(\mathcal{E})$). For a QO $\mathcal{E} = \sum_i E^{(i)}(\_)(E^{(i)})^\dagger$, we define an $m \times m$ matrix $M(\mathcal{E})$ by

$$M(\mathcal{E}) := \sum_i (E^{(i)})^\dagger E^{(i)}.$$ 

Lemma 2.9. The matrix $M(\mathcal{E})$ for a QO $\mathcal{E}$ does not depend on the choice of an operator-sum representation.

Proof. There is only “unitary freedom” in the choice of an operator-sum representation [37, Thm. 8.2]: given two operator-sum representations $\mathcal{E} = \sum_i E^{(i)}(\_)(E^{(i)})^\dagger = \sum_j F^{(j)}(\_)(F^{(j)})^\dagger$, there exists a unitary matrix $U = (u_{i,j})_{i,j}$ such that $E^{(i)} = \sum_j u_{i,j} F^{(j)}$. We have

$$\sum_i (E^{(i)})^\dagger E^{(i)} = \sum_i (\sum_j u_{i,j}^* (F^{(j)})^\dagger (\sum_k u_{i,k} F^{(k)})) = \sum_j (F^{(j)})^\dagger F^{(j)},$$

where the last equality is because $\sum_i u_{i,j}^* u_{i,k}$ is the $(j,k)$-entry of $U^\dagger U = I$. □

The following property is immediate.

Lemma 2.10. The operation $M(\_)$ preserves sums. More precisely, let $(\mathcal{E}_i)_{i \in I}$ be a family of quantum operations of the same dimensions; assume that $\sum_i \mathcal{E}_i$ is again a quantum operation. Then $M\left(\sum_i \mathcal{E}_i\right) = \sum_i M(\mathcal{E}_i)$. □

We exhibit some concrete QOs. Application of a unitary transformation $U$ to a quantum state (pure or mixed) that is represented by a density matrix $\rho$ corresponds to a QO $U(\_): \rho \rightarrow U\rho U^\dagger$.

For illustration consider a special case where $\rho = |v\rangle\langle v|$; the outcome is

$$U|v\rangle\langle v| U^\dagger = U|v\rangle\langle v| U^\dagger = U|v\rangle\langle U|v\rangle^\dagger,$$

i.e. the density matrix that is induced by the state vector $U|v\rangle$.

We explain measurement using a concrete example. Recall the Bell state $|\Phi^+\rangle$ in (1) and the corresponding density matrix $|\Phi^+\rangle\langle\Phi^+|$ in (2). Consider now the measurement of the first qubit with respect to the basis consisting of $|0_1\rangle$ and $|1_1\rangle$. The corresponding QO is

$$|0_1\rangle \_ |0_1\rangle + |1_1\rangle \_ |1_1\rangle,$$

where, for example, $|0_1\rangle$ is concretely given by

$$|0_1\rangle = |0\rangle \otimes I_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$
Here we followed the lexicographic indexing convention (Notation 2.2). Applying the measurement to the Bell state $|\Phi^+\rangle$—i.e. applying the QO in (5) to the density matrix in (2)—results in the following density matrix.

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]

\[
\frac{1}{2} \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
+ \frac{1}{2} \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
= \frac{1}{2} |0\rangle\langle 0| + \frac{1}{2} |1\rangle\langle 1|.
\]

This density matrix represents the ensemble $[|0\rangle \mapsto \frac{1}{2}, |1\rangle \mapsto \frac{1}{2}]$, or $[|0_2\rangle \mapsto \frac{1}{2}, |1_2\rangle \mapsto \frac{1}{2}]$ to be more explicit about which qubit in the original system we are referring to.

Two remarks are in order. Firstly, in the ensemble we have obtained, the first qubit in the original system has been discarded. This is a matter of choice: we could use a different QO that does retain the first qubit, resulting in the density matrix

\[
\frac{1}{2} \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]

that corresponds to the ensemble $[|0_10_2\rangle \mapsto \frac{1}{2}, |1_11_2\rangle \mapsto \frac{1}{2}]$.

Our choice above is because of the type $\text{qbit} \rightarrow \text{bit}$ (rather than $\text{qbit} \rightarrow \text{bit} \otimes \text{qbit}$) of the measurement primitive in our calculus. The second remark is numbered for future reference.

Remark 2.11. For the purpose of denotational semantics introduced later, we find it useful to split up a measurement into two separate “projections,” each of which corresponds to a possible outcome of the measurement. For example, the QO in (5) would rather be thought of as a pair of projection QOs

\[
\langle 0_1| \_| 0_1\rangle \quad \text{and} \quad \langle 1_1| \_| 1_1\rangle.
\]

The two projection QOs describe “what happens to the quantum state when the measurement outcome is $|0_1\rangle$ (or $|1_1\rangle$, respectively).” Having them separate allows us to model classical control structures that rely on the outcome of quantum measurements. This point will be more evident in §4.

Given a density matrix $\rho$, the probability for observing $|0_1\rangle$ or $|1_1\rangle$ can then be calculated as

\[
\text{tr}(\rho |0_1\rangle \langle 0_1|) \quad \text{or} \quad \text{tr}(\rho |1_1\rangle \langle 1_1|),
\]

respectively. For example, in the special case where $\rho = |v\rangle\langle v|$

\[
\text{tr}(\rho |0_1\rangle \langle 0_1|) = \text{tr}(|v\rangle\langle v| (|0_1\rangle \langle 0_1|)^T) = \|0_1|v\| \|^2.
\]

See [37, §8.2] for more details. This way we let density matrices implicitly carry probabilities (specifically by their trace values). This is why we allow density matrices with trace less than 1.
We extend the order \( \sqsubseteq \) in Def. 2.4 in a pointwise manner to obtain an order between QOs. This is done also in [7].

**Definition 2.12 (Order \( \sqsubseteq \) on \( \text{QO}_{m,n} \)).** Given \( E, F \in \text{QO}_{m,n} \), we define \( E \sqsubseteq F \) if and only if \( E(\rho) \sqsubseteq F(\rho) \) for each \( \rho \in \text{DM}_m \). The latter \( \sqsubseteq \) is the Löwner partial order (Def. 2.4).

**Proposition 2.13.** The order \( \sqsubseteq \) on \( \text{QO}_{m,n} \) is an \( \omega \)-CPO.

**Proof.** See Appendix AppendixA; also [7, Lem. 6.4]. \( \square \)

For illustration, notice that for any density matrix \( \rho \),
\[
\langle 0_1 | \rho | 0_1 \rangle \sqsubseteq \langle 0_1 | \rho | 0_1 \rangle + \langle 1_1 | \rho | 1_1 \rangle \quad \text{and} \quad \langle 1_1 | \rho | 1_1 \rangle \sqsubseteq \langle 0_1 | \rho | 0_1 \rangle + \langle 1_1 | \rho | 1_1 \rangle ,
\]
in the setting of (6). This establishes
\[
\langle 0_1 | 1_1 \rangle \sqsubseteq \langle 0_1 | 0_1 \rangle + \langle 1_1 | 1_1 \rangle \quad \text{and} \quad \langle 1_1 | 1_1 \rangle \sqsubseteq \langle 0_1 | 0_1 \rangle + \langle 1_1 | 1_1 \rangle ,
\]
where \( \sqsubseteq \) is the order of Def. 2.12. This example is prototypical of our use of the Löwner partial order (Def. 2.4 & 2.12): \( E \sqsubseteq F \) means that \( E \) is a projection (or “partial measurement”) that is a “component” of \( F \).

### 2.2. Monads for Branching

The notion of monad is standard in category theory. In computer science, after Moggi [38], the notion has been used for encapsulating computational effect in functional programming. One such monad denoted by \( T \) appears in this paper—at the last stage, as part of our categorical model.

There is another monad \( Q \)—called the quantum branching monad—that marks the beginning of our development. It is introduced in §4. The idea is drawn from the coalgebraic study of state-based systems (see e.g. [39, 40] for introduction); in particular from the use of a monad \( B \) on \( \text{Sets} \) for modeling branching, e.g. in [29].

**Example 2.14.** We list some examples of such “branching monads” \( B \).

- **The lift monad**
  \[
  \mathcal{L} \ X = 1 + X
  \]
  models potential nontermination. Its unit \( \eta^\mathcal{L} : X \to 1 + X \) and multiplication \( \mu^\mathcal{L} : 1 + (1 + X) \to 1 + X \) are obvious.

- **The powerset monad**
  \[
  \mathcal{P} \ X = \{ X' \subseteq X \}
  \]
  models nondeterminism. Its unit \( \eta^\mathcal{P} : X \to \mathcal{P} \ X \) returns a singleton set and its multiplication \( \mu^\mathcal{P} : \mathcal{P} \mathcal{P} \ X \to \mathcal{P} \ X \) takes the union.
The subdistribution monad

\[ \mathcal{D}X = \{ c : X \rightarrow [0,1] \mid \sum_x c(x) \leq 1 \} \]

models probabilistic branching. Its unit \( \eta^\mathcal{D} : X \rightarrow \mathcal{D}X \) carries \( x \in X \) to the so-called Dirac distribution \( [x \mapsto 1] \); and its multiplication \( \mu^\mathcal{D} : \mathcal{D}(\mathcal{D}X) \rightarrow \mathcal{D}X \) “suppresses” a distribution over distributions into a distribution (see also (8) below):

\[ \mu^\mathcal{D}(\xi) = \lambda x. \sum_{c \in \mathcal{D}X} \xi(c) \cdot c(x) . \]

The monad structures (units and multiplications) of the operations \( \mathcal{L}, \mathcal{P}, \mathcal{D} \) in the above list have a natural meaning in terms of branching. Among others, a multiplication \( \mu \) collapses “branching twice” into “branching once,” abstracting the internal branching structure. For example, \( \mathcal{P} \)'s multiplication

\[ \mu^\mathcal{P}_X : \mathcal{P}\mathcal{P}X \rightarrow \mathcal{P}X , \quad \text{like } \{ \{ x, y \}, \{ z \} \} \mapsto \{ x, y, z \} \]

can be understood as follows.

![Diagram of \( \mu^\mathcal{P}_X \) operation]

For \( \mathcal{D} \), its multiplication

\[ \mu^\mathcal{D}_X : \mathcal{D}\mathcal{D}X \rightarrow \mathcal{D}X , \quad \text{like } \begin{bmatrix} x \mapsto 1/2 \\ y \mapsto 1/2 \\ z \mapsto 1 \end{bmatrix} \mapsto \begin{bmatrix} 1/3 \\ 2/3 \\ 1/3 \end{bmatrix} \mapsto \begin{bmatrix} x \mapsto 1/6 \\ y \mapsto 1/6 \\ z \mapsto 2/3 \end{bmatrix} \]

can be understood as follows.

![Diagram of \( \mu^\mathcal{D}_X \) operation]

Furthermore, these monads come with natural order structures which turn out to be \( \omega \)-CPOs. This is exploited in [29] to prove—using a domain-theoretic technique from [41]—that a final coalgebra in \( \mathcal{K}\ell(B) \) coincides with an initial algebra in \( \text{Sets} \). The final coalgebra in \( \mathcal{K}\ell(B) \) thus identified provides a fully abstract semantic domain for trace semantics—execution trace-based (i.e. “linear-time”) semantics for state-based systems that is coarser than (“branching-time”) bisimilarity. See [29].
2.3. Geometry of Interaction

Girard’s Geometry of Interaction (GoI) [20] is an interpretation of proofs in linear logic in terms of dynamic information flow. It seems GoI’s position as a tool in denotational semantics is close to that of the game-based interpretations of computation [22, 23]. Its original formulation [20] utilizes a $C^*$-algebra; later in [24] the same idea is given a more concrete operational representation which is now commonly called token machines. For an introduction to GoI, our favorite reference is [42].

Besides these presentations of GoI by $C^*$-algebras and token machines, particularly important for our developments is the categorical axiomatization of GoI by Abramsky, Haghverdi and Scott [21]. They isolated some axiomatic properties of a category $C$ on which one can build a GoI interpretation. Such a category $C$ (together with some auxiliary data) is called a GoI situation in [21]; among other conditions, a crucial one is that $C$ is a traced symmetric monoidal category (TSMC) [43]. Then applying what they call the GoI construction $G$—it is isomorphic to the Int-construction in [43]—one is led to a compact closed category $G(C)$ of “bidirectional computations” or “(stateless) games.”

The resulting category $G(C)$ comes close to a categorical model of linear logic—a so-called linear category [30, 31]—but not quite, lacking an appropriate operator that models the ! modality of linear logic. A step ahead is taken in [21]: they extract a linear combinatory algebra (LCA) from $G(C)$. The notion of LCA is a variation of partial combinatory algebra (PCA) and corresponds to a Hilbert-style axiomatization of linear logic, including the ! modality (see Def. 4.9 later).

Remark 2.15 (Three “traces”). In this paper we use three different notions of trace. One is the trace operator in linear algebra; in quantum mechanics a probability for a certain observation outcome is computed by “tracing out” a density matrix, like in (7). Another “trace” is in trace semantics in the context of process theory; see §2.2. The other is in traced monoidal categories that play a central role in categorical GoI [21].

These three notions are not unrelated. The first “linear algebra trace” is an example of the last “monoidal trace”: namely in the category $\mathbf{fdVect}$ of finite-dimensional vector spaces and linear maps where the monoidal structure is given by the tensor product $\otimes$ of vector spaces. The second trace—which we would like to call “coalgebraic trace”—also yields an example of “monoidal trace.” This result, shown in [33], will be exploited for construction of a traced monoidal category $K(\mathcal{Q})$ on which we run the machinery of categorical GoI. See §4.

2.4. Realizability

Roughly speaking, an LCA can be thought of as a collection of untyped closed linear $\lambda$-terms. LCAs are, therefore, for interpreting untyped calculi.

What turns such a combinatory algebra into a model of a typed calculus is the technique of realizability. It dates back to Kleene; and its use in denotational semantics of programming languages is advocated e.g. in [34]. We shall be based
on its formulation found in [32]. It goes as follows. Starting from an LCA \( A \), we define the category \( \text{PER}_A \) of partial equivalence relations (PERs) on \( A \); a PER on \( A \) is roughly a subset of \( A \) with some of its elements mutually identified. An arrow of \( \text{PER}_A \) is represented by a code \( c \in A \).

To turn \( \text{PER}_A \) into a model of a typed linear \( \lambda \)-calculus (more specifically into a linear category) one needs type constructors like \( \otimes \), \( \triangleright \) and ! on \( \text{PER}_A \). They can be introduced by “programming in untyped linear \( \lambda \)-calculus”—it is much like encoding pairs, natural numbers, coproducts, etc. in the (untyped) \( \lambda \)-calculus \((x, y) := \lambda z. zxy \), with a first projection \( \lambda w. w(\lambda xy.x) \), and so on). More details can be found later in this paper; see also [32].

This linear version of realizability has been worked out e.g. in [32, 35]. The outcome of this construction is a model of a typed linear \( \lambda \)-calculus—i.e. a model of linear logic. There is a body of literature that seeks for what the latter means exactly—including [44, 45, 31, 30]—and there have been a few different notions proposed. It now seems that: the essence lies in what is called a linear-non-linear adjunction between a symmetric monoidal closed category and a CCC; and that the different notions of model proposed earlier in the literature are different constructions of such an adjunction. See the extensive survey in [46]; also [9, §9.6].

Among those notions of “model of linear logic,” in this paper we use the notion of linear category [30, 31] since its relationship to linear realizability has already been worked out in [32].

3. The Language Hoq

Here we introduce our target calculus. It is a variant of Selinger and Valiron’s quantum \( \lambda \)-calculus [9]. The calculus shall be called Hoq—for higher-order quantum computation.

The (only) major difference between Hoq and the calculus in [9] is separation of two tensors \( \otimes \) and \( \boxtimes \).

- The former Hilbert space tensor \( \otimes \) denotes, as usual in quantum mechanics, the tensor product \( \mathcal{H}_1 \otimes \mathcal{H}_2 \) of Hilbert spaces and designates compound quantum systems.

- We use the latter linear logic tensor \( \boxtimes \) for the “multiplicative and” connective in linear logic (hence in a linear \( \lambda \)-calculus). It is also denoted by \( \otimes \) commonly in the literature; but we choose to use the symbol \( \boxtimes \).

In fact, in Hoq the Hilbert space tensor \( \otimes \) will not be visible since we let \( n\text{-qbit} \) stand for \( \text{qbit} \otimes n \). The difference between \( n\text{-qbit} \boxtimes m\text{-qbit} \) and \( (n + m)\text{-qbit} \)

---

4Another standard technique is to use \( \omega \)-sets (also called assemblies) in place of PERs. This has been done for LCAs too; see [35].

5Hoq is a minor modification of the calculus q\( \lambda \) that we used in the conference version [1] of the current paper.
is: the former stands for two \((n\text{- and } m\text{-qubit})\) quantum states that are \textit{for sure} not entangled; the latter is for the composite system in which two states are \textit{possibly} entangled.

In contrast, in \cite{ref9} they use the same tensor operator \(\otimes\) for both—that is, the linear logic tensor is interpreted using the Hilbert space tensor. The reason for this difference will be explained in §3.3, as well as the design choices that we share with \cite{ref9}.

In this section we first introduce the syntax (including the type system) of Hoq in §3.1, followed by the operational semantics (§3.2). Then in §3.3 we discuss our design choices, especially the reason for the difference from the calculus in \cite{ref9}. In §3.4 we establish some properties on Hoq, including some safety properties such as substitution, subject reduction and progress.

### 3.1. Syntax

**Definition 3.1 (Types of Hoq).** The \textit{types} of Hoq are:

\[
A, B \::= \ n\text{-qbit} \mid !A \mid A \rightarrow B \mid \top \mid A \otimes B \mid A + B ,
\]

with conventions \(\text{qbit} \equiv 1\text{-qbit}\) and \(\text{bit} \equiv \top + \top\).

Here \(n \in \mathbb{N}\) is a natural number.

**Definition 3.2 (Terms of Hoq).** The \textit{terms} of Hoq are:

\[
M, N, P \::= x \mid \lambda x^A M \mid MN \mid \langle M, N \rangle \mid \text{let}(x^A, y^B) = M \text{ in } N \mid * \mid \text{let } * = M \text{ in } N \mid \text{inj}^B M \mid \text{inj}^A M \mid \text{match } P \text{ with } (x^A \mapsto M \mid y^B \mapsto N) \mid \text{letrec } f^A x = M \text{ in } N \mid \text{new } [\text{meas}_i^{n+1} U \mid \text{cmp}_{m,n}] ,
\]

with conventions \(\text{tt} \equiv \text{inj}^A(*)\) and \(\text{ff} \equiv \text{inj}^B(*)\).

Here \(m, n \in \mathbb{N}\) and \(i \in [1, n + 1]\) are natural numbers; \(U\) is a \(2^k \times 2^k\) unitary matrix, for some \(k \in \mathbb{N}\); and \(A\) and \(B\) are \textit{type labels}. The terms are almost the same as in \cite{ref9}: \texttt{newtt} designates preparation of the qubit \(|0\rangle\); \texttt{newff} is for \(|1\rangle\). The additional \textit{composition} operator \texttt{cmp} will have the type \(m\text{-qbit} \otimes n\text{-qbit} \rightarrow (m + n)\text{-qbit}\) and embed nonentangled states as possibly entangled states. For measurements we have operators \(\text{meas}_1, \text{meas}_2, \ldots, \text{meas}_{n+1}\) takes an \((n + 1)\text{-qubit}\) system, measures its \(i\text{-th}\) qubit, and returns the outcome (in the \texttt{bit} type) as well as the remaining quantum state that consists of \(n\) qubits.

The set \(\text{FV}(M)\) of \textit{free variables} in \(M\) is defined in the usual manner.

**Definition 3.3 (Subtype relation \(<\): in Hoq).** For typing in Hoq we employ the same subtype relation \(<\): as in \cite{ref9} and implicitly track the \(!\) modality.
(see §3.3). The rules that derive \( \ll <: \) are as follows.

\[
\begin{align*}
\text{\( \ll^n \text{k-bit} <: \!^m \text{k-bit} \)} & \quad \text{\( \ll^n \top <: \!^m \top \)} \quad (\top) \\
\frac{A_1 \ll B_1 \quad A_2 \ll B_2 \quad n = 0 \Rightarrow m = 0}{\!^n (A_1 \otimes A_2) \ll \!^m (B_1 \otimes B_2)} & \quad (\otimes) \\
\frac{A_1 \ll B_1 \quad A_2 \ll B_2 \quad n = 0 \Rightarrow m = 0}{\!^n (A_1 + A_2) \ll \!^m (B_1 + B_2)} & \quad (+) \\
\frac{B_1 \ll A_1 \quad A_2 \ll B_2 \quad n = 0 \Rightarrow m = 0}{\!^n (A_1 \rightarrow A_2) \ll \!^m (B_1 \rightarrow B_2)} & \quad (\rightarrow)
\end{align*}
\]

All the rules come with a condition \( n = 0 \Rightarrow m = 0 \), which is equivalent to \( m = 0 \lor n \geq 1 \).

**Definition 3.4 (Typing in Hoq).** The typing rules of Hoq are as in Table 1. Here \( \Delta, \Gamma \), etc. denote (unordered) contexts. Given a context \( \Delta = (x_1 : A_1, \ldots, x_m : A_m) \),

- \( ! \Delta \) denotes the context \( (x_1 : !A_1, \ldots, x_m : !A_m) \); and
- \( |\Delta| := \{x_1, \ldots, x_m\} \) is the domain of \( \Delta \).

When we write \( \Delta, \Gamma \) as the union of two contexts, we implicitly require that \( |\Delta| \cap |\Gamma| = \emptyset \). In the rule (Ax.2), \( c \) is a constant and its default type \( \text{DType}(c) \) is defined as follows.

\[
\begin{align*}
\text{DType}(\text{new}) & : \equiv \text{bit} \rightarrow \text{qbit} \\
\text{DType}(\text{meas}_{n+1}) & : \equiv (n+1)-\text{qbit} \rightarrow (\text{bit} \otimes n-\text{qbit}) \quad \text{for} \ n \geq 1 \\
\text{DType}(\text{meas}_1) & : \equiv \text{qbit} \rightarrow !\text{bit} \\
\text{DType}(U) & : \equiv k-\text{qbit} \rightarrow k-\text{qbit} \quad \text{for a} \ 2^k \times 2^k \ \text{unitary matrix} \ U \\
\text{DType}(\text{cmp}_{m,n}) & : \equiv (m-\text{qbit} \otimes n-\text{qbit}) \rightarrow (m+n)-\text{qbit}
\end{align*}
\]

We shall write \( \Pi \vdash \Delta \vdash M : A \) if a derivation tree \( \Pi \) derives the type judgment. We write \( \not\vdash \Delta \vdash M : A \) if there exists such \( \Pi \), that is, the type judgment is derivable.

### 3.2. Operational Semantics

First we introduce small-step operational semantics, from which we derive big-step one. The latter is given in the form of probability distributions over the bit type and is to be compared with the denotational semantics.

**Definition 3.5 (Extended Hoq).** For the purpose of operational semantics, we extend Hoq-terms by the following additional set of constants:

\[
\text{qstate}_\rho \quad \text{for each} \ k \in \mathbb{N} \ \text{and} \ \rho \in \text{DM}_{2^k}.
\]

Their default types are:

\[
\text{DType}(\text{qstate}_\rho) : \equiv k-\text{qbit} \quad \text{for} \ \rho \in \text{DM}_{2^k}.
\]
\[
\frac{A <: A'}{(\text{Ax.1})} \quad \frac{\text{!DType}(e) <: A}{\Delta \vdash e : A} \quad (\text{Ax.2})
\]
\[
\frac{x : A, \Delta \vdash M : B \quad A' <: A}{\Delta \vdash \lambda x^A.M : A' \to B} \quad (\rightarrow_1)
\]
\[
\frac{x : A, \Delta, \Gamma \vdash M : B \quad \text{FV}(M) \subseteq |\Delta| \cup \{x\} \quad A' <: A}{\Delta, \Gamma \vdash \lambda x^A.M : !^n(A' \to B)} \quad (\rightarrow_2)
\]
\[
\frac{\Delta, \Gamma_1 \vdash M : A \to B \quad \Delta, \Gamma_2 \vdash N : C \quad C <: A}{\Delta, \Gamma_1, \Gamma_2 \vdash MN : B} \quad (\rightarrow_\text{E})
\]
\[
\frac{\Delta, \Gamma_1 \vdash M_1 : !^n A_1 \quad \Delta, \Gamma_2 \vdash M_2 : !^n A_2}{\Delta, \Gamma_1, \Gamma_2 \vdash (M_1, M_2) : !^n(A_1 \otimes A_2)} \quad (\otimes_1)
\]
\[
\frac{\Delta, \Gamma_1 \vdash M : !^n(A_1 \otimes A_2) \quad \Delta, \Gamma_2, x_1 : !^n A_1, x_2 : !^n A_2 \vdash N : A}{\Delta, \Gamma_1, \Gamma_2 \vdash \text{let } (x_1, x_2) = M \text{ in } N : A} \quad (\otimes_\text{E})
\]
\[
\frac{\Delta \vdash * : !^n \top}{(\top_1)} \quad \frac{\Delta \vdash \text{let } * = M \text{ in } N : A}{\Delta, \Gamma_1, \Gamma_2 \vdash N : A} \quad (\top_\text{E})
\]
\[
\frac{\Delta \vdash M : !^n A_1 \quad A_2 <: A'_2}{(+1)} \quad \frac{\Delta \vdash N : !^n A_2 \quad A_1 <: A'_1}{(+1_2)}
\]
\[
\frac{\Delta \vdash \text{inj}_1 A : !^n(A_1 + A_2)}{(+) \quad \Delta \vdash \text{inj}_2 A : !^n(A'_1 + A_2)}
\]
\[
\frac{\Delta, \Gamma \vdash P : !^n(A_1 + A_2) \quad \Delta, \Gamma', x_1 : !^n A_1 \vdash M_1 : B \quad \Delta, \Gamma', x_2 : !^n A_2 \vdash M_2 : B}{\Delta, \Gamma, \Gamma' \vdash \text{match } P \text{ with } (x_1, x_2) \to M_1 \mid (x_1, x_2) \to M_2 : B} \quad (+_\text{E})
\]
\[
\frac{\Delta, \Gamma \vdash f : (A \to B) \vdash x : A \vdash M : B \quad \Delta, \Gamma, f : (A \to B) \vdash N : C}{\Delta, \Gamma \vdash \text{letrec } f^{A \to B} x = M \text{ in } N : C \quad (\text{rec})}
\]

Table 1: Typing rules for Hoq
Remark 3.6. The term $qstate_\rho$ designates the (mixed) quantum state represented by the density matrix $\rho$; therefore $qstate_{|0\rangle\langle 0|}$ and new $t t$ designate the same thing. They are distinguished in that the former is a value while the latter is not; this is important in consideration of Hoq’s no-cloning property (see Rem. 3.21). We note that the term $qstate_\rho$ is not in the language Hoq itself but is an additional one used only in defining operational semantics. Therefore a programmer has no access to it.

Definition 3.7 (Value, evaluation context). The values and evaluation contexts of (extended) Hoq are defined in the following (mostly standard) way.

Values  
$V, V_1, V_2 ::= x \mid \lambda x^A.M \mid \langle V_1, V_2 \rangle \mid * \mid inj^B V \mid inj^A V \mid new \mid meas_i^{n+1} \mid U \mid cmp_{m,n} \mid qstate_\rho$ ;

Evaluation contexts  
$E ::= [\_] \mid E[\_] \mid E[\_M] \mid E[V\_] \mid E[(\_), M] \mid E[(V, \_)] \mid E[let \ (x^A, y^B) = \_ \_ in M] \mid E[let \_* = \_ \_ in N] \mid E[inj^B\_] \mid E[inj^A\_] \mid E[match \_ \_ with (x^A \rightarrow M \mid y^B \rightarrow N)]$ .

Here $E[F]$ is the result of replacing $E$’s unique hole $\_\_$ with the expression $F$.

As usual, all the constants (new, meas$_i^{n+1}$, and so on) are values.

The definition of evaluation context is “top-down.” A “bottom-up” definition is also possible and will be used in later proofs.

Lemma 3.8. The following BNF notation defines the same notion of evaluation context as in Def. 3.7.

$D ::= [\_] \mid DM \mid V D \mid \langle D, M \rangle \mid \langle V, D \rangle \mid \langle x^A, y^B \rangle = D \_ \_ in M \mid \_ \_ = D \_ \_ in N \mid inj^B D \mid inj^A D \mid match D with (x^A \rightarrow M \mid y^B \rightarrow N)$ .

Here $V$ is a value and $M, N$ are terms, as before. □

Definition 3.9 (Small-step semantics). The reduction rules of (extended) Hoq are defined as follows. Each reduction $\rightarrow$ is labeled with a real number
For each semantics for such terms.

Observe that the label $p$ in reduction $\to_p$ is like a probability but not quite: from $\text{meas}_{\lambda}^n(\text{qstate}_p)$ there are two $\to_1$ reductions, to $\text{qstate}_{0,\rho(0)}$ and to $\text{qstate}_{1,\rho(1)}$. We understand that the probabilities are implicitly carried by the trace values of the matrices $0,\rho(0)$ and $1,\rho(1)$. See (7) and the remarks that follow it.

As is standardly done, we will prove adequacy of our denotational semantics focusing on $\text{bit}$-type closed terms. For that purpose we now introduce big-step semantics for such terms.\footnote{We swapped the notations $\mathcal{Y}$ and $\mathcal{Y}$ from the previous version [1].}

**Definition 3.10 (Big-step semantics).** For each $n \in \mathbb{N}$ we define a relation $\mathcal{Y}^n$ between closed $\text{bit}$-terms $M$ and pairs $(p,q)$ of real numbers. This is by induction on $n$.

For $n=0$, we define

$$\text{tt} \ \mathcal{Y}^0 (1,0) , \ \text{ff} \ \mathcal{Y}^0 (0,1), \ \text{and} \ M \ \mathcal{Y}^0 (0,0)$$

for the other $M$.

For $n+1$, if $M$ has a reduction $M \to_1 M'$ caused by a rule other than the measurement rules, we set:

$$M \ \mathcal{Y}^{n+1} (p,q) \overset{\text{def}}{\iff} M' \ \mathcal{Y}^n (p,q) .$$
If \( M \) has a reduction \( M \longrightarrow_r N \) caused by one of the measurement rules, there is always its buddy reduction \( M \longrightarrow_r N' \). In this case we set
\[
M \uparrow^{n+1} (rp + r'p', rq + r'q') \overset{\text{def}}{=} N \uparrow^n (p, q) \text{ and } N' \uparrow^n (p', q') .
\]

Finally, we define a relation \( \uparrow \) as the supremum of \( \uparrow^n \). That is,
\[
M \uparrow (p, q) \overset{\text{def}}{=} (p, q) = \sup \{ (p', q') \mid M \uparrow^n (p', q') \text{ for some } n \}
\]
where sup is with respect to the pointwise order on \([0, 1] \times [0, 1]\). It is easy to see that for each \( M \) and \( n \), there is only one pair \((p, q)\) such that \( M \uparrow^n (p, q) \). The same holds for \( \downarrow \), too.

The intuition of \( M \uparrow (p, q) \) is: the term \( M \) (which is closed and of type \( \text{bit} \)) reduces eventually to \( \texttt{tt} \) with the probability \( p \); to \( \texttt{ff} \) with the probability \( q \).

**Remark 3.11.** The operational semantics of [9] employs the notions of quantum array and quantum closure—it thus has the flavor of a language with quantum stores (cf. [6]). This is the very key in their setup that allows for using the Hilbert space tensor \( \otimes \) as the linear logic tensor. We chose to separate the two tensors so that the ! modality and recursion can be smoothly accommodated using known techniques (namely GoI and realizability). Accordingly, our operational semantics for Hoq is much more simplistic without quantum arrays.

### 3.3. Design Choices

#### 3.3.1. What We Share with the Calculus of Selinger and Valiron

Our calculus Hoq share the following design choices with the original calculus in [9]:

- building on linear \( \lambda \)-calculus—in particular the enforcement no-cloning by a linear type discipline;
- a call-by-value reduction strategy;
- uniformity of data, in the sense that classical and quantum data are dealt with in the same manner;
- a formulation of the \texttt{letrec} operator, as is usually done in a call-by-value setting (namely, recursion is only at function types, see the (rec) rule in Table 1); and
- implicit linearity tracking.

The last means the following (see also [9]). Linear \( \lambda \)-calculi, including the one in [30, 31], typically have explicit syntax for operating on the ! modality. An example is the \texttt{derelict} operator in
\[
\Gamma \vdash M : !A \quad \Rightarrow \quad \Gamma \vdash \texttt{derelict} M : A
\]

In [9] a subtype relation \(<:\) is introduced so that such explicit operators can be dispensed with. For example, the subtype relation \(!A <: A\) replaces the \texttt{derelict} operator in the above. This design choice is intended to aid programmers; and we follow [9] with regard to this choice.
3.3.2. What Are Different

We now turn to the major difference from the original calculus, namely the separation of $\otimes$ from $\boxdot$ (mentioned already at the beginning of the current section). In [9] they use the same symbol $\otimes$ for both tensors; in other words, the linear logic tensor is interpreted using the Hilbert space tensor. This leads to their clean syntax: a 2-qubit system is naturally designated by the type $\text{qbit} \otimes \text{qbit}$; and this is convenient when we translate quantum circuits into programs. Moreover, their ingenious operational semantics—which carries the flavor of quantum store—allows such double usage of $\otimes$ (see Rem. 3.11).

However, in developing interaction-based denotational semantics, we found this double usage of $\otimes$ inconvenient. We would like the linear logic tensor interpreted in the same way as it is interpreted in the conventional interpretation of classical computation. This seems to be a natural thing to do when working with a language with “quantum data, classical control”—leaving the classical control part untouched. Moreover, there exists ample semantical machinery that provides natural interpretations of the operators like $!$ and $\boxdot$ and recursion that go along well with $\otimes$.

The latter is not easily the case with $\boxdot$. While the duality $\mathcal{H}^* \cong \mathcal{H}$ gives a compact closed structure (hence the interpretation of $\boxdot$) to the category $\text{fdHilb}$ of finite-dimensional Hilbert spaces, such is not available in the category $\text{Hilb}$ of general Hilbert spaces, on the one hand. On the other hand, $\text{Hilb}$ is a natural choice for a semantic domain: for interpreting the $!$ modality (“as many copies as requested”), we will need some kind of “infinity,” whose first candidate would be infinite-dimensional Hilbert spaces.\footnote{In [6, §1] it is argued—rather on a conceptual level—that the Hilbert space tensor $\otimes$ does not seem quite compatible with a closed structure (i.e. with respect to $\boxdot$).}

**Remark 3.12.** A different approach is taken in [14]. The work keeps the original language of [9]—where the monoidal and quantum tensors coincide—and starts from an axiomatic description of categorical models of the language. The latter is the notion of linear category for duplication [9] that combines a linear-non linear adjunction and monadic effects.

To construct a concrete instance of such models, the work [14] employs a series of constructions known in category theory, notable among which is cocomplete completion that embeds (via Yoneda) a monoidal category $\mathcal{C}$ in a monoidal closed category $[\mathcal{C}^{op}, \text{Sets}]$ [47]. The base category is $\mathcal{C} = \mathcal{Q}$ from [7] where arrows are essentially quantum operations and a monoidal structure is given by the quantum tensor $\otimes$.

The work [15] can be seen as a drastic simplification of the results in [14] by: 1) simplifying the calculus (but still maintaining coincidence of the monoidal and quantum tensors); and 2) using quantitative semantics for linear logic [16, 17] instead of completion by presheaves. The latter step is comparable to the simplification of Girard’s normal functor semantics [18] to quantitative semantics.

In comparison to this categorical and axiomatic approach, our approach to
a denotational model is operational, relying on intuitions from token abstract
machines and transition systems. This approach is served well by the categorical
formulation of geometry of interaction, and the theory of coalgebras as a catego-
rical theory of state-based dynamics. Moreover, it allows us to establish
correspondences to operational semantics (soundness and adequacy), a feature
that is lacking in [14].

**Remark 3.13.** In [15], the previous version [1] of the current paper is discussed,
and the authors say: “... the model drops the possibility of entangled states,
and thereby fails to model one of the defining features of quantum computation.”
We believe that this is not the case and the examples in §3.5 will convince the
reader. Entanglement may not be expressed by means of a type constructor,
but is certainly there.

Besides the separation of $\otimes$ from $\oplus$, Hoq’s difference from the calculus in [9]
is that bound variables and injections have explicit type labels (such as $A$ in
$\lambda x^A.M$). This choice is to ensure well-definedness of the interpretation $[\Delta \vdash M : A]$ of type judgments (Lem. 5.32)—a delicate issue with Hoq especially
because of the subtype relation $\llbracket < \rrbracket$.

**Remark 3.14 (Type labels and well-definedness of interpretations).** In
general a derivable type judgment $\Delta \vdash M : A$ can have multiple derivations.
Since denotational semantics is defined inductively on derivations, it is not al-
ways trivial if the interpretation $[\Delta \vdash M : A]$ is well-defined or not.

It is in fact nontrivial already for the simply typed $\lambda$-calculus in the Curry-
style (i.e. variables’ types are not predetermined but are specified in type con-
texts). An example is given by

$$x : A \vdash (\lambda y.x)(\lambda z.z) : A,$$

where the type of $z$ can be anything. When we turn to classical textbooks: in [48]
a Church-style calculus is used (variables come with their intrinsic types); in [49]
its Curry-style calculus has explicit type labels (much like in Hoq).

For the Curry-style simply typed $\lambda$-calculus, we can actually do without
explicit type labels and still maintain well-definedness of $[\Delta \vdash M : A]$. Its
proof can be given exploiting strong normalization of the calculus. The same
proof strategy is used in [50, Chap. 9–11]—where the strategy is identified as
normalization by evaluation—to prove the well-definedness of $[\Delta \vdash M : A]$ for
the quantum lambda calculus of [9]. (The proof is long and complicated,
reflecting the complexity of the calculus.) We believe the same strategy can be
employed and will get rid of type labels in Hoq; however the proof will be very
lengthy and it is therefore left as future work.

We also note that all these troubles would be gone if we adopt explicit lin-
earity tracking (explicit operators like derelict in (13), instead of the subtype
relation $\llbracket < \rrbracket$), and move to the Church-style (variables have their own types).
The reasons for not doing so are:
• we agree with [9] in that explicit tracking of the usage of the ! modality is a big burden to programmers; and

• our denotational model—based on GoI and realizability—has a merit of supporting implicit linearity tracking. This is not the case with every linear category, since we need certain type isomorphisms like $!!A \cong !A$ (see Lem. 5.3).

3.4. Syntactic and Operational Properties of Hoq

Here we establish some syntactic and operational properties of Hoq, including some safety properties such as substitution, subject reduction and progress. Although they are mostly parallel to [9, §9.3], syntax is fragile and we have to redo all the proofs. We shall defer most of the proofs to Appendix AppendixB.

Lemma 3.15 (Properties of the subtype relation $\prec$).
1. $\prec$ is a preorder.
2. $!$ is monotone: $A \prec B$ implies $!A \prec !B$.
3. If $n = 0 \Rightarrow m = 0$ holds, we have $!^n A \prec !^m A$.
4. Assume that $!^n A \prec !^m B$. If neither $A$ nor $B$ is of the form $!^n C$, we have $(n = 0 \Rightarrow m = 0)$ and $A \prec B$.
5. The relation $\prec$ has directed sups and infs. The former means the following (the latter is its dual). If $A_1 \prec A$ and $A_2 \prec A$, then there is a type $A_0$ such that: 1) $A_1 \prec A_0$ and $A_2 \prec A_0$; 2) $A_1 \prec A'$ and $A_2 \prec A'$ imply $A_0 \prec A'$.

□

Notation 3.16 (Subtyping $\prec$ between contexts). We write $\Delta' \prec \Delta$ when:
• $|\Delta'| = |\Delta|$, and
• for each $(x : A') \in \Delta'$, there is $(x : A) \in \Delta$ with $A' \prec A$.

In Table 1, some rules including $(-\otimes I_1)$ have type coercion: while the term $\lambda x : A'. M$ has a type label $A$, its actual type is $A' \otimes B$ with $A' \prec A$. This is so that the following holds.

Lemma 3.17. The monotonicity rule is admissible in Hoq.

\[
\Delta' \prec \Delta \quad \Delta \vdash M : A \quad A \prec A' \quad (\text{Mon})
\]

□

Corollary 3.18. The dereliction and comultiplication rules are admissible in Hoq.

\[
\begin{align*}
\Delta \vdash M : !A \\
\Delta \vdash M : A' \\
\Delta \vdash M : !!A
\end{align*}
\]

(Comult)

□

Lemma 3.19.
1. If $\vdash \Delta \vdash M : B$, then $\text{FV}(M) \subseteq |\Delta|$.
2. If $x \not\in \text{FV}(M)$ and $\vdash \Delta, x : A \vdash M : B$, then $\vdash \Delta \vdash M : B$.
3. The following rule is admissible.

\[
\begin{align*}
\Delta \vdash M : A \\
\Gamma, \Delta \vdash M : A
\end{align*}
\]

(Weakening)
Proof. Straightforward, by induction on derivation. □

Many linear lambda calculi have the promotion rule

\[
\frac{\,!\Delta, \Gamma \vdash M : A}{\,!\Delta, \Gamma \vdash \text{promote} M : !A} \quad \text{(Prom)}
\]

or its variant, like in [30, 31]. Much like the original calculus (see [9, Remark 9.3.27]), Hoq lacks the general promotion rule but it has a restriction to values admissible.

Lemma 3.20 (Value promotion). Let \( V \) be a value.

1. If \( \vdash \Delta \vdash V : !A \), then for each \( x \in \text{FV}(V) \), we have \( (x : !B) \in \Delta \) for some type \( B \).
2. Conversely, the following rule is admissible in Hoq.

\[
\frac{\,!\Delta, \Gamma \vdash V : A \quad \text{FV}(V) \subseteq |\Delta|}{\,!\Delta, \Gamma \vdash V : !A} \quad (\text{ValProm}) , \quad \text{where} \ V \ \text{is a value.}
\]

Remark 3.21. It is not hard to see that the term \texttt{newt} (preparation of the qubit \( |0\rangle \)) can have a type \texttt{qbit} but not \( !\texttt{qbit} \). In particular, since \texttt{newt} is not a value, it is not possible to apply the \( (\text{ValProm}) \) rule in Lem. 3.20 to obtain \( \vdash \texttt{newt} : !\texttt{qbit} \) from \( \vdash \texttt{newt} : \texttt{qbit} \). This fact—witnessing no-cloning—is due to the choice that \( \texttt{new} : \texttt{bit} \rightarrow \texttt{qbit} \) is a constant (hence as a value), instead of \texttt{newt} being a constant.\(^8\) See also Rem. 3.6.

A general substitution rule

\[
\frac{\,!\Delta, \Gamma_1 \vdash M : A \quad \,!\Delta, \Gamma_2, x : A \vdash N : B}{\,!\Delta, \Gamma_1, \Gamma_2 \vdash N[M/x] : B} \quad \text{(Subst)}
\]

is not admissible in Hoq. A counter example can be given as follows. The following two judgments are both derivable.

\[
x : \texttt{qbit} \rightarrow !\texttt{qbit}, y : \texttt{qbit} \vdash xy : !\texttt{qbit} \quad w : !\texttt{qbit} \vdash \lambda z^{!\texttt{qbit}}. w : !(!\texttt{qbit} \rightarrow !\texttt{qbit})
\]

In particular, the latter relies on the \( (\rightarrow, \text{I}_2) \) rule. However, the result of substitution

\[
x : \texttt{qbit} \rightarrow !\texttt{qbit}, y : \texttt{qbit} \vdash \lambda z^{!\texttt{qbit}}. xy : !(!\texttt{qbit} \rightarrow !\texttt{qbit})
\]

is not derivable: since the types of the free variables \( x, y \) are not of the form \( !\Delta \), the \( (\rightarrow, \text{I}_2) \) rule is not applicable. Therefore we impose some restrictions.

\(^8\)In the conference version [1] we had this wrong. Thanks are due to Peter Selinger for pointing out.
Lemma 3.22 (Substitution). The following rules are admissible in Hoq.

\[
\begin{align*}
\frac{!\Delta, \Gamma_1 \vdash M : A \quad !\Delta, \Gamma_2, x : A \vdash N : B \quad A \neq A'}{(Subst_1)} & \quad (Subst_1) \\
\frac{!\Delta \vdash M : A \quad !\Delta, \Gamma_2, x : A \vdash N : B}{!\Delta, \Gamma_1, \Gamma_2 \vdash N[M/x] : B} & \quad (Subst_2) \\
\frac{!\Delta \vdash V : A \quad !\Delta, \Gamma_2, x : A \vdash N : B}{!\Delta, \Gamma_1, \Gamma_2 \vdash N[V/x] : B} & \quad (Subst_3) \\
\frac{!\Delta, \Gamma_1 \vdash M : A \quad !\Delta, \Gamma_2, x : A \vdash E[x] : B \quad x \text{ does not occur in } E}{!\Delta, \Gamma_1, \Gamma_2 \vdash E[M] : B} & \quad (Subst_4)
\end{align*}
\]

Note that in the first assumption of the (Subst_2) rule, the whole context must be of the form $!\Delta$. In the (Subst_4) rule $E$ denotes an evaluation context (Def. 3.7); the side condition means that $x$ occurs exactly once in the term $E[x]$. □

Lemma 3.23 (Subject reduction). Assume that $\not\vdash \Delta \vdash M : A$, and that there is a reduction $M \rightarrow_p N$ (we allow $p = 0$). Then $\not\vdash \Delta \vdash N : A$. □

Lemma 3.24 (Progress). A typable closed term that is not a value has a reduction. More precisely: assume that $\not\vdash \Delta \vdash M : A$ (therefore $M$ is closed by Lem. 3.19.1), and that $M$ is not a value. Then there exists a term $N$ and $p \in [0, 1]$ such that $M \rightarrow_p N$. □

We note that, given $M$, the sum of the values $p$ for all possible reductions $M \rightarrow_p N$ is not necessarily equal to 1. See the remark right after Def. 3.9.

3.5. Examples of Hoq Programs

3.5.1. Quantum Teleportation

We give an example of a Hoq program that simulates quantum teleportation: a procedure in which Alice sends a quantum state to Bob using a classical communication channel.

They start with preparing an EPR-pair. Alice keeps the first qubit of the EPR-pair; and Bob keeps the second qubit. In Hoq, we can prepare an EPR-pair by applying a suitable unitary transformation to a qubit constructed by new:

\[
\text{EPR} \equiv U_0(\text{cmp}_{1,1}(\text{new tt, new tt})) : 2\text{-qbit}
\]

where $U_0$ is a unitary transformation given by

\[
U_0 := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix}.
\]

We note that $U_0$ is usually defined as the following composition of the Hadamard gate $H$ and the conditional-not gate $N$:

\[
U_0 = N(H \otimes I_2),
\]
where $I_2$ is the $2 \times 2$ identity matrix. However, since the tensor product $\boxtimes$ of Hoq is different from the tensor product $\otimes$ of vector spaces, we cannot program $U_0$ as a simple composition of $H$ and $N$ in Hoq. That is, we need to calculate $U_0$ outside Hoq. In fact it is straightforward to equip Hoq with additional language constructs that aid composition of unitary operators. We do not do so in this paper, since our focus is on the integration of classical and quantum information using the GoI and realizability techniques.

Then Alice applies a Bell measurement to the first two qubits of a quantum state $\rho \boxtimes EPR$ where $\rho$ is a quantum bit that Alice wishes to send to Bob.

Bell measure $\equiv \lambda w.3$-qbit. let $\langle b^\text{bit}_0, \rho^2\text{-qbit} \rangle = \text{meas}_3^3(U_1 w)$ in

let $\langle b^\text{bit}_1, q^\text{qbit} \rangle = \text{meas}_3^2 \rho$ in $\langle b^\text{bit}_0, \langle b^\text{bit}_1, q^\text{qbit} \rangle \rangle$: 3-qbit $\rightarrow$ bit $\boxtimes$ (bit $\boxtimes$ qbit)

(14)

Here $U_1$ is the following unitary transformation:

$$U_1 := \frac{1}{\sqrt{2}} \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -1 & 0 & 0
\end{pmatrix},$$

which is equal to $((H \otimes I_2)N) \otimes I_2$. Although the third qubit has nothing to do with the Bell measurement, we need to include the third qubit in the program (14) because the type 2-qbit $\boxtimes$ qbit is different from 3-qbit in Hoq. This kind of awkwardness will also be gone, once we equip Hoq with constructs for composing unitary operations.

Alice tells the result $(i, j)$ of measurement to Bob, and Bob applies a unitary transformation $U_{i,j}$ to his qubits:

corr $\equiv \lambda x.\text{bit}(\text{bit}\boxtimes\text{qbit}). \text{let } (b^\text{bit}_0, y^\text{bit}\boxtimes\text{qbit}) = x \text{ in let } (b^\text{bit}_1, q^\text{qbit}) = y \text{ in }$

match $b_0$ with (z$^\top_0 \mapsto$ match $b_1$ with (w$^\top_1 \mapsto$ U$^0_{i,j}$ q | w$^\top_1 \mapsto$ U$^1_{i,j}$ q))

| z$^\top_1 \mapsto$ match $b_1$ with (w$^\top_1 \mapsto$ U$^{10}_{i,j}$ q | w$^\top_1 \mapsto$ U$^{11}_{i,j}$ q))

: bit $\boxtimes$ (bit $\boxtimes$ qbit) $\rightarrow$ qbit

where $U_{i,j}$ are given as follows.

$$U_{00} := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad U_{01} := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad U_{10} := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad U_{11} := \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

The result is the qubit that Alice wishes to send to Bob.
We combine the above programs into one: we define a closed value \( qtel : \text{qbit} \to \text{qbit} \) to be
\[
\lambda x. \text{corr} (\text{Bellmeasure} (\text{cmp}_{1,2} (x, \text{EPR})))
\]
We can observe that Bob receives Alice’s qubit.

**Proposition 3.25.** For any unitary transformation \( U \), the reduction tree of \( qtel \) is of the following form.

\[
\text{qtel}(U(\text{new tt})) \\
\begin{array}{c}
\begin{array}{c}
\text{qstate}_\rho \text{tt} \\
1 \\
\ast \\
\ast \\
\ast \\
\ast \\
\text{qstate}_\rho \text{tt} \\
\text{qstate}_\rho \text{tt} \\
\text{qstate}_\rho \text{tt} \\
\text{qstate}_\rho \text{tt}
\end{array}
\end{array}
\]

Here \( \rho \in \text{DM}_2 \) is \( U \ket{0} \bra{0} U^\dagger \) and \( \rightarrow_1 \) is the transitive closure of \( \rightarrow_1 \).

**3.5.2. Fair Coin Toss**
This example is in [9]: it simulates a fair coin toss with quantum primitives.

\[
\text{fcoin} \equiv H(\text{new tt}) \quad \text{toss} \equiv \lambda x. \text{meas}^1_1 x
\]

**Proposition 3.26.** The reduction tree of \( \text{toss} \text{ fcoin} \) is the following one.

\[
\begin{array}{c}
\begin{array}{c}
\text{toss} \text{ fcoin} \\
\frac{1}{1} \\
\frac{1}{1} \\
\frac{1}{1} \\
\frac{1}{1} \\
\frac{1}{1} \\
\frac{1}{1} \\
\frac{1}{1}
\end{array}
\end{array}
\]

Here we omitted reductions \( \rightarrow_1 \).

**3.5.3. Grover’s Algorithm**
We give a program that executes a quantum searching algorithm, called Grover’s algorithm. Since the algorithm has a process “repeat ... for \( n \) times”, we first add natural numbers to Hoq.

We extend Hoq by adding a type of natural numbers, as follows.

| Type   | \( A \) ::= \cdots | \text{nat} |
|--------|-------------------|
| Term   | \( M \) ::= \cdots | n \in \mathbb{N} \mid \text{isZero} \mid \text{pred} |
| Value  | \( V \) ::= \cdots | n \in \mathbb{N} \mid \text{isZero} \mid \text{pred} |
| Evaluation context | \( E \) ::= \cdots | E[S[-]] \mid E[\text{isZero}[-]] \mid E[\text{pred}[-]] |

Default types of these constants are given as follows.

\[
\begin{array}{ll}
\text{DType}(n) = \text{nat} & \text{DType}(S) = \text{nat} \to \text{nat} \\
\text{DType}(\text{isZero}) = \text{nat} \to \text{bit} & \text{DType}(\text{pred}) = \text{nat} \to \text{nat}
\end{array}
\]

26
Operational semantics is extended in the standard way.

\[
\begin{align*}
E[S\ n] &\to E[n + 1] & E[\text{isZero} \ 0] &\to E[\text{tt}] \\
E[\text{isZero} \ (n + 1)] &\to E[\text{ff}] & E[\text{pred} \ n] &\to E[n - 1]
\end{align*}
\]

Here \( n - 1 \) denotes normalized subtraction: it is \( n - 1 \) if \( n > 0 \); and \( 0 - 1 := 0 \).

**Remark 3.27.** Since \( \text{PER}_Q \) has a natural number object, we can extend the interpretation of \( \text{Hoq} \) in a canonical manner. Furthermore, Cor. 6.7 holds for the extension.

In this extension of \( \text{Hoq} \) we express Grover’s algorithm. We suppose that there is an oracle \( U_1 : n\text{-qbit} \to n\text{-qbit} \) that tells us which quantum state is an answer \( w \in \{0, 1\}^n \):

\[
U_1[n] = \begin{cases} 
|n\rangle & (n \neq w), \\
-|w\rangle & (n = w).
\end{cases}
\]

The goal of Grover’s algorithm is to find the answer \( w \). The algorithm is described as follows.

1. Prepare a quantum state

\[
|s\rangle = 2^{-\frac{n}{2}} \sum_{i \in \{0, 1\}^n} |i\rangle.
\]

2. Do the following clauses sufficiently many times.
   (a) Apply \( U_1 \) to the quantum state.
   (b) Apply \( U_2 = 2|s\rangle\langle s| - I \) to the quantum state.
3. Observe the quantum state.

The result of observation will be \( w \) with a high probability: given a desired degree of precision, there is a formula that computes how many iterations of Step 2 are needed.

We can express the above procedure as follows:

\[
\text{Grover} \equiv \lambda u^{(n\text{-qbit} \to n\text{-qbit})}. \lambda q^{n\text{-qbit}}. \lambda m^{\text{nat}}. \text{meas} \ (\text{repeat} \ u \ m \ q)
\]

\[
\text{repeat} \ u, m, q \text{ computes } u^m(q). \text{ That is,}
\]

\[
\text{repeat} \equiv \lambda f^{(n\text{-qbit} \to n\text{-qbit})}. \lambda m^{\text{nat}}. \lambda x^{n\text{-qbit}}.
\]

\[
\text{letrec } g^{\text{nat} \to n\text{-qbit}} k = \text{match isZero } k \text{ with } (y^\top \mapsto x \mid z^\top \mapsto f(g(\text{pred} k)))
\]

\[
in \ g \ m
\]

\[
\text{: } !(n\text{-qbit} \to n\text{-qbit}) \to !\text{nat} \to n\text{-qbit} \to n\text{-qbit}
\]
• \text{meas}_n \text{ measures } n\text{-qbit:}

\text{meas}_n \equiv \lambda q^n\text{-qbit. let } \langle b_{1}^{\text{bit}}, q_1^{(n-1)\text{-qbit}} \rangle = \text{meas}_0^n q \text{ in }
\text{let } \langle b_{2}^{\text{bit}}, q_2^{(n-2)\text{-qbit}} \rangle = \text{meas}_1^{n-1} q_1 \text{ in } \cdots
\text{let } \langle b_{n-1}^{\text{bit}}, q_{n-1}^{\text{qbit}} \rangle = \text{meas}_1^n q_{n-2} \text{ in }
\langle b_{1}, \langle b_{2}, \cdots , \langle b_{n-1}, \text{meas}_1^n q_{n-1} \cdots \rangle \rangle : n\text{-qbit} \rightarrow \text{bit}^n.

The program \textsf{Grover}(u, q, m) applies \(u^m\) to \(q\), and then measures the result. Therefore, we can run Grover’s algorithm by the following code:

\textsf{Grover}\left(\lambda q^n\text{-qbit. } U_1 (U_2 q) \right) \left(U ((\text{new } \texttt{tt}) \bowtie \cdots \bowtie (\text{new } \texttt{tt})) M_0 \right)

where \(U : n\text{-qbit} \rightarrow n\text{-qbit}\) is a unitary transformation that maps \(|0\cdots 0\rangle\) to \(|s\rangle\), and \(M_0\) is a sufficiently large number so that we can observe the answer with a desired degree of precision. We suppressed occurrences of \texttt{cmp}.

4. The Quantum Branching Monad \(\mathcal{Q}\) and The Category \(\text{PER}_{\mathcal{Q}}\)

We now turn to denotational semantics of Hoq.

4.1. Background

The starting point of our current work was Jacobs’ observation [33] that relates: monads for branching (§2.2, used in coalgebraic trace semantics) and traced monoidal categories that appear in categorical GoI (§2.3). See also Rem. 2.15. This relationship establishes the first among the three steps in Fig. 1.

Examples of a traced monoidal category \(\mathcal{C}\) used in categorical GoI [21] are divided into two groups: the so-called wave-style ones where \(\mathcal{C}\)’s monoidal structure is given by products \(\times\); and the particle-style ones where it is given by coproducts \(+\). The former are of static nature and includes domain-theoretic examples like \(\omega\text{-CPO}\). The latter particle-style examples are often dynamic, in the sense that we can imagine a “particle” (or a “token”) moving around (we will further elaborate this point later). This is the class of examples we are more interested in. The examples include:

• the category \(\text{Pfn}\) of sets and partial functions;

• the category \(\text{Rel}_+\) of sets and binary relations, where the subscript \(+\) indicates that the relevant monoidal structure is the one given by disjoint unions of sets; and

• the category \(\text{SRel}\) of measurable spaces and stochastic relations.
For us the crucial observation is that these examples are (close to) the Kleisli categories $K(\mathcal{B})$ for the “branching” monads $\mathcal{B}$ in Example 2.14, §2.2. Indeed, it is easy to see that $\text{Pfn}$ and $\text{Rel}$ are precisely $K(\mathcal{L})$ and $K(\mathcal{P})$, respectively; the category $K(\mathcal{D})$ can be thought of as a discrete variant of $\text{SRel}$.

Generalizing this observation, Jacobs [33] proves that a monad $\mathcal{B}$ for branching—i.e., a monad on $\text{Sets}$ with order enrichment, subject to some additional conditions—has its Kleisli category $K(\mathcal{B})$ traced monoidal (see Thm. 4.5 later). Here the monoidal structure is given by $+$, coproduct in $\text{Sets}$ (and also in $K(\mathcal{B})$, since the Kleisli embedding $\text{Sets} \to K(\mathcal{B})$ preserves coproducts).

Let us elaborate on such a Kleisli category $K(\mathcal{B})$. We look at it as a category of piping. An arrow $f : X \to Y$ in $K(\mathcal{B})$ is understood as a bunch of pipes, with $|X|$-many entrances and $|Y|$-many exits. The pipes are where a particle (or token) runs through. See below; here the shaded box $f$ consists of a lot of pipes.

According to the choice of a monad $\mathcal{B}$ (see Example 2.14), different “branching” of such pipes is allowed.

- When $\mathcal{B} = \mathcal{L}$ a pipe can be “stuck” or “looped.” A pipe connects an entrance $x$ with the exit $f(x)$—hence a token entering at $x$ comes out of $f(x) \in Y$—when $f(x)$ is defined. A token is caught in the piping and does not come out in case $f(x)$ is undefined, i.e., if $f(x)$ belongs to $1$ in $\mathcal{L} = {1 + (-)}$.
- When $\mathcal{B} = \mathcal{P}$ a pipe can branch, with one entrance $x$ connected to possibly multiple (or zero) exits (namely those in $f(x) \subseteq Y$).
- When $\mathcal{B} = \mathcal{D}$ a pipe can branch too, but this time the branching is probabilistic.

For all these monads $\mathcal{B}$ it is shown in [33] that the Kleisli category $K(\mathcal{B})$ is symmetric traced monoidal, with respect to $+$ as a monoidal product and $0$ (the empty set) as a monoidal unit. In view of Fig. 1, all these Kleisli categories can support construction of a linear category via categorical GoI and realizability.

---

9 We shall use $\rightarrow$ to denote an arrow in a Kleisli category.

10 Our piping analogy is not completely faithful: in a Kleisli arrow $f$ the two crossings $\times$ and $\bigtimes$ are identified, but they are different as physical pipes.
Moreover, it is plausible that the resulting linear category inherit some features from its ingredients—ultimately from the branching monad $B$. For example, we start with $B = \mathcal{P}$ and the outcome would be a linear category with some nondeterminism built-in, hence suited for interpreting a language with nondeterminism.

Therefore for our purpose of obtaining a linear category with a quantum flavor—and interpreting a quantum lambda calculus in it—the first question is to find a branching monad with a quantum flavor. Our answer is the quantum branching monad $Q$ that we introduce now.

4.2. The Quantum Branching Monad $Q$

The following formal definition of $Q$ below is hardly illustrative. The intuition will be explained shortly, using the piping analogy (15) for arrows in the Kleisli category $\mathcal{K}(Q)$.

**Definition 4.1 (The quantum branching monad $Q$).** The quantum branching monad $Q : \text{Sets} \to \text{Sets}$ is defined as follows. On objects,

$$QX = \left\{ c : X \to \prod_{m,n \in \mathbb{N}} \text{QO}_{m,n} \mid \text{the trace condition (16)} \right\}$$

where the *trace condition* is:

$$\sum_{x \in X} \sum_{n \in \mathbb{N}} \text{tr}\left( (c(x))_{m,n}(\rho) \right) \leq 1, \quad \forall m \in \mathbb{N}, \forall \rho \in \text{DM}_m. \quad (16)$$

Here $(c(x))_{m,n}$ is the $(m,n)$-component of $c(x) \in \prod_{m,n} \text{QO}_{m,n}$. On arrows, given $f : X \to Y$ we define $Qf : QX \to QY$ as follows. For $c \in QX$ and $y \in Y$:

$$\left((Qf)(c)(y)\right)_{m,n} := \sum_{x \in f^{-1}(y)} (c(x))_{m,n}. \quad (17)$$

As for the monad structure, its unit $\eta_X : X \to QX$ is:

$$\left(\eta_X(x)(x')\right)_{m,n} := \begin{cases} \mathcal{I}_m & \text{if } x = x' \text{ and } m = n, \\ 0 & \text{otherwise}. \end{cases} \quad (18)$$

Here $\mathcal{I}_m$ is the identity map; 0 is the constant QO that maps everything to 0. The multiplication $\mu_X : QQX \to QX$ is defined by:

$$\left(\mu_X(\gamma)(x)\right)_{m,n} := \sum_{c \in QX} \sum_{k \in \mathbb{N}} \left((c(x))_{k,n} \circ (\gamma(c))_{m,k}\right). \quad (19)$$

The QO $(c(x))_{k,n} \circ (\gamma(c))_{m,k}$ on the RHS is the sequential composition of QOs: given a density matrix $\rho \in \text{DM}_m$ it first applies $(\gamma(c))_{m,k} \in \text{QO}_{m,k}$ and then applies $(c(x))_{k,n} \in \text{QO}_{k,n}$, transforming $\rho$ eventually into an $n$-dimensional density matrix.
In Appendix AppendixC we prove that the sums in (17) and (19) exist, that \( \mathcal{Q} \) is indeed a functor, and that \( \mathcal{Q} \) is indeed a monad.

Let us first note a common pattern that is exhibited by \( \mathcal{Q} \) and the previous examples of branching monads \( B \), namely:

\[
BX = \{ c : X \to W \mid \text{a normalizing condition} \} , \quad \text{where } W \text{ is a set of weights.}
\]

Specifically:

- For \( B = \mathcal{L} \) the set \( W \) is \( 2 = \{0, 1\} \) (stuck or through); the normalizing condition is
  \( c(x) = 1 \) for at most one \( x \in X \).
- For \( B = \mathcal{P} \) the set \( W \) is \( 2 = \{0, 1\} \) again, but there is no normalizing condition.
- For \( B = \mathcal{D} \) the set \( W \) is the unit interval \([0, 1]\) and the normalizing condition is \( \sum_{x \in X} c(x) \leq 1 \).
- For \( B = \mathcal{Q} \) the weights are a tuple of quantum operations and the normalizing condition is the trace condition (16).

Let us continue (15) and think of an arrow \( f : X \to Y \) in \( K\ell(\mathcal{Q}) \) as piping. The piping analogy is still valid for \( \mathcal{Q} \); a crucial difference however is that, for \( B = \mathcal{Q} \),

\[
\text{a token that runs through pipes is no longer a mere particle, but it carries a quantum state.}
\]

Each entrance \( x \in X \) is ready for an incoming token that carries \( \rho \in \text{DM}_m \) of any finite dimension \( m \). Such a token gives rise to one outcoming token. However, its exit can be any \( y \in Y \) and the quantum state carried by the token can be of any finite dimension \( n \in N \). We think of the piping to be applying a certain QO to the quantum state carried by the token; the QO to be applied is concretely given by

\[
( f(x)(y) )_{m,n} \in \text{QO}_{m,n} ,
\]
that is determined by: which exit $y \in Y$ the token takes; and what is the dimension $n$ of the resulting quantum state.

The trace condition (16) now reads, for an arrow $f : X \rightarrow Y$ in $K\ell(Q)$:

$$\sum_{y \in Y} \sum_{n \in \mathbb{N}} \text{tr} \left( (f(x)(y))_{m,n}(\rho) \right) \leq 1 , \text{ for each } m \in \mathbb{N}, \rho \in DM_m \text{ and } x \in X. \quad (21)$$

The trace value $\text{tr} \left( (f(x)(y))_{m,n}(\rho) \right)$ is understood as the probability with which a token $\rho$ entering at $x$ leads to an $n$-dimensional token at $y$. These probabilities must add up to at most 1 when the exit $y$ and the outcoming dimension $n$ vary. This is precisely the condition (21).

The composition $\circ$ of Kleisli arrows can then be understood as sequential connection of such piping, one after another.

Here the numbers $m, k$ and $n$ stand for the dimension of the quantum states carried by the token, at each stage of the piping.

Concretely, the Kleisli composition $\circ$ is described as follows.
Lemma 4.2 (Composition $\circ$ in $K\ell(Q)$). Given two successive arrows $f : X \to Y$ and $g : Y \to U$ in $K\ell(Q)$, their composition $g \circ f : X \to U$ is concretely given as follows.

$$(g \circ f)(x)(u)_{m,n} = \sum_{y \in Y} \sum_{k \in \mathbb{N}} (g(y)(u))_{k,n} \circ (f(x)(y))_{m,k}. $$

Proof. See Appendix AppendixC.1.

This description of $\circ$ in $K\ell(Q)$ is ultimately due to the definition (19) of the multiplication operation $\mu$. We notice its similarity to the multiplication operation of the subdistribution monad $\mathcal{D}$. The latter is defined by

$$\mu_X^\mathcal{D}(\gamma)(x) = \sum_{c \in \mathcal{D}X} \gamma(c) \cdot c(x),$$

where $\cdot$ denotes multiplication of real numbers. This notably resembles (19).

Remark 4.3. The reason why $Q$ is called a quantum branching monad is that a Kleisli arrow $f : X \to Y$ in $K\ell(Q)$—thought of as piping like (20)—is a “quantum branching function.” This is in the same sense as an arrow $f : X \to Y$ in $K\ell(D)$ is a “nondeterministically branching function” and an arrow $f : X \to Y$ in $K\ell(P)$ is a “probabilistically branching function.”

An example of such an arrow in $K\ell(Q)$ is given by the following $f_1$. Here $k, l, m$ and $n$ are natural numbers, and $\rho \in \mathbb{D}M_m$ is an $m$-dimensional density matrix.

$$f_1 : \mathbb{N} \to \mathbb{N},$$

$$(f_1(k)(l))_{m,n}(\rho) := \begin{cases} (0|\rho|0) & \text{if } m = 2, n = 1 \text{ and } l = 2k; \\ (1|\rho|1) & \text{if } m = 2, n = 1 \text{ and } l = 2k + 1; \\ 0 & \text{otherwise}. \end{cases}$$

Imagine a token carrying a quantum state $\rho \in \mathbb{D}M_m$ entering this piping at the entrance $k \in \mathbb{N}$. The token does not come out at all unless $\rho$ is 2-dimensional. If $\rho$ is 2-dimensional, the token might come out of the exit $2k \in \mathbb{N}$ or $2k + 1 \in \mathbb{N}$. To each of these exits the assigned value is $\langle 0|\rho|0 \rangle$ and $\langle 1|\rho|1 \rangle$, respectively: these numbers in $\mathbb{C}^1$ (or rather $[0,1]$) are understood as the probabilities with which the token takes the exit.

This way we are modeling branching structure that depends on quantum data—or classical control and quantum data. The principle is:

- a classical control structure is represented by the pipe the token is in; and
- quantum data is the one carried by the token.

Notice also that we are essentially relying on the separation of a measurement into projections (see Rem 2.11).
A slightly more complicated example is the following $f_2$. Here $N \in \mathbb{N}$ is a natural number.

$$f_2 : \mathbb{N} \rightarrow \mathbb{N},$$

$$(f_2(k)(l))_{m,n}(\rho) := \begin{cases} 
(0_1|\rho|0_1) & \text{if } m = 2^{N+1}, n = 2^N \text{ and } l = 2k, \\
(1_1|\rho|1_1) & \text{if } m = 2^{N+1}, n = 2^N \text{ and } l = 2k + 1, \\
0 & \text{otherwise.}
\end{cases}$$

Here an incoming token carries an $(N+1)$-qubit state $\rho \in \text{DM}_{2^{N+1}}$, and the arrow $f_2$ measures its first qubit (with respect to the basis consisting of $|0_1\rangle$ and $|1_1\rangle$), resulting in the token sent to different exists according to the outcome.

The outgoing token carries a state

$$(0_1|\rho|0_1) \text{ or } (1_1|\rho|1_1) \in \text{DM}_{2^N}$$

that represents the qubits from the second to the $(N+1)$-th. Here the trace value of each of the two density matrices implicitly represents the probability with which the token is sent to the corresponding exit. See Rem. 2.11.

It is not only measurements that we can model using arrows in $K\ell(Q)$. Consider the following $f_3$. Here $K \in \mathbb{N}$ is a natural number.

$$f_3 : \mathbb{N} \rightarrow \mathbb{N},$$

$$(f_3(k)(l))_{m,n}(\rho) := \begin{cases} 
\rho \otimes |0\rangle\langle 0| & \text{if } m = 2^N, n = 2^{N+1} \text{ and } k = l = 2K, \\
\rho & \text{if } m = n = 2^N \text{ and } k = l = 2K + 1, \\
0 & \text{otherwise.}
\end{cases}$$

This arrow models (conditional) state preparation: it adds, to a token coming in at $k = 2K$, a prepared state $|0\rangle\langle 0|$ as an $(N+1)$-th qubit.

Furthermore, the composition $f_3 \circ f_2 : \mathbb{N} \rightarrow \mathbb{N}$ represents the following operation: it measures the first qubit; and if the outcome is $|0\rangle$, it adjoins a new qubit.

The monad $Q$ indeed satisfies the conditions in [33]—much like $\mathcal{L}, \mathcal{P}$ and $\mathcal{D}$, it is equipped with a suitable cpo structure—so that the Kleisli category $K\ell(Q)$ is a traced symmetric monoidal category (TSMC).

**Definition 4.4 (Order on $QX$).** We endow the set $QX$ with the pointwise extension of the order in Def. 2.12. Namely: given $c, d \in QX$, we have $c \sqsubseteq d$ if for each $x \in X, m, n \in \mathbb{N}$, $(d(x))_{m,n} \sqsubseteq (c(x))_{m,n}$.

**Theorem 4.5.** The category $K\ell(Q)$ is partially additive (a notion from [51]). Therefore by [52, Chap. 3], $(K\ell(Q), +, 0)$ is a TSMC, with its trace operator given explicitly by Girard’s execution formula.

**Proof.** By [33, Prop. 4.8]; see Thm. AppendixC.5 for details. Notably, the Kleisli category $K\ell(Q)$ is $\omega$-CP0 enriched: a homset $K\ell(Q)(X,Y)$ is equipped with the order that is the pointwise extension of that on $QY$ (Def. 4.4).
4.3. A Linear Combinatory Algebra via Categorical GoI

A TSMC is a main component of the notion of GoI situation in [21] (see Fig. 1 and §2.3). In categorical GoI [21] (whose idea goes back to [53, 54]), one derives from a GoI situation a linear combinatory algebra (LCA) that is a “model of an untyped linear calculus.” For that purpose, what are needed on top of a TSMC are: a functor \(F\) for interpreting the ! modality; and a reflexive object \(U\) that would yield the carrier of an LCA. These data can be provided to \(K\ell(Q)\) in the same way as to \(Pfn \cong K\ell(L)\) and \(Rel_+ \cong K\ell(P)\), as is done in [21].

**Definition 4.6 (Retraction).** Let \(X\) and \(Y\) be objects of a category \(C\). A retraction from \(X\) to \(Y\) is a pair of arrows \(f : X \rightarrow Y\) and \(g : Y \rightarrow X\) such that \(g \circ f = \text{id}_X\), that is,

\[
\begin{array}{ccc}
\text{id} & X & \xrightarrow{f} Y \\
& \downarrow{g} & \ \ \\
& Y & \xrightarrow{g} X
\end{array}
\]

Such a retraction shall be denoted by \(f : X \triangleleft Y : g\), following [21].

**Definition 4.7 (GoI situation).** A GoI situation is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \rightarrow C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF \triangleleft F : e'\) Comultiplication
  - \(d : \text{id} \triangleleft F : d'\) Dereliction
  - \(c : F \otimes F \triangleleft F : c'\) Contraction
  - \(w : K_I \triangleleft F : w'\) Weakening

Here \(K_I\) is the constant functor into the monoidal unit \(I\);

- \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
  - \(j : U \otimes U \triangleleft U : k\)
  - \(I \triangleleft U\)
  - \(u : FU \triangleleft U : v\)

**Theorem 4.8.** The triple \((K\ell(Q), N \cdot _, N)\) forms a GoI situation. Here the functor \(N \cdot _- : K\ell(Q) \rightarrow K\ell(Q)\) carries \(X\) to the copower \(N \cdot X\).

**Proof.** The only nontrivial part is to show that \(N \cdot _-\) preserves traces. Since the trace operator in \(K\ell(Q)\) can be described using Girard’s execution formula (due to the results in [33, 52]), we can use a lemma that is similar to [21, Lem. 5.1].

\[\square\]
We describe the outcome of the categorical GoI construction.

**Definition 4.9 (Linear combinatory algebra, LCA).** A linear combinatory algebra (LCA) is a set $A$ equipped with

- a binary operator (called an applicative structure) $\cdot : A^2 \to A$;
- a unary operator $! : A \to A$; and
- distinguished elements (called combinators) $B, C, I, K, W, D, \delta, \text{ and } F$, of $A$.

These are required to satisfy the following equalities.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Bxyz = x(yz)$</td>
<td>Composition, Cut</td>
<td></td>
</tr>
<tr>
<td>$Cxyz = (xz)y$</td>
<td>Exchange</td>
<td></td>
</tr>
<tr>
<td>$lr = x$</td>
<td>Identity</td>
<td></td>
</tr>
<tr>
<td>$Kxy = x$</td>
<td>Weakening</td>
<td></td>
</tr>
<tr>
<td>$Wxy = x!y!y$</td>
<td>Contraction</td>
<td></td>
</tr>
<tr>
<td>$D!x = x$</td>
<td>Dereliction</td>
<td></td>
</tr>
<tr>
<td>$\delta!x = !!x$</td>
<td>Comultiplication</td>
<td></td>
</tr>
<tr>
<td>$F!x!y = !(xy)$</td>
<td>Monoidal functoriality</td>
<td></td>
</tr>
</tbody>
</table>

The notational convention is: $\cdot$ associates to the left; $\cdot$ is suppressed; and $!$ binds stronger than $\cdot$ does.

We apply [21, Prop. 4.2] to the GoI situation of Thm. 4.8 and obtain an LCA.

**Theorem 4.10 (The quantum LCA $A_Q$).** The homset

$$A_Q := \mathcal{K}(\mathcal{Q})(N, N)$$

is a linear combinatory algebra (LCA).

We apply [21, Prop. 4.2] to the GoI situation of Thm. 4.8 and obtain an LCA.

$A_Q$ is a linear combinatory algebra (LCA).

We describe the LCA $A_Q$ in some detail. Its application operator $\cdot$ and the $!$ operator are concretely as follows. Given $a, b \in A_Q$,

$$a \cdot b := tr_{N,N}^{N,N} \left( N + N \to N \to N \to N \to N + N \to N + N \to N \to N \to N \to N \right)$$

$$!a := \left( N \to N \cdot N \to N \cdot N \to N \to N \to N \right)$$

(23)
Here $j : \mathbb{N} + \mathbb{N} \cong \mathbb{N} : k$ and $u : \mathbb{N} \cdot \mathbb{N} \cong \mathbb{N} : v$ are isomorphisms in $\textbf{Sets}$ (which we fix henceforth) that are embedded in $\mathcal{K}\ell(\mathcal{Q})$. This is like in [21, §5.1] with $\text{Pfn}$ and $\text{Rel}_+$. The above two diagrams in (23) are string diagrams in the TSMC $\mathcal{K}\ell(\mathcal{Q})$: a plain string represents $\mathbb{N}$ and a double wire represents $\mathbb{N} \cdot \mathbb{N}$; the dashed box in the second diagram is like a functorial box (see e.g. [55, 46]) and represents application of the functor $\mathbb{N} \cdot -$. Note that these string diagrams are different from the previous pipe diagrams like (15). The two string diagrams above, drawn as pipe diagrams, are as follows. In the first diagram we are using the decomposition of $\mathbb{N}$ into even and odd numbers as the fixed isomorphism $\mathbb{N} \cong \mathbb{N} + \mathbb{N}$.

For $\mathcal{A}_\mathcal{Q}$ from Thm. 4.10, the LCA combinators $B, C, I, K, W, D, \delta$ and $F$ are defined using the general definitions in [21, §4] that work for any GoI situation. For example, the $B$ combinator is given by the following element of $\mathcal{A}_\mathcal{Q} = \mathcal{K}\ell(\mathcal{Q})(\mathbb{N}, \mathbb{N})$.

This diagram is a string diagram in $\mathcal{K}\ell(\mathcal{Q})$: the triangles denote the isomorphisms $j : \mathbb{N} + \mathbb{N} \cong \mathbb{N} : k$. By expanding the application operation $\cdot$ according to (23), it is not hard to see that the equation $Bxyz = x(yz)$ holds. See Fig. 2; an important point there is that two triangles pointing to each other cancel out, that is,

$$
\begin{array}{c}
\quad \\
\quad = \\
\quad , \\
\quad \quad \quad \quad \quad \text{since } k \circ j = \text{id}.
\end{array}
$$

The following observation follows from [32, §2.2].

**Proposition 4.11.** The LCA $\mathcal{A}_\mathcal{Q}$ in Thm. 4.10 is affine: it has the full K combinator such that $Kxy = x$. □
Figure 2: Proof of $B_{xyz} = x(yz)$
4.4. A Linear Category via Realizability

We employ the (linear) realizability technique [32, 35] and turn an LCA (an untyped model) into a linear category (a typed model). See §2.4.

Definition 4.12 (PER). A partial equivalence relation (PER) over \( A_Q \) is a symmetric and transitive relation \( X \) on the set \( A_Q \). The domain of a PER \( |X| \) is defined by

\[
|X| := \{ x \mid (x, x) \in X \} = \{ x \mid \exists y. (x, y) \in X \}.
\]

When restricted to its domain \( X \) is an equivalence relation; therefore \( X \) can be thought of as a subset \( |X| \subseteq A_Q \) quotiented.

PERs over \( A_Q \) form a category \( \text{PER}_Q \). Its arrow \( X \to Y \) is defined to be an equivalence class of the PER

\[
X \rightarrow Y := \{ (c, c') \mid (x, x') \in X \Rightarrow (cx, c'x') \in Y \}.
\]

We denote by \([c] \) the equivalence class in \( X \rightarrow Y \) to which \( c \in A_Q \) belongs. That is, \([c] \) is an arrow that is “realized by the code \( c \).”

We elaborate further on the definition (24). Its domain \( |X \rightarrow Y| \) is the set of \( c \in A_Q \) such that \( (x, x') \in X \) implies \( (cx, c'x') \in Y \); this requirement is that the function \([c] : |X|/X \to |Y|/Y \) is well-defined. The PER \( X \rightarrow Y \) identifies \( c \) and \( c' \) such that \( (cx, c'x') \in Y \) for each \( |X| \). This is the extensionality of the functions of the type \( |X|/X \to |Y|/Y \).

Theorem 4.13. The category \( \text{PER}_Q \) is a linear category [30, 31], equipped with a symmetric monoidal structure \((I, \otimes)\) and a so-called linear exponential comonad \(! \). The latter means that \(! \) is a symmetric monoidal comonad, with natural transformations

\[
\begin{align*}
der : & \ !X \to X, & \delta : & \ !X \to !!X, \\
\varphi : & \ !X \otimes !Y \to !(X \otimes Y), & \varphi' : & \ I \to !I,
\end{align*}
\]

that is further equipped with monoidal natural transformations

\[
\begin{align*}
\text{weak} : & \ !X \to I \text{ and } \text{con} : \ !X \to !X \otimes !X,
\end{align*}
\]

subject to certain additional conditions.

Proof. By [32, Thm. 2.1]. See also Lem. 5.2 later.

We use the symbol \( \otimes \) for the monoidal product in \( \text{PER}_Q \); it is distinguished from the tensor product of quantum states denoted by \( \otimes \). See §3.3 for the discussion on this issue.
5. Interpretation of Hoq

We now present our interpretation of Hoq in the category \( \mathbf{PER}_Q \). We have seen in Thm. 4.13 that the category is a linear category, hence models (standard) linear \( \lambda \)-calculi. See e.g. [46].

However, the specific calculus Hoq calls for some extra features. Firstly, the linear exponential comonad \( ! \) on \( \mathbf{PER}_Q \) should be idempotent (\( ! ! X \cong ! X \)). This is because in Hoq we chose to implicitly track linearity by subtyping \( \langle \cdot \rangle \) in contrast to explicit tracking by constructs like \textit{derelict} in standard linear \( \lambda \)-calculi. See §3.3. This issue is addressed in §5.1. Secondly we need some quantum mechanical constructs in \( \mathbf{PER}_Q \) for interpreting constants of Hoq; see §5.2. Finally, we need a strong monad \( T \) on \( \mathbf{PER}_Q \) for the probabilistic effect that arises inevitably in quantum computation (more specifically through measurements). In fact we will use for \( T \) a continuation monad \( (\_ \rightarrow R) \rightarrow R \) with the result type \( R \) described as a final coalgebra; see §5.3.

5.1. Type Constructors in \( \mathbf{PER}_Q \)

Here we first present a concrete description of type constructors (such as \( \boxtimes \) and \( ! \)) in \( \mathbf{PER}_Q \) as a linear category (Thm. 4.13). In its course we follow [32]. After that we describe some specific features of \( \mathbf{PER}_Q \)—in fact they are present in \( \mathbf{PER}_A \) for any affine LCA \( A \)—such as the isomorphisms \( ! ! X \cong ! X \) and \( !(X + Y) \cong ! X + ! Y \). These extra isomorphisms allow us to model implicit linearity tracking.

In what follows, an element of the LCA \( A_Q \) is often designated by an untyped linear \( \lambda \)-term. This is justified by combinatory completeness of LCAs; see e.g. [32, 56].

**Definition 5.1.** We introduce the following combinators in \( A_Q \).

\[
\begin{align*}
P &:= \lambda xyz.zxy \quad \text{Pairing} \\
K &:= \text{KI} \quad \text{Weakening} \quad Kxy = y \\
P_l &:= \lambda w.w \quad \text{Left Projection} \quad P_l(Pxy) = x \\
P_r &:= \lambda w.w \quad \text{Right Projection} \quad P_r(Pxy) = y
\end{align*}
\]

Recall that the full \( K \) combinator is available in the affine LCA \( A_Q \) (Prop. 4.11). Obviously the pairing in \( \mathbf{PER}_Q \) is extensional: \( Pxy = Px'y' \) implies \( x = x' \) and \( y = y' \).

**Lemma 5.2 (Type constructors in \( \mathbf{PER}_Q \)).** The category \( \mathbf{PER}_Q \) is a symmetric monoidal closed category with respect to the following operations.

\[
\begin{align*}
X \boxtimes Y &:= \{ (Pxy, Px'y') \mid (x, x') \in X \land (y, y') \in Y \} \\
I &:= \{(I, I)\} \\
X \rightarrow Y &:= (\text{see (2)}} \}
\end{align*}
\]

The operations’ action on arrows is defined in a straightforward manner. For example, given \( [c_1] : X_1 \rightarrow Y_1 \) and \( [c_2] : X_2 \rightarrow Y_2 \) in \( \mathbf{PER}_Q \),

\[
[c_1] \boxtimes [c_2] := [\lambda w.w(\lambda uv.P(c_1u)(c_2v))] : X_1 \boxtimes X_2 \rightarrow Y_1 \boxtimes Y_2 .
\]

40
Moreover, the monoidal unit $1$ is final (i.e. terminal) in $\text{PER}_Q$, with a unique arrow $\text{weak} : X \to I$ given by\footnote{We shall use the same notation, $\text{weak}$, for both the unique arrow $X \to I$ and a structure morphism for a linear exponential comonad $!X \to I$ (Thm. 4.13). They are indeed the same arrow $[KI]$.}

$$\text{weak} := [KI].$$

The category has a linear exponential comonad $!$ (see [32]):

$$! X := \{(!!x, !!x') \mid (x, x') \in X\}, \quad ![c] := [F[c]],$$

(26)

where the $!$'s on the right-hand sides are an operation on the LCA $A_Q$ (Thm. 4.10).

The natural transformations $\text{der}$, $\delta$, $\varphi$, weak and con that accompany a linear exponential comonad (see (25–4.13)) are concretely given as follows.

$$\begin{align*}
\text{der} & := [D] : !X \to X \\
\delta & := [\delta] : !X \to !!X \\
\varphi & := [\lambda w.w(\lambda uv.(F[(P)u])v)] : !X \times !!Y \to !(X \boxtimes Y) \\
\varphi' & := [\lambda w.w(1!)] : I \to !!I \\
\text{weak} & := [KI] : X \to I \\
\text{con} & := [WP] : !X \to !!X \\
\end{align*}$$

(27)

Recall $[c]$ denotes an arrow in $\text{PER}_Q$ that is realized by the code $c \in A_Q$.

The category also has binary products and coproducts. Products are realized by a CPS-like encoding.

$$\begin{align*}
X \times Y & := \{(PK_1(Pk_2u), PK_1'(Pk_1'k_2'uv)) \mid (k_1u, k_1'k_2'uv) \in X \land (k_2u, k_2'uv) \in Y\}, \\
X + Y & := \{(PKx, PKx') \mid (x, x') \in X\} \cup \{(PKy, PKy') \mid (y, y') \in Y\}.
\end{align*}$$

For example, the projection maps are concretely as follows.

$$\begin{align*}
\pi_1 & := \left[\lambda w.w(\lambda kv.v(\lambda lu.ku))\right] : X \times Y \to X; \\
\pi_2 & := \left[\lambda w.w(\lambda kv.v(\lambda lu.lu))\right] : X \times Y \to Y.
\end{align*}$$

(28)

Proof. Straightforward; see e.g. [32, 35].

Logically $\boxtimes$ is “multiplicative and”; $\times$ is “additive and.”

Lemma 5.3. In $\text{PER}_Q$ we have the following canonical isomorphisms.

$$\begin{align*}
!(X + Y) & \xrightarrow{\sim} !!X + !Y \\
\text{der}_X : !!X & \xrightarrow{\sim} !!X : \delta \\
\text{der}_X : !!X & \xrightarrow{\sim} !!X : \delta \\
!(X \boxtimes Y) & \xrightarrow{\sim} !X \boxtimes !!Y : \varphi \\
\text{weak} : !!I & \xrightarrow{\sim} !I : \varphi'
\end{align*}$$

(29)
Here the arrows $\text{der, } \delta, \varphi, \varphi'$ are from Thm. 4.13. Therefore $!$ on $\text{PER}_Q$ is idempotent and strong monoidal; it also preserves coproducts. Additionally, as in any linear category:

$$
! (X \times Y) \cong X \boxtimes ! Y,
$$

$$
(X + Y) \boxtimes Z \cong X \boxtimes Z + Y \boxtimes Z,
$$

$$
I \rightarrow X \cong X,
$$

$$
(X + Y) \rightarrow Z \cong (X \rightarrow Z) \times (Y \rightarrow Z).
$$

(30)

Proof. The ones in (30) are standard; see [57, §2.1.2]. The ones in (29) also hold in any $\text{PER}_A$ with an affine LCA $A$. For the first ($!$ distributes over $+$), from right to left one takes $[! \kappa, ! \kappa]$ where $\kappa, \kappa'$ are coprojections; from left to right one can take

$$
[a] : !(X + Y) \rightarrow X + ! Y \quad \text{with} \quad a := \lambda w. W P w (\lambda u v. P (D w K) (F (! P) v)) ;
$$

indeed it is straightforward to see that $a([(PK x)]) = PK(! x)$ and $a([PK x]) = PK(! x)$ for the above $a$. The second line of (29) ($!$ is idempotent) follows immediately from the definitions of $\text{der}$ and $\delta$ in (27). For example,

$$
(! \text{der}_X)(!)x = (!! D)(!! x) = F (! D)(!! x) \quad \text{by (26), def. of } !'s \text{ action on arrows}
$$

$$
= ![D ! x] = ! x .
$$

For the third line ($!$ distributes over $\boxtimes$), an inverse of $\varphi$ can be given by the following composite, exploiting the finality of $I$ (Lem. 5.2).

$$
!(X \boxtimes Y) \rightarrow !(X \boxtimes Y) \boxtimes !(X \boxtimes Y) \rightarrow !(X \boxtimes I) \boxtimes !(I \boxtimes Y) \rightarrow ! X \boxtimes ! Y
$$

This concludes the proof.

We introduce a different pairing combinator $\hat{P}$ that leads to a different “implementation” $\hat{\times}$ of products. The merit of $\hat{\times}$ is that it exhibits better order-theoretic properties; we will need them for recursion. In contrast, $P$ enjoys a useful combinatorial property: $(P x y) z = z x y$.

Definition 5.4 (Combinator $\hat{P}$, binary product $X \hat{\times} Y$). We define an element $\hat{P} \in A_Q$ by the string diagram in $K(Q)$ shown below on the left. The triangles denote $j : N + N \cong N : k$ in Thm. 4.10. Then $\hat{P} x y$ becomes as shown bottom on the right.

$$
\hat{P} := \\
\hat{P} x y =
$$

(31)
Let $\dot{P}_l$ and $\dot{P}_r$ be the following elements of $A_Q$.

$\dot{P}_l := \begin{array}{c}
\text{Diagram 1}
\end{array}$

$\dot{P}_r := \begin{array}{c}
\text{Diagram 2}
\end{array}$

Here the nodes 1 and 1 denote the unique arrows $\mathbb{N} \rightarrow 0$ and $0 \rightarrow \mathbb{N}$ in $\mathcal{K}l(Q)$, respectively. Furthermore, let us introduce the following conversion combinators.

$C_{\dot{P} \rightarrow P} := \lambda w. w \dot{P}$

$C_{\dot{P} \rightarrow P} := \begin{array}{c}
\text{Diagram 3}
\end{array}$

We define $X \times Y$ by replacing $P$ with $\dot{P}$ in $X \times Y$ (Lem. 5.2).

**Lemma 5.5.** We have, for each $x, y \in A_Q$,

$\dot{P}_l(\dot{P}xy) = x$, \hspace{1em} $\dot{P}_r(\dot{P}xy) = y$ \hspace{1em} $C_{\dot{P} \rightarrow P}(Pxy) = \dot{P}xy$ \hspace{1em} $C_{\dot{P} \rightarrow P}(\dot{P}xy) = Pxy$.

The latter two result in a canonical natural isomorphism $X \times Y \cong X \times Y$ in $\text{PER}_Q$. Therefore in what follows we shall use $\times$ and $\times$ interchangeably. That is, we suppress use of the conversion combinators $C_{\dot{P} \rightarrow P}$ and $C_{P \rightarrow \dot{P}}$.

**Proof.** For the first equality, the proof goes as follows.

$\dot{P}_l(\dot{P}xy) = \begin{array}{c}
\text{Diagram 4}
\end{array}$

where $(\ast)$ and $(\dagger)$ hold because of the dinaturality (also called sliding) and yanking axioms of traced monoidal categories, respectively (see [43, 21]); and
(‡) holds by a direct calculation in $\mathcal{K}(Q)$. The second equality $\hat{P}_r(\hat{P}_{xy}) = y$ is similar.

The third equality is easy exploiting the combinatorial property $(P_{xy})z = zxy$ of $P$:

$$C_{\hat{P}_r}(P_{xy}) = (P_{xy})\hat{P} = \hat{P}_{xy}.$$ The last equality is shown as follows, where $(\ast)$ holds due to the dinaturality axiom.

$$C_{\hat{P}_r}(\hat{P}_{xy}) = \begin{array}{c}
\vdots \\
\vdots
\end{array} = P_{xy} \quad \square$$

We go ahead and show that the category $\text{PER}_Q$ has countable limits and colimits.

**Definition 5.6 (Combinators $(x_i)_{i \in \mathbb{N}}, D_i$).** Let $x_0,x_1,\ldots \in A_Q$. We define the element $(x_i)_{i \in \mathbb{N}} \in A_Q$ by:

$$(x_i)_{i \in \mathbb{N}} := \left( \begin{array}{c}
N \xrightarrow{u} N \cdot N \xrightarrow{\bigoplus x_i} N \cdot N \xrightarrow{\bar{N}} N \\
\end{array} \right) = \begin{array}{c}
\vdots \\
\vdots
\end{array}$$

where $u : N \cdot N \cong N \cdot u$ are (fixed) isomorphisms in Thm. 4.10.

For each $i \in \mathbb{N}$, we define element $D_i \in A_Q$ by

$$D_i := \left( \begin{array}{c}
N \xrightarrow{k} N + N \xrightarrow{\kappa_i \cdot \bar{N}} N \cdot N \xrightarrow{\bar{N} + N} N \cdot N \xrightarrow{\bar{N}} N \\
\end{array} \right) = \begin{array}{c}
\vdots \\
\vdots
\end{array}$$

Here $\kappa_i$ (for $i \in \mathbb{N}$), $\kappa_L, \kappa_R$ (for “left” and “right”) are coprojections; and $p_i : N \to 1$ is a “projection” map in [33] (see also Thm. AppendixC.5) that satisfies

$$p_i \odot \kappa_j = \begin{cases}
\text{id}_1 & \text{if } i = j, \\
\bot & \text{otherwise},
\end{cases}$$

44
where \( \odot \) denotes composition of arrows in \( \mathcal{K}(\mathcal{Q}) \) and \( \perp \) is the least element (the "zero map") in the homset \( \mathcal{K}(\mathcal{Q})(1,1) \).

**Lemma 5.7.** We have \( D_j \cdot (x_i)_{i \in \mathbb{N}} = x_j \). Here \( \cdot \) denotes the applicative structure of \( A_{\mathcal{Q}} \) (Thm. 4.10).

**Proof.** By easy manipulation of string diagrams, the claim boils down to the equality
\[
\left( \prod_i x_i \right) \odot (\kappa_j \cdot \mathbb{N}) = \left( \prod_i x_i \right) \odot (\kappa_j \cdot \mathbb{N}) = x_j .
\]
This follows from the fact that \( (\prod_i x_i) \odot (\kappa_j \cdot \mathbb{N}) = \left( \prod_i x_i \right) \odot (\kappa_j \cdot \mathbb{N}) = \kappa_j \cdot \mathbb{N} \odot x_j \)
and that \( p_j \odot \kappa_j = \text{id} \).

**Proposition 5.8.** The category \( \text{PER}_{\mathcal{Q}} \) has countable limits and colimits.

**Proof.** Constructions of equalizers and coequalizers in realizability categories are described in [34]; they work in the current setting of PERs over an LCA, too. Concretely: given \( \langle c, d \rangle : X \rightarrowtail Y \) in \( \text{PER}_{\mathcal{Q}} \), let
\[
E := \{ (x, x') \mid (x, x') \in X \wedge (c x, d x') \in Y \} ; \text{ and } C := \{ (y, y') \mid (y, y') \in Y \} \cup \{ (c x, d x') \mid (x, x') \in X \} .
\]
Then it is straightforward to see that \( \llbracket l \rrbracket : E \rightarrow X \) and \( \llbracket l \rrbracket : Y \rightarrow C \) are an equalizer and a coequalizer of \( \llbracket c \rrbracket \) and \( \llbracket d \rrbracket \), respectively.

It suffices to show that \( \text{PER}_{\mathcal{Q}} \) has countable products and coproducts.

Given a countable family \( \{ X_i \}_{i \in \mathbb{N}} \) of objects of \( \text{PER}_{\mathcal{Q}} \), we use the constructs in Def. 5.6 and define
\[
\prod_{i \in \mathbb{N}} X_i := \{ (p(k_i)_{i \in \mathbb{N}} u, p(k'_i)_{i \in \mathbb{N}} u') \mid (k_i u, k'_i u') \in X_i \text{ for each } i \in \mathbb{N} \} ;
\]
\[
\pi_i := [\lambda w. w(\lambda vu. (D_i v) u)] : \prod_{i \in \mathbb{N}} X_i \rightarrow X_i .
\]
Then, given a family \( \{ [c_i] : Y \rightarrow X_i \}_{i \in \mathbb{N}} \) of arrows, its tupling can be given by
\[
\langle [c_i] \rangle_{i \in \mathbb{N}} := [\lambda y. p[c_i]_{i \in \mathbb{N}} y] : Y \rightarrow \prod_{i \in \mathbb{N}} X_i .
\]

On coproducts, we define
\[
\coprod_{i \in \mathbb{N}} X_i := \{ (p \odot_d x_i, p \odot_d x'_i) \mid i \in \mathbb{N}, (x_i, x'_i) \in X_i \} ;
\]
\[
\kappa_i := [p \odot_d] : X_i \rightarrow \coprod_{i \in \mathbb{N}} X_i .
\]
Given a family \((c_i : X_i \to Y)_{i \in \mathbb{N}}\) of arrows, its cotupling can be given by
\[
[c_i]_{i \in \mathbb{N}} := \left[\lambda w. w(\lambda dx. d(c_i))_{i \in \mathbb{N}} x\right] : \prod_i X_i \to Y.
\]
This concludes the proof. □

**Remark 5.9.** Although we have focused on the specific linear category \(\text{PER}_Q\), what are said in the current §5.1 are true in more general settings.

One point (that is already mentioned) is that the extra canonical isomorphisms in (29) hold in any \(\text{PER}_A\) with an affine LCA \(A\). This makes such categories \(\text{PER}_A\) suitable for modeling linear \(\lambda\)-calculi with implicit linearity tracking.

Another point is about the constructions in Def. 5.6, Lem. 5.7 and Prop. 5.8. It is not hard to see that these are all possible in the category \(\text{PER}_{A_B}\), where the LCA \(A_B\) is obtained via categorical GoI [21, Prop. 4.2] from the Kleisli category \(\mathcal{K}(B)\) for any “branching monad” like \(B = \mathcal{L}, \mathcal{P}, \mathcal{D}\) and \(Q\) (see §2.2).

### 5.2. Quantum Mechanical Constructs in \(\text{PER}_Q\)

**Definition 5.10 (Combinator A).** We define \(A \in A_Q\) by the string diagram in \(\mathcal{K}(Q)\) shown below. The triangles are \(j : \mathbb{N} + \mathbb{N} \cong \mathbb{N} : k\) in Thm. 4.10. It is easily seen to satisfy the equation
\[
Axy = x \circ y,
\]
where \(\circ\) denotes composition of arrows in \(\mathcal{K}(Q)\) (see §4.2).

\[
A := \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \ quad
In the definition of $Q_\rho$, $\otimes$ denotes tensor product of matrices. The combinator $Q_\rho$ “adjoins an auxiliary state $\rho$”: an incoming token carrying $\sigma$ comes out of the same pipe, with its state composed with $\rho$. In particular the state $1 \in \text{DM}_1$ comes out as $\rho$. Similarly, the combinator $Q_U$ applies the unitary transformation $U$ to (the first $N$ qubits of) the incoming quantum state; and $Q_{N+1}^{0i}$ and $Q_{N+1}^{1i}$ apply suitable projections (cf. Rem. 2.11). Indeed:

**Lemma 5.12.** We have the following equalities.

$$Q_{\rho} \otimes Q_{\sigma} = Q_{\rho \otimes \sigma}, \quad Q_{U \rho U^\dagger} = Q_{U \rho U^\dagger}, \quad Q_{N+1}^{0i} \circ Q_{\sigma} = Q_{(0_i | \sigma | 0_i)_1}, \quad Q_{N+1}^{1i} \circ Q_{\sigma} = Q_{(1_i | \sigma | 1_i)_1}$$

**Definition 5.13 ([N-qbit],[bit]).** For each $N \in \mathbb{N}$ we define a PER $[N\text{-qbit}]$ over $\mathbb{A}_Q$ by:

$$[N\text{-qbit}] := \{ (Q_\rho, Q_\rho) \mid \rho \in \text{DM}_{2^N} \}.$$  

In particular, $[0\text{-qbit}] = \{ (Q_\rho, Q_\rho) \mid \rho \in [0,1] \}$ (cf. Def. 2.3). This type can be thought of as the unit interval $[0,1]$.

A PER $[\text{bit}]$ is defined to be $I + I$ (see Lem. 5.2).

The following fact supports the idea that $!$ stands for duplicable, hence classical, data.

**Lemma 5.14.** There is a canonical isomorphism $[\text{bit}] \cong [!\text{bit}]$ in PER$_Q$.

**Proof.** Use the isomorphisms (29) in Lem. 5.3.

The quantum combinators in Def. 5.11 are combined with the A combinator in Def. 5.10 and yield the following combinators for quantum operations.

**Definition 5.15 (Combinators $U_U$, $Pr_{N+1}^{0i}$, $Pr_{N+1}^{1i}$).** We define

$$U_U := AQ_U, \quad Pr_{N+1}^{0i} := AQ_{N+1}^{0i}, \quad Pr_{N+1}^{1i} := AQ_{N+1}^{1i}.$$  

**Lemma 5.16.** For combinators in Def. 5.11 and 5.15, and $\rho, \sigma, U$ of suitable dimensions, we have

$$AQ_{\rho} Q_{\sigma} = Q_{\rho \otimes \sigma}, \quad U_U Q_{\rho} = Q_{U \rho U^\dagger}, \quad Pr_{N+1}^{0i} Q_{\sigma} = Q_{(0_i | \sigma | 0_i)_1}, \quad Pr_{N+1}^{1i} Q_{\sigma} = Q_{(1_i | \sigma | 1_i)_1}.$$  

**Proof.** Obvious from Lem. 5.12.

### 5.3. Continuation Monad $T$

Our categorical model PER$_Q$ employs another monad $T$ in addition to $Q$: the interpretation of a type judgment $\Delta \vdash M : A$ will be an arrow $[\Delta] \to T[A]$ in the category PER$_Q$. The monad $T$ is in fact a continuation monad $T = (\_ \rightarrow R) \rightarrow R$ with a suitable result type $R$; hence our semantics is in the continuation-passing style (CPS). The resulting CPS model is fairly complex as

47
a matter of fact, but our efforts for its simplification have so far been barred by technical problems, leading us to believe that CPS is a right way to go. Informally, the reason is as follows.

Think of the construct $\text{meas}_1$ that measures one qubit; for the purpose of case-distinction based on the outcome, it is desired that $\text{meas}_1$ is of the type $\text{qbit} \to \text{bit}$. Therefore it is natural to use monadic semantics \[38\]: we use a monad $T$—with a probabilistic flavor—so that we have $[\text{meas}_1] : [\text{qbit}] \to T[\text{bit}]$.

For our GoI semantics based on local interaction, however, a simple “probability distribution” monad (something like $D$ in §2.2) would not do. One explanation is as follows. Think of the construct $\text{meas}_2 : 2\text{-qbit} \to \text{bit} \otimes \text{qbit}$: it takes a state $\rho$ of a 2-qubit system; measures the first qubit; and returns its outcome ($t$ or $f$) together with the remaining qubit. The probability of observing $tt$ is calculated by $\text{tr} \left( \left( \langle 0_1 | \_ | 0_1 \rangle \otimes I_2 \right) \rho \right)$, and use of a naive “probability distribution monad” requires calculation of the explicit value of this probability. However, the calculation traces out the second qubit, destroys and leaves it inept for further quantum operations.

(To put it differently: since we let a quantum state $\rho$ implicitly carry a probability in the form of its trace value $\text{tr}(\rho)$, a naive interpretation of $\text{meas}_2$ would have the codomain $\text{bit} \otimes \text{qbit}$—with entanglement—rather than the desired codomain $\text{bit} \otimes \text{qbit}$ that goes along well with ! and recursion.)

Hence we need to postpone such calculation of probabilities until the very end of computation. Use of continuations is a standard way to do so. As a result type $R$, we take that of complete binary trees with each edge labeled with a real number $p \in [0, 1]$—obtained as a final coalgebra.

**Definition 5.17 (The functor $F_{\text{pbt}}$).** We define an endofunctor $F_{\text{pbt}} : \text{PER}_Q \to \text{PER}_Q$ by

$$F_{\text{pbt}} := [\text{bit}] \to ([0\text{-qbit}] \hat{} \times \_)$$

where the objects $[\text{bit}]$ and $[0\text{-qbit}]$ are as in Def. 5.13.

In the functor $F_{\text{pbt}}$ we use $\hat{}$ instead of $\times$ for Cartesian products. This ensures a good order-theoretic property (namely admissibility) of the carrier $R$ of the final coalgebra.

The functor $F_{\text{pbt}}$ represents the branching type of probabilistic binary trees like the one shown below. In the functor, the $[\text{bit}]$ part designates which of the left and right successors; and the $[0\text{-qbit}]$ part designates the value $p_i \in [0, 1]$ that is assigned to the edge (here $i \in \{0, 1\}$).

\[
\begin{array}{c}
p_0 \\
\bullet \\
p_1 \\
\end{array}
\]

(33)
The values $p_0$ and $p_1$ can be thought of as probabilities, although they might not add up to 1.

As is usual in the theory of coalgebras (see e.g. [39, 40]), the collection of such trees is identified with (the carrier of) a final coalgebra.

**Lemma 5.18** (The result type $R$). *The functor $F_{\text{pbt}}$ has a final coalgebra*

$$r : R \xrightarrow{\cong} F_{\text{pbt}} R.$$  \hspace{2cm} (34)

**Proof.** We use the standard construction by a *final sequence* (see e.g. [58, 59]). Let

$$\text{Bt} := \{ (\bot, \bot) \}$$

be a final object; although $\text{Bt} \cong I$, we use $\text{Bt}$ due to its order-theoretic property. Consider the final sequence

$$\text{Bt}\xrightarrow{\text{weak}} F_{\text{pbt}} \text{Bt} F_{\text{pbt}}^2 \text{Bt} \xrightarrow{\text{weak}} \cdots$$  \hspace{2cm} (35)

in $\text{PER}_Q$. Here weak denotes the unique arrow to final $\text{Bt}$. For $j \leq i$, let $c_{i,j}$ be (the canonical choice of) a realizer of the arrow $F_{\text{pbt}}^i \text{Bt} \to F_{\text{pbt}}^j \text{Bt}$ in the final sequence.

By Prop. 5.8 there is a limit $R$ of the sequence (34). Moreover the PER $R$ can be concretely described as: the symmetric closure of

$$\{ \{ \hat{P}(k_i)u, \hat{P}(k'_i)u' \} \mid j \leq i \text{ implies } (c_{i,j}(k_i u), k'_i u') \in F_{\text{pbt}}^j \text{Bt} \text{ and } (k_j u, c_{j,i}(k'_i u')) \in F_{\text{pbt}}^j \text{Bt} \}.$$  \hspace{2cm} (36)

Now the functor $F_{\text{pbt}}$ preserves limits: $[0\text{-qbit}] \times -$ does since $\times$ is for products; and $[\text{bit}] \xrightarrow{\text{left adj}} -$ does since it has a left adjoint $[\text{bit}] \boxtimes -$ (Lem. 5.2). Therefore the well-known argument (see e.g. [58, 59]) proves that $R$ carries a final $F_{\text{pbt}}$-coalgebra, with the coalgebraic structure $r : R \cong F_{\text{pbt}} R$ obtained as a suitable mediating arrow. \hfill $\square$

**Lemma 5.19.** Let

$$T := (\_ \xrightarrow{\_ \circ} R) \circ R.$$  \hspace{2cm} (37)

Then $T$ is a strong monad on $\text{PER}_Q$.

**Proof.** This is a standard fact that is true for any monoidal closed category $(\mathcal{C}, I, \boxtimes, \_ \circ)$ and for any $R \in \mathcal{C}$. \hfill $\square$

We introduce a map

$$\text{mult} : [0\text{-qbit}] \boxtimes R \to R$$

that will be needed later. Intuitively, what it does is to receive $p \in [0, 1]$ and a binary tree $t$ like (33) and returns the tree in which the probabilities assigned
to all the edges are multiplied by $p$. For example,

$$
\begin{pmatrix}
1 & 1  \\
2 & 1  \\
3 & 1  \\
\vdots & \vdots
\end{pmatrix}
\quad \overset{\text{mult}}{\mapsto}
\begin{pmatrix}
1  \\
2  \\
3  \\
\vdots
\end{pmatrix}
$$

The precise definition of mult is by coinduction as a definition principle.

**Definition 5.20 (mult).** Let a coalgebra

$$
c_{\text{mult}} : \llbracket 0\text{-qbit} \otimes R \rrbracket \rightarrow F_{\text{pbt}}(\llbracket 0\text{-qbit} \otimes R \rrbracket) \quad \text{in PER}_Q
$$

be defined the adjoint transpose of the following composite.

$$
\begin{array}{c}
\llbracket 0\text{-qbit} \otimes R \otimes \llbracket \text{bit} \rrbracket \xrightarrow{id \otimes \text{Id}} \llbracket 0\text{-qbit} \otimes (\llbracket \text{bit} \rrbracket \otimes R) \otimes \llbracket \text{bit} \rrbracket \\
\llbracket \lambda w.w \| A \rrbracket \otimes \text{Id}
\end{array}
$$

Here the map $\llbracket \lambda w.w \| A \rrbracket : \llbracket 0\text{-qbit} \otimes \llbracket 0\text{-qbit} \rrbracket \rightarrow \llbracket 0\text{-qbit} \rrbracket$—that carries $P_{xy}$ to $x \otimes y$—plays the role of multiplication over $[0, 1]$. By finality, this coalgebra $c_{\text{mult}}$ induces a unique arrow from $c_{\text{mult}}$ to $r$. This is denoted by mult.

$$
F_{\text{pbt}}(\llbracket 0\text{-qbit} \otimes R \rrbracket) \xrightarrow{\text{mult}} F_{\text{pbt}}R
$$

5.4. Admissible PER and Fixed Point Operator

The proofs for §5.4 are deferred to Appendix AppendixD.

We shall interpret recursion in Hoq using the $\omega$-CPO structure of $A_Q$; this is like in [60].

**Lemma 5.21 (A$_Q$ is an $\omega$-CPO).** The set $A_Q$ is an $\omega$-CPO with the smallest element $\perp$. Furthermore:

1. the application operator $\cdot : A_Q^2 \rightarrow A_Q$ and the $!$ operator are continuous; and
2. application $\cdot$ is left strict, that is, $\perp \cdot a = \perp$ for each $a \in A_Q$. \hfill $\Box$

The order on $A_Q$ comes from the $\omega$-CPO enriched structure of $K\ell(Q)$, hence is essentially the Löwner partial order (Def. 2.4).

The following notion of admissibility is standard. See e.g. [61].

**Definition 5.22 (Admissible PER).** A PER $U \in $ PER$_Q$ is said to be admissible if:
• (strictness) \((\bot, \bot) \in U\) for the least element \(\bot \in A_Q\); and

• (inductiveness) \(x_0 \sqsubseteq x_1 \sqsubseteq \cdots, y_0 \sqsubseteq y_1 \sqsubseteq \cdots\) and \((x_i, y_i) \in U\) for each \(i \in \mathbb{N}\) imply \((\sup_i x_i, \sup_i y_i) \in U\).

We note that admissibility is not preserved by isomorphisms in \(\text{PER}_Q\). Observe \(I = \{(i, 1)\} \cong \{(\bot, \bot)\} = B_t\).

Lemma 5.23. For admissible \(U, V\) and any \(X\), we have \(U \times V\) and \(X \to U\) admissible. \(\Box\)

Lemma 5.24. The \(\text{PER} R\) in (35) is admissible. Therefore by Lem. 5.23, \(TX = (X \to R) \to R\) is admissible for each \(X\); so is \(X \to TY\). \(\Box\)

Definition 5.25 (Fixed point operator). Let \(U, X \in \text{PER}_Q\); assume that \(U\) is admissible. We introduce a fixed point operator (denoted by \(\text{fix}\)) that carries \(f : !X \to U\) to \(\text{fix}(f) : !X \to U\) in the following way. Let \(c\) be a code of \(f\). We define \(c_0, c_1, \ldots \in |!X \to U|\) by

\[
c_0 := \bot ;
\]

\[
c_{n+1} := \text{the canonical code of } (\text{con} !X \otimes !X \otimes \text{id} !X \to ![c_n] \otimes \text{id} !U \otimes !X \otimes ![c]) .
\]

A concrete description of \(c_{n+1}\) in terms of \(c_n\) can be easily given using in particular (27). Since \(U\) is admissible and \(\bot \cdot x = \bot\), \(c_0 = \bot\) is a valid code. It is not hard to show that \(c_0 \sqsubseteq c_1 \sqsubseteq \cdots\) by induction; since \(!X \to U\) is admissible its supremum \(\sup_i c_i\) belongs to the domain \(|!X \to U|\). Finally, we define

\[
\text{fix}(f) := [\sup_i c_i] .
\]

It is easily seen too that the above definition of \(\text{fix}(f)\) does not depend on the choice of a code \(c\) of \(f\). Here admissibility of \(U\) is crucial.

5.5. Interpretation

Definition 5.26 (Interpretation of types). For each Hoq-type \(A\), we assign \([A] \in \text{PER}_Q\) as follows, using the constructors in Lem. 5.2. For base types, \([N\text{-qbit}]\) is as in Def. 5.13.

\[
\begin{align*}
[!] A & := [A] \\
[\top] & := 1 \\
[ A + B] & := [A] + [B]
\end{align*}
\]

Definition 5.27 (Interpretation of the subtype relation). We shall assign, to each derivable subtype relation \(A <: B\), an arrow

\[
[A <: B] : [A] \to [B] \quad \text{in } \text{PER}_Q.
\]
For that purpose we first introduce a natural transformation

\[ \delta_{n,m} : \texttt{!}^n X \rightarrow \texttt{!}^m X \] natural in \( X \),

for each \( n, m \in \mathbb{N} \) that satisfy \( n = 0 \Rightarrow m = 0 \). This is as follows.

\[
\delta_{n,m} := \begin{cases} 
\text{id} & \text{if } n = m \\
\delta \circ \cdots \circ \delta & \text{if } n < m \text{ (note that in this case } n > 0) \\
\text{der} \circ \cdots \circ \text{der} & \text{if } n > m
\end{cases}
\]

Using \( \delta_{n,m} \), an arrow \([A <: B]\) is defined by induction on the derivation (that is according to the rules in (11)).

\[
[\texttt{!}^n \texttt{k-qbit} <: \texttt{!}^m \texttt{k-qbit}] := (\texttt{!}^n [\texttt{k-qbit}] \xrightarrow{\delta_{n,m}} \texttt{!}^m [\texttt{k-qbit}]) ,
\]

\[
[\texttt{!}^n \top <: \texttt{!}^m \top] := (\texttt{!}^n [\top] \xrightarrow{\delta_{n,m}} \texttt{!}^m [\top]) ,
\]

\[
[\texttt{!}^n (A_1 \otimes A_2) <: \texttt{!}^m (B_1 \otimes B_2)] := (\texttt{!}^n ([A_1] \otimes [A_2]) [A_1 <: B_1])_A \xrightarrow{\delta_{n,m}} \texttt{!}^m ([B_1] \otimes [B_2]) \xrightarrow{\delta_{n,m}} \texttt{!}^m (([B_1] \otimes [B_2])) .
\]

\([\texttt{!}^n (A_1 + A_2) <: \texttt{!}^m (B_1 + B_2)]\) and \([\texttt{!}^n (A_1 \rightarrow A_2) <: \texttt{!}^m (B_1 \rightarrow B_2)]\) are defined in a similar manner.

It is obvious from the rules in (11) that a derivable judgment \( A <: B \) has only one derivation. Therefore \([A <: B]\) is well-defined.

**Lemma 5.28.** Let \( A <: B \) and \( B <: C \). Then \( A <: C \) by Lem. 3.15.1; moreover \([A <: C] = [B <: C] \circ [A <: B] \).

**Proof.** Much like the proof of Lem. 3.15.1, in whose course we exploit naturality of \( \delta_{n,m} \), and that \( \delta_{n,k} = \delta_{m,k} \circ \delta_{n,m} \). The latter follows from Lem. 5.3. \(\square\)

We now interpret constants. In the general definition (Def. 5.31) a typed term \( A + M : A \) will be interpreted as an arrow \([M] : [\Delta] \rightarrow T[A] \)—the monad \( T \) is there because of our CPS semantics. For constants however we do not need \( T \): intuitively this is because a constant \( c \) can always have the type \( \texttt{!DType}(c) \) (see \( \texttt{Ax2} \) in Table 1). Therefore we first define \([c]_{\text{const}} : I \rightarrow [\texttt{DType}(c)]\) whose descriptions are simpler, and then \([c] : I \rightarrow T[\texttt{DType}(c)]\) will be defined to be the embedding via the unit \( \eta_T : \text{id} \Rightarrow T \) of the monad \( T \).

The technical core is in the interpretation of measurements. We explain its idea after its formal definition.

**Definition 5.29 (Interpretation of constants).** To each constant \( c \) in Hoq we assign an arrow

\([c]_{\text{const}} : I \rightarrow [\texttt{DType}(c)]\)

as follows. For \( c \equiv \texttt{new},

\([\texttt{new}]_{\text{const}} : I \rightarrow [\texttt{bit} \rightarrow \texttt{qbit}] = ([\texttt{bit}] \rightarrow T[\texttt{qbit}])\)

\[52\]
is the adjoint transpose of

\[
\text{[[bit]]} = I + I \left[ \lambda x. Q_{0}(0) \cdot \lambda x. Q_{1}(1) \right] \xrightarrow{\eta^T} \text{[qbit]} \xrightarrow{T} \text{[qbit]}. 
\]

For \(c \equiv \text{meas}^{n+1}\) with \(n \geq 1\), by transpose we need an arrow

\[
\text{eval} \circ ([n+1]-\text{qbit}) \otimes ((\text{[bit]} \otimes [n-\text{qbit}]) \rightarrow \circ) \xrightarrow{m_{R}} R ,
\]

where we also used \(\text{eval} \equiv \text{[bit]}\) (Lem. 5.14). By \(R\)'s fixed point property (namely \(r : R \Rightarrow [\text{bit}] \rightarrow (0-qbit) \times R\)), this is further reduced to an arrow

\[
\text{eval} \circ ([n+1]-\text{qbit}) \otimes ([\text{bit}] \otimes [n-\text{qbit}]) \rightarrow \circ [\text{bit}] \rightarrow [0-qbit] \times R .
\]

This can be obtained as follows.

\[
\text{eval} \circ ([n+1]-\text{qbit}) \otimes ([\text{bit}] \otimes [n-\text{qbit}]) \rightarrow \circ [\text{bit}]
\]

\[
\cong ([n+1]-\text{qbit}) \otimes (([\text{bit}] + [n-\text{qbit}]) \rightarrow \circ R) \otimes \text{[bit]} \quad \text{by (30)} \quad \text{and} \quad I \otimes X \cong X
\]

\[
\cong ([n+1]-\text{qbit}) \otimes (\text{[bit]} \rightarrow R) \times ([n-\text{qbit}] \rightarrow \circ) \otimes \text{[bit]} \quad \text{by (30)}
\]

\[
\cong ([n+1]-\text{qbit}) \otimes ([n-qbit] \rightarrow \circ R) \times ^2 + ([n+1]-\text{qbit}) \otimes (\text{[bit]} \rightarrow \circ R) \times ^2 \quad \text{by (30)}
\]

\[
\xrightarrow{\text{eval}} \text{eval} \circ ([\text{bit}] \otimes [n-qbit] \rightarrow \circ R) + \text{[bit]} \otimes ([n-qbit] \rightarrow \circ R)
\]

\[
\cong \text{eval} \circ ([\text{bit}] \rightarrow \circ R) \xrightarrow{m^R} \circ R ,
\]

Here \(\text{Pr}_{i_{0}+1}^{n+1}\) and \(\text{Pr}_{i_{1}+1}^{n+1}\) are from Def. 5.15, and \(Q_{0}\) is from Def. 5.11 (see also Def. 5.13).

For \(c \equiv \text{meas}^{1}\), similarly, by transpose we need an arrow

\[
\text{eval} \circ ([\text{bit}] \rightarrow \circ R) \xrightarrow{m^R} \circ R ,
\]

which is equivalent to (by \(R\) being a final coalgebra)

\[
\text{eval} \circ ([\text{bit}] \rightarrow \circ R) \otimes \text{[bit]} \rightarrow [0-qbit] \times R .
\]

This is obtained as follows.

\[
\text{eval} \circ ([\text{bit}] \rightarrow \circ R) \otimes \text{[bit]}
\]

\[
\cong [\text{bit}] \otimes ([\text{bit}] \rightarrow \circ R) + \text{[bit]} \otimes ([\text{bit}] \rightarrow \circ R) \quad \text{by (30)}
\]

\[
\cong [\text{bit}] \otimes R \times ^2 + \text{[bit]} \otimes R \times ^2 \quad \text{by (30)}
\]

\[
\xrightarrow{\text{eval} \circ \text{eval}} [0-qbit] \otimes R + [0-qbit] \otimes R
\]

\[
\cong [0-qbit] \otimes R \xrightarrow{\text{eval}} [0-qbit] \times R ;
\]

Here mul is from Def. 5.20.

For the other constants we use Lem. 5.2, Def. 5.11 and Def. 5.15. The arrow \([U]_{\text{const}}\) is the transpose of

\[
\text{eval} \circ ([\text{bit}] \rightarrow \circ R) \xrightarrow{m^R} \circ R .
\]
\[ \text{const} \text{ is the transpose of } \]
\[ [(n+q)\text{-qbit}] \xrightarrow{\lambda w. wA} [(m + n)\text{-qbit}] \xrightarrow{\eta^T} T[(m + n)\text{-qbit}]; \]
\[ \text{const is } \]
\[ \text{I}^{\lambda x. Q} [(n)\text{-qbit}]. \]

The use of \( Q_0 \) (that stands for the value \( 0 \in [0,1] \)) in the last line indicates that a tree \( m(t) \in |R| \) that can arise as an outcome of the map \( m \) in (36) looks as follows.

This is strange if we think of the values attached to edges as probabilities. In fact they are not probabilities: as we discussed in the beginning of \( \S 5.3 \), the actual probabilities are carried implicitly by the remaining quantum states (consisting of \( n \) qubits) as their trace values. The labels 0 in (38) mean calculation of probabilities is postponed; they are done later and probabilities occur on some lower level in the tree (38).

More specifically, the idea for \( \text{meas}^{n+1} \) is as follows. Let \( n \geq 1 \) and consider the map \( m \) in (36), which can be identified (via Lem. 5.3) with an arrow
\[ [(n+1)\text{-qbit}] \boxtimes [(n)\text{-qbit}] \twoheadrightarrow R \times \overrightarrow{\mathbb{R}} R. \]
Roughly speaking its input is a triple \( (\rho, f_{tt}, f_{ff}) \) of \( \rho \in \text{DM}^{2n+1} \) and \( f_{tt}, f_{ff} : \text{DM}^{2n} \to R \). Then \( m \)'s output is the following tree.

Here we put 0 as the labels on the edges of depth one; the probabilities for observing \( |0_i\rangle \) or \( |1_i\rangle \) are implicitly passed down in the form of the trace of the projected matrices.

When there is only one qubit left, we finally compute actual probabilities. This is what \( \text{meas}^{1} \) does. Consider \( m' \) in (37), which can be identified with
\[ [(q)\text{-qbit}] \boxtimes (R \times R) \overset{\overrightarrow{m}}{\rightarrow} R. \]
Its input is roughly a triple \( (\rho, t_{tt}, t_{ff}) \) of \( \rho \in \text{DM}_2 \) and trees \( t_{tt}, t_{ff} \in R \). Let \( p = (0|\rho|0) \) and \( q = (1|\rho|1) \); these are the probabilities for each outcome of the measurement. Then the output of \( \overrightarrow{m}' \) is the following tree.
Recall that \( \text{mult}(p,t) \) multiplies all the labels of the input tree \( t \) by \( p \).

This way we only generate edges with its label 0. This is no problem: once we use trees with nonzero labels as \( t_\pi \) and \( t_\pi \) in the above, we observe nonzero probabilities.

The following definition of interpretation of type judgments looks rather complicated. It is essentially the usual definition, as with other typed (linear) calculi. The subtype relation needs careful handling, however—especially so that well-definedness (Lem. 5.32) holds—and this adds all the details.

**Definition 5.30 (Interpretation of contexts).** We fix an enumeration of variables, i.e., a predetermined linear order \( \prec \) between variables. Given an (unordered) context \( \Delta = (x_i : A_i)_{i \in [1,n]} \), we define \( [\Delta] \in \text{PER}_Q \) by \( [\Delta_{\sigma(1)}] \otimes \cdots \otimes [\Delta_{\sigma(n)}] \), where \( \sigma \) is a bijection s.t. \( x_{\sigma(1)} \prec \cdots \prec x_{\sigma(n)} \).

**Definition 5.31 (Interpretation of type judgments).** For each derivation \( \Pi \vdash \Delta \vdash M : A \) of a type judgment in Hoq, we assign an arrow

\[
[\Pi] : [\Delta] \to T[A]
\]

in the following way. First we define

\[
[\Pi]_{\text{FV}} : [\Delta]_{\text{FV}(M)} \to T[A]
\]

by induction on the derivation, which is used in

\[
([\Delta] \xrightarrow{[\Pi]} T[A]) := ([\Delta] \xrightarrow{\text{weak}} [\Delta]_{\text{FV}(M)}) \xrightarrow{[\Pi]_{\text{FV}}} T[B]
\]

The definition of \([\Pi]_{\text{FV}}\) as follows.

\[
\begin{align*}
\text{Ax.1} & \quad [A] \xrightarrow{[\Delta \prec A']} [A'] \xrightarrow{\eta^T} T[A'] \\
\text{Ax.2} & \quad I \xrightarrow{\text{weak}} ![\text{const} (\text{cf. Def. 5.29})] \xrightarrow{[\Delta \prec \text{Type}]} T [[\Delta \prec \text{Type}]] \xrightarrow{[\Delta \prec A]} T[A]
\end{align*}
\]

**-o.I_1**

\[
[\Delta]_{\text{FV}(1+A,M)} \xrightarrow{\eta^g} T[B] \xrightarrow{[\Delta \prec A]} T[B] = ([A \to B] \xrightarrow{[\Delta \prec A]} T[B]) \xrightarrow{[\Delta \prec A]} T[A' \to B],
\]

where \( g = [\Delta]_{\text{FV}} \) if \( x \in \text{FV}(M) \); otherwise \( g \) is the adjoint transpose of

\[
([A] \otimes [\Delta]_{\text{FV}(M)}) \xrightarrow{\text{weak}} [\Delta]_{\text{FV}(M)} \xrightarrow{[\Pi]_{\text{FV}}} [B]
\]

**-o.I_2**

\[
([\Delta, \Gamma])_{\text{FV}(1+A,M)} = ![\Delta]_{\text{FV}(1+A,M)} \xrightarrow{\delta} \eta^{n+1} [\Delta]_{\text{FV}(1+A,M)} \xrightarrow{\eta^g} \eta^n ([A] \to T[B])
\]

\[
([A \prec A] \xrightarrow{\eta^n} ([A'] \to T[B]) = ([A' \to B] \xrightarrow{\eta^T} T[A' \to B]),
\]

where \( g \) is defined as in the case -o.I_1.

**-E**

\[
([\Delta, \Gamma_1, \Gamma_2])_{\text{FV}(M,N)} \xrightarrow{\text{ev}} ([\Delta, \Gamma_1])_{\text{FV}(M)} \otimes ([\Delta, \Gamma_2])_{\text{FV}(N)}
\]

\[
[M]_{\text{FV}} \otimes [N]_{\text{FV}} \xrightarrow{T([A] \to T[B]) \otimes T[C]) \xrightarrow{[\Delta \prec A]} T([A] \to T[B]) \otimes T[A] \xrightarrow{T([A] \to T[B]) \otimes T[A]} T([A] \to T[B])
\]

\[
\xrightarrow{T} T([A] \to T[B]) \otimes T[A] \xrightarrow{\text{str}'} T([A] \to T[B]) \otimes T[A] \xrightarrow{\text{ev} \mu \mu} T[B]
\]

55
\[ \langle ! \Delta, \Gamma_1, \Gamma_2 \rangle_{\text{FV}(\langle M_1, M_2 \rangle)} \xrightarrow{\text{con}} \langle ! \Delta, \Gamma_1 \rangle_{\text{FV}(M_1)} \otimes \langle ! \Delta, \Gamma_2 \rangle_{\text{FV}(M_2)} \]
\[ \langle M_1 \rangle_{\text{FV}} \otimes \langle M_2 \rangle_{\text{FV}} \]
\[ T^n \llbracket A_1 \rrbracket \otimes T^n \llbracket A_2 \rrbracket \xrightarrow{\text{str'}} \text{and then } \text{str} \mu T^n \llbracket A_1 \rrbracket \otimes !^n \llbracket A_2 \rrbracket \xrightarrow{\text{Lem. 5.3}} T^n \llbracket [A_1] \rrbracket \otimes [A_2] \]
\[ \langle ! \Delta, \Gamma_1, \Gamma_2 \rangle_{\text{FV}(\llbracket x_1 \rrbracket_{A_1}, x_2 \rrbracket_{A_2} = M \in N)} \xrightarrow{\text{con}} \langle ! \Delta, \Gamma_1 \rangle_{\text{FV}(M)} \otimes \langle ! \Delta, \Gamma_2 \rangle_{\text{FV}(N)} \]
\[ \llbracket M \rrbracket_{\text{FV}} \otimes \llbracket \text{id} \rrbracket_{\text{FV}} \]
\[ T^n \llbracket [A_1] \rrbracket \otimes !^n \llbracket [A_2] \rrbracket \otimes \langle ! \Delta, \Gamma_2 \rangle_{\text{FV}(N)} \]
\[ \text{str'}, \text{Lem. 5.3}, T (\llbracket [A_1] \rrbracket \otimes !^n \llbracket [A_2] \rrbracket \otimes \langle ! \Delta, \Gamma_2 \rangle_{\text{FV}(N)}), \llbracket N \rrbracket_{\text{FV}} \]
\[ +^n \llbracket [A_1] \rrbracket \xrightarrow{\text{Lem. 5.3}} T^n \llbracket [A_1] + [A_2] \rrbracket \]
\[ \langle A_1 \leq [A_2] \rangle \]
\[ T^n \llbracket [A_1] + [A_2] \rrbracket \]
\[ \langle +, \llbracket \rangle \rangle_{\text{FV}(\llbracket (x_1, x_2) \mapsto M_1, x_2 \mapsto M_2 \rangle)} \]
\[ \xrightarrow{\text{con}} \langle ! \Delta, \Gamma \rangle_{\text{FV}(P)} \otimes \langle ! \Delta, \Gamma' \rangle_{\text{FV}(M_1) \cup \text{FV}(M_2)} \]
\[ \llbracket P \rrbracket_{\text{FV}} \xrightarrow{\text{str'}} T^n \llbracket [A_1] \rrbracket + [A_2] \rrbracket \otimes \langle ! \Delta, \Gamma' \rangle_{\text{FV}(M_1) \cup \text{FV}(M_2)} \]
\[ \llbracket \text{str} \mu T^n \llbracket [A_1] + [A_2] \rrbracket \otimes \langle ! \Delta, \Gamma' \rangle_{\text{FV}(M_1) \cup \text{FV}(M_2)} \]
\[ \text{Lem. 5.3}, T \left( T^n \llbracket [A_1] + !^n [A_2] \rrbracket \otimes \langle ! \Delta, \Gamma' \rangle_{\text{FV}(M_1) \cup \text{FV}(M_2)} \right) \]
\[ \text{Lem. 5.3}, T \left( !^n \llbracket [A_1] \rrbracket \otimes \langle ! \Delta, \Gamma' \rangle_{\text{FV}(M_1) \cup \text{FV}(M_2)} \right) + !^n \llbracket [A_2] \rrbracket \otimes \langle ! \Delta, \Gamma' \rangle_{\text{FV}(M_1) \cup \text{FV}(M_2)} \right) \]
\[ T \left( \llbracket [M_1] \rrbracket_{\text{FV}} \otimes \llbracket [M_2] \rrbracket_{\text{FV}} \right) \]
\[ \xrightarrow{\text{Lem. 5.3}} T^2 \llbracket B \rrbracket \xrightarrow{\text{str'}} T \llbracket B \rrbracket \]
\[ \text{where, in } (\ast), \text{weak is applied if needed} \]
\[ \text{rec} \]
\[ \langle ! \Delta, \Gamma \rangle_{\text{FV}(\llbracket (x_1, x_2) \mapsto M_1, x_2 \mapsto M_2 \rangle)} \]
\[ \text{con} \triangleleft \llbracket ! \Delta \rrbracket_{\text{FV}(M)} \otimes \langle ! \Delta, \Gamma \rangle_{\text{FV}(N)} \]
\[ \xrightarrow{\text{Lem. 5.3}, T} T \llbracket C \rrbracket \]
\[ \langle A \xrightarrow{\text{weak}} B \rangle \otimes \langle ! \Delta, \Gamma \rangle_{\text{FV}(N)} \]
\[ \llbracket N \rrbracket_{\text{FV}, \text{weak}} \]
\[ \langle A \xrightarrow{\text{g}} B \rangle \otimes \langle ! \Delta, \Gamma \rangle_{\text{FV}(N)} \]
\[ \text{where } g : \langle ! \Delta \rangle_{\text{FV}(M)} \rightarrow [A \xrightarrow{\text{weak}} B] \text{ is obtained as follows.} \]
\[ \llbracket [M] \rrbracket_{\text{FV}} \]
\[ \langle A \xrightarrow{\text{weak}} B \rangle \otimes \langle ! \Delta \rangle_{\text{FV}(N)} \]
\[ \llbracket B \rrbracket \]
\[ \langle A \xrightarrow{\text{weak}} B \rangle \xrightarrow{\text{Lem. 5.3}} [A \xrightarrow{\text{weak}} B] \text{ is obtained as } [M]_{\text{FV}} \]
\[ \text{possibly with weak applied too);} \]
\[ \langle ! \Delta \rangle_{\text{FV}(M)} \]
\[ \langle A \xrightarrow{\text{weak}} B \rangle \rightarrow [A \xrightarrow{\text{weak}} B] \text{ as its adjoint transpose; and then} \]
\[ \langle A \xrightarrow{\text{weak}} B \rangle \rightarrow [A \xrightarrow{\text{weak}} B] \text{ via the fixed point operator fix in Def. 5.25.} \]

Recall that \text{weak} denotes a unique map \( X \rightarrow I \) to the tensor unit I that is final (Lem. 5.2). The arrows \text{der}, \delta, \varphi, \varphi' and \text{con} are from Thm. 4.13 (see also Lem. 5.3). In the above some obvious elements are omitted: we write \text{weak} in place of \text{weak} \otimes \text{id}, \llbracket M \rrbracket in place of \( \Delta \vdash M : A \), etc. We denote \( f^\triangleleft \)s transpose by \( f^\Delta \). The strength \( X \otimes Y \rightarrow T(X \otimes Y) \) is denoted by \text{str}; \text{str}' stands for \( T X \otimes Y \rightarrow T(X \otimes Y) \). For the rule (rec) we use the fixed point operator from
Def. 5.25; note that the PER \([A] \rightarrow T[B]\) is admissible (Lem. 5.24).

The proof of the following important lemma is rather complicated due to implicit linearity tracking. It is deferred to Appendix AppendixE.

**Lemma 5.32** (Interpretation of well-typed terms is well-defined). If \(\Pi, \Pi'\) are derivations of the same type judgment \(\Delta \vdash M : A\), their interpretations are the same: \([\Pi] = [\Pi']\). Therefore the interpretation \([\Delta \vdash M : A]\) of a derivable type judgment is well-defined. □

To compare with operational semantics (introduced in §3.2), the interpretation \([\Delta \vdash M : A]\) obtained thus is too fine. Hence we go further and extract \(M\)'s denotation which is given by a probability distribution. We do so only for closed terms \(M\) of type \(\text{bit}\). This is standard: for non-bit terms one will find distinguishing contexts of type \(\text{bit}\).

In the following definition, the intuitions are:

\(t_0 = \) (the tree whose labels are all 0),

\[ t_\infty = \begin{array}{c}
1 \\
\hline
0
\end{array} \]  
and \( t_\infty = \begin{array}{c}
0 \\
\hline
1
\end{array} \).

**Definition 5.33** (Trees \(t_0, t_\infty, t_\infty\), and test). Let \(c_{\text{test}}\) be the coalgebra

\[
F_{pbt}(I + I + I) \xrightarrow{c_{\text{test}}} R
\]
whose transpose

\[
[\text{bit}] \boxtimes (I + I + I) \rightarrow [0-\text{qbit}] \times (I + I + I)
\]
is described as follows, using informal notations.

\[
\begin{align*}
\langle \text{tt}, \kappa_1(*) \rangle & \mapsto \langle 1, \kappa_2(*) \rangle, \\
\langle \text{ff}, \kappa_1(*) \rangle & \mapsto \langle 0, \kappa_2(*) \rangle, \\
\langle \text{tt}, \kappa_2(*) \rangle & \mapsto \langle 0, \kappa_2(*) \rangle, \\
\langle \text{ff}, \kappa_2(*) \rangle & \mapsto \langle 0, \kappa_2(*) \rangle, \\
\langle \text{tt}, \kappa_3(*) \rangle & \mapsto \langle 0, \kappa_2(*) \rangle, \\
\langle \text{ff}, \kappa_3(*) \rangle & \mapsto \langle 1, \kappa_2(*) \rangle.
\end{align*}
\]

By coinduction we obtain the following arrow \(c_{\text{test}}\).

\[
F_{pbt}(I + I + I) \xrightarrow{c_{\text{test}}} R \\
\xrightarrow{\cong} \xrightarrow{R}
\]

Now the trees \(t_0, t_\infty, t_\infty : I \rightarrow R\) are defined by

\[ t_0 := c_{\text{test}} \circ \kappa_2, \quad t_\infty := c_{\text{test}} \circ \kappa_1, \quad t_\infty := c_{\text{test}} \circ \kappa_3. \]

The arrow

\[ \text{test} : I \rightarrow ([\text{bit}] \rightarrow R) \]
in \(\text{PER}_Q\) is defined to be the adjoint transpose of \([t_\infty, t_\infty] : I + I \rightarrow R\).

57
Definition 5.34 (Operation prob on trees). For each arrow $t : I \rightarrow R$ thought of as a tree, we define $\text{prob}(t) \in (R \cup \{\infty\})^2$ by:

$$\text{prob}(t) := \left( \sum \{\text{labels on edges going down-left}\}, \sum \{\text{labels on edges going down-right}\} \right)$$

For example, $\text{prob}(t_\pi) = (1, 0)$ and $\text{prob}(t_\tau) = (0, 1)$.

The operation $\text{prob}$ quotients the interpretation $[\Delta \vdash M : A] : [\Delta] \rightarrow T[\llbracket A \rrbracket]$ and yields a denotation relation $\downarrow$, which is to be compared with the big-step operational semantics $\Uparrow$ (Def. 3.10). We note again that we swapped the notations $\downarrow$ and $\llbracket \downarrow \rrbracket$ from the previous version [1].

Definition 5.35 (Denotation relation $\downarrow$). We define a relation $\downarrow$ between closed bit-terms $M$—i.e. those terms for which $\vdash M : \text{bit}$ is derivable—and pairs $(p, q)$ of real numbers, as follows. Such a term $M$ gives rise to an arrow $\text{tree}(M) : I \rightarrow R$ in $\text{PER}_Q$ by:

$$\text{tree}(M) := \left( I \xrightarrow{\text{test}} I \bigotimes I \xrightarrow{\text{test}[\llbracket M : \text{bit} \rrbracket]} ([\text{bit}] \rightarrow R) \bigotimes ([\text{bit}] \rightarrow R) \xrightarrow{\text{ev}} R \right). \quad (39)$$

We say $M \downarrow (p, q)$ if $\text{prob}(\text{tree}(M)) = (p, q)$. Obviously such $(p, q)$ is uniquely determined by $M$.

Example 5.36. Let $H$ be the Hadamard matrix. The term $M := \text{meas}_1(1 1 (H(new \tau \tau))$ is a closed bit-term; it measures the qubit $(|0\rangle + |1\rangle)/\sqrt{2}$. Indeed we have $M \downarrow (1/2, 1/2)$.

6. Adequacy

As we use a continuation monad to capture probabilistic branching raised by measurements, our interpretation of Hoq-terms contains intentional data. For example, the interpretation of a term $\vdash M : \text{bit}$ is a tree in the result type $R$ (Lem. 5.18) that reflects the evaluation tree of $M$. In this section, we show that the operation $\text{prob}$ in Def. 5.34—that reduces a tree in $R$ to a pair $(p, q)$ of probabilities—correctly extracts the evaluation result of $M$, that is, we have $M \downarrow (p, q)$ if and only if $M \Uparrow (p, q)$.

We list several basic properties of our denotational semantics. Their proofs are found in Appendix AppendixF. Many of them follow common patterns found in the study of call-by-value languages, although we need to be careful about the fact that a term can have multiple types (due to subtyping $<$):

Lemma 6.1. Let $E$ be an evaluation context, and $x$ be a variable that does not occur in $E$. Assume that $x : A \vdash E[x] : B$ is derivable. Then for any term $M$ such that $\vdash \Gamma \vdash M : A$, the interpretation $[\Gamma \vdash E[M] : B] : [\Gamma] \rightarrow T[\llbracket B \rrbracket]$ is calculated by

$$[\Gamma \vdash E[M] : B] = \mu_{[\llbracket B \rrbracket]}^T \circ T[x : A \vdash E[x] : B] \circ [\Gamma \vdash M : A]. \quad \square$$
Lemma 6.2. For a closed term \( \vdash M : A \), if there is a reduction \( M \to_1 N \) that is not due to a measurement rule (\( \text{meas}_1\text{-meas}_4 \) in Def. 3.9), then
\[
[\vdash M : A] = [\vdash N : A].
\]
Note that \( \vdash N : A \) is derivable by Lem. 3.23. \( \square \)

Theorem 6.3 (Soundness). For any closed bit-term \( M \) (meaning that \( \vdash M : \text{bit} \) is derivable),
\[
M \uparrow^k (p, q) \quad \text{and} \quad M \downarrow (p', q') \quad \text{imply} \quad (p, q) \leq (p', q').
\]
Here the last inequality is the pointwise one and means \( p \leq p' \) and \( q \leq q' \).

Proof. By induction on \( k \). When \( k = 0 \), if \( M \) is neither \( \text{tt} \) nor \( \text{ff} \), then \( p, q = 0 \) and the statement is true. If \( M \equiv \text{tt} \), then \( p = p' = 1 \) and \( q = q' = 0 \). If \( M \equiv \text{ff} \), then \( p = p' = 0 \) and \( q = q' = 1 \).

When \( k > 0 \), if there is a reduction \( M \to_1 N \) that is not due to a measurement rule, then \( [\vdash M : \text{bit}] \) is equal to \( [\vdash N : \text{bit}] \) by Lem. 6.2. Therefore, \( M \downarrow (p', q') \) if and only if \( N \downarrow (p', q') \). Since \( M \uparrow^k (p, q) \) if and only if \( N \uparrow^{k-1} (p, q) \), we obtain \( p \leq p' \) and \( q \leq q' \) from the induction hypothesis. If \( M \) is of the form \( E[\text{meas}^{n+1}_{qstate}] \), then we have \( M \uparrow^k (p_0 + p_1, q_0 + q_1) \) where \( E[\text{tt}, \text{qstate}_{0|0}] \uparrow^{k-1} (p_0, q_0) \) and \( E[\text{ff}, \text{qstate}_{1|1}] \uparrow^{k-1} (p_1, q_1) \).

By Lem. 6.1 we have
\[
[\vdash E[\text{meas}^{n+1}_{qstate}] : \text{bit}] = \\
\mu_{\text{bit}} \circ T[x : !\text{bit} \otimes \text{n-qbit}] \circ E[\vdash \text{meas}^{n+1}_{qstate} : !\text{bit} \otimes \text{n-qbit}] \\
I \rightarrow T[\text{bit}].
\]
(40)

By the definition of the interpretation of \( \text{meas}^{n+1}_i \), the transpose of the interpretation
\[
[\vdash \text{meas}^{n+1}_{qstate} : !\text{bit} \otimes \text{n-qbit}] : I \rightarrow T[!\text{bit} \otimes \text{n-qbit}]
\]
is equal to
\[
(\{[!\text{bit}] \otimes [\text{n-qbit}]\} \rightarrow R) \\
\xrightarrow{\sim} I \otimes (\{[!\text{bit}] \otimes [\text{n-qbit}]\} \rightarrow R) \\
(\text{qstate} \xrightarrow{\text{const} \otimes \text{id}} [n + 1]-\text{qbit}) \otimes (\{[!\text{bit}] \otimes [\text{n-qbit}]\} \rightarrow R) \xrightarrow{m} R,
\]
(41)
where \( m \) is from (36). Recall that \( [\text{bit}] \xRightarrow{\sim} ![\text{bit}] \); see Lem. 5.14. Under the following identifications
\[
([!\text{bit}] \otimes [\text{n-qbit}]) \rightarrow R \xRightarrow{\sim} ([\text{n-qbit}] \rightarrow R)^{\times 2} \quad \text{and} \quad R \xRightarrow{\sim} ([0\text{-qbit}] \times R)^{\times 2}
\]
59
that are derived from Lem. 5.3, 5.14 and 5.18, the value of (41) at \(\langle f, g \rangle : I \rightarrow ([n\text{-}\text{qbit}] \to R)^{\times 2}\) is

\[
\begin{align*}
\langle \lambda x. Q_0 \rangle, \quad f \rightarrow [0\text{-}\text{qbit}], \quad I \rightarrow [n\text{-}\text{qbit}] & \quad \lambda x. Pr_{(0)}^{x+1} Q_0 \rightarrow ([n\text{-}\text{qbit}] \rightarrow R) \otimes [n\text{-}\text{qbit}] \xrightarrow{ev} R, \\
\langle \lambda x. Q_0 \rangle, \quad g \rightarrow [0\text{-}\text{qbit}], \quad I \rightarrow [n\text{-}\text{qbit}] & \quad \lambda x. Pr_{(0)}^{x+1} Q_0 \rightarrow ([n\text{-}\text{qbit}] \rightarrow R) \otimes [n\text{-}\text{qbit}] \xrightarrow{ev} R \\
\text{where} & \quad I \rightarrow ([0\text{-}\text{qbit}] \times R)^{\times 2}.
\end{align*}
\]

(42)

Here combinators like \(Q_0\), \(Q_\rho\) and \(Pr_{(0)}\) are from Def. 5.11 and 5.15; linear \(\lambda\)-terms like \(\lambda x. Q_0\) denote suitable elements of \(A_Q\) by combinatory completeness; and the arrow \([\lambda x. Q_0]\) is the one in \(\text{PER}_D\) that is realized by \(\lambda x. Q_0 \in A_Q\). From Lem. 5.16, it is easy to see that the last arrow (42) is equal to

\[
\langle [\lambda x. Q_0], \quad \text{ev} \circ (f \otimes [\text{qstate}(0,|1|)]_{\text{const}}), \quad [\lambda x. Q_0], \quad \text{ev} \circ (g \otimes [\text{qstate}(0,|1|)]_{\text{const}}) \rangle,
\]

where \([\_]_{\text{const}}\) is from Def. 5.29. Let us now define \(f_0, f_1 : [n\text{-}\text{qbit}] \to T[\text{bit}]\) to be the following arrows:

\[
\begin{align*}
f_0 & := \left( [n\text{-}\text{qbit}] \xrightarrow{\varphi} I \otimes [n\text{-}\text{qbit}] \xrightarrow{\varphi' \otimes \text{id}} !I \otimes [n\text{-}\text{qbit}] \xrightarrow{\text{id} \otimes [\text{bit} \otimes [n\text{-}\text{qbit}]]} T[\text{bit}] \right), \\
f_1 & := \left( [n\text{-}\text{qbit}] \xrightarrow{\varphi} I \otimes [n\text{-}\text{qbit}] \xrightarrow{\varphi' \otimes \text{id}} !I \otimes [n\text{-}\text{qbit}] \xrightarrow{\text{id} \otimes [\text{bit} \otimes [n\text{-}\text{qbit}]]} T[\text{bit}] \right),
\end{align*}
\]

where \(\varphi' : I \rightarrow !I\) is from Thm. 4.13. By (43), the transpose of (40) is equal to

\[
\langle [\lambda x. Q_0], g_0, [\lambda x. Q_0], g_1 \rangle : [\text{bit}] \rightarrow R \rightarrow R
\]

where we identified \(R\) with \(([0\text{-}\text{qbit}] \times R)^{\times 2}\), and \(g_k : [\text{bit}] \rightarrow R \rightarrow R\) is the transpose of

\[
\langle \lambda x. Q_{(k,|1|)} \rangle, [n\text{-}\text{qbit}] \xrightarrow{f_k} T[\text{bit}] \rangle.
\]

Hence,

\[
\langle \text{tree}(E[\text{meas}^{n+1}\text{qstate}]) \rangle \xrightarrow{\text{ev}} ((0\text{-}\text{qbit}) \times R)^{\times 2}
\]

\[
= \langle [\lambda x. Q_0], \text{tree}(f_0 \circ [\text{qstate}(0,|1|)]_{\text{const}}), [\lambda x. Q_0], \text{tree}(f_1 \circ [\text{qstate}(1,|1|)]_{\text{const}}) \rangle,
\]

where we abused the notation \(\text{tree}\) from (39). This means that the tree \(\text{tree}(E[\text{meas}^{n+1}\text{qstate}])\)

60
can be illustrated as follows.

\[
\begin{array}{c}
\text{tree}(f_0 \circ [\text{qstate}_{(0, |p|0,1)}]_{\text{const}}) \\
\text{tree}(f_1 \circ [\text{qstate}_{(1, |p|1,1)}]_{\text{const}})
\end{array}
\]

Therefore

\[
\text{prob}\left(\text{tree}(\vdash E[\text{meas}_{i}^{n+1} \text{qstate}_p : \text{bit}])\right)
\]

is equal to

\[
(0,0) + \text{prob}(\text{tree}(f_0 \circ [\text{qstate}_{(0, |p|0,1)}]_{\text{const}})) + \text{prob}(\text{tree}(f_1 \circ [\text{qstate}_{(1, |p|1,1)}]_{\text{const}}))
\]

where the summation is pointwise. By Lem. 6.1, we have the following equalities:

\[
\begin{align*}
&f_0 \circ [\text{qstate}_{(0, |p|0,1)}]_{\text{const}} = [\vdash E[\langle \text{tt}, \text{qstate}_{(0, |p|0,1)} \rangle]] \\
&f_1 \circ [\text{qstate}_{(1, |p|1,1)}]_{\text{const}} = [\vdash E[\langle \text{ff}, \text{qstate}_{(1, |p|1,1)} \rangle]].
\end{align*}
\]

Therefore, if

\[
E[\langle \text{tt}, \text{qstate}_{(0, |p|0,1)} \rangle] \Downarrow (p'_0, q'_0) \quad E[\langle \text{ff}, \text{qstate}_{(1, |p|1,1)} \rangle] \Downarrow (p'_1, q'_1),
\]

then \(E[\text{meas}_{i}^{n+1} \text{qstate}_p] \Downarrow (p'_0 + p'_1, q'_0 + q'_1) \geq (p_0 + p_1, q_0 + q_1).\) We can similarly show the statement when the reductions are due to the \(\text{meas}_3\) and \(\text{meas}_4\) rules in Def. 3.9.

We shall now show the other direction: if \(M \not\Downarrow (p, q)\) and \(M \Downarrow (p', q')\), then \((p', q') \leq (p, q)\). Our proof employs the techniques of logical relations (see e.g. [62, 63]) and \(\top\top\)-lifting (see e.g. [61, 64]). We write \(\text{Val}(A)\) for the set of closed values of type \(A\) and \(\text{ClTerm}(A)\) for the set of closed terms of type \(A\). We write \(\text{EC}(A)\) for the set of evaluation contexts \(E\) such that \(x : A \vdash E[x] : \text{bit}\) is derivable.

Firstly, we introduce a relation \(\prec\) between \(\text{PER}_Q(I, T[\text{bit}])\) and \(\text{ClTerm}(\text{bit})\). It is defined by by

\[t \prec M \quad \text{def} \quad \text{if } M \not\Downarrow (p, q) \text{ then } \text{prob}(\text{tree}(t)) \leq (p, q).\]

Secondly we introduce the operation of \(\top\top\)-lifting. Given a relation

\[S \subseteq \text{PER}_Q(I, [A]) \times \text{Val}(A),\]

we define a relation \(S^\top \subseteq \text{PER}_Q([A], T[\text{bit}]) \times \text{EC}(A)\) by

\[S^\top := \{ (k, E) \mid \forall (t, V) \in S. k \circ t \prec E[V]\};\]

Hoshino-san, are the references OK?
and we define a relation $S^\top \subseteq \text{PER}_Q(I, T[A]) \times \text{CTerm}(A)$ by
\[
S^\top := \{ (t, M) \mid \forall (k, E) \in S^\top . \mu^T_{[k, E]} \circ T_k \circ t \ll E[M] \}.
\]
This operation of $\top$-lifting is applied to the following relation $R_A \subseteq \text{PER}_Q(I, [A]) \times \text{Val}(A)$. It is inductively defined for each type $A$.

- $R_i := \{(\text{id}_1, \ast)\}$
- $R_{\eta\cdot\text{qbit}} := \{ ([\text{qstate}_\rho]_{\text{const}}, \text{qstate}_\rho) \mid \rho \in \text{DM}_2^a \}$
- $R_{A \otimes B} := \{ (t \otimes s, (V, W)) \mid (t, V) \in R_A \text{ and } (s, W) \in R_B \}$
- $R_{A \to B} := \{ (t, V) \mid \forall (s, W) \in R_A. (\text{ev}_{[A], [B]} \circ (t \otimes s), V W) \in R_B^\top \}$ (44)
- $R_{t_A} := \{ ((t \circ \phi'), V) \mid (t, V) \in R_A \}$
- $R_{t_A} := \{ (s \circ t, \text{inj}_{\phi'}(V)) \mid (t, V) \in R_A \text{ and } B' \ll B \} \\
\cup \{ (\kappa, t, \text{inj}_{\phi'}(V)) \mid (t, V) \in R_B \text{ and } A' \ll A \}$

Here $\phi' : I \to !I$ is from Thm. 4.13. In order to prove the basic lemma for the logical relation \{RA\}AType, we show some properties of $R_A$.

**Lemma 6.4.**
1. If $(t, V)$ is in $R_A$, then $(\eta^T_{[A]} \circ t, V)$ is in $R_A^\top$.
2. If $(t, V) \in R_A$ and $A' : A'$, then $([A < A'] \circ t, V) \in R_A$.

The following property is much like the admissibility requirement. See e.g. [61].

**Lemma 6.5.** Let $M$ be a closed term of type $A$.
1. $([\Box], M) \in R_A^\top$.
2. If there exists a sequence of realizers $a_1 \subseteq a_2 \subseteq \cdots$ of arrows in $\text{PER}_Q(I, T[A])$ such that $([a_n], M) \in R_A^\top$ for each $n$, then we have $([\forall_{n \geq 1} a_n], M) \in R_A^\top$.

**Theorem 6.6** (Basic Lemma). Let $M$ be a term such that $x_1 : A_1, \ldots, x_n : A_n \vdash M : A$ is derivable. If $(t_i, V_i)$ is in $R_A$, for each $i \in [1, n]$, then the pair
\[
([x_1 : A_1, \ldots, x_n : A_n \vdash M : A] \circ (t_1 \otimes \cdots \otimes t_n), \quad M[V_1/x_1, \ldots, V_n/x_n])
\]
is in $R_A^\top$.

**Proof.** By induction on $M$. When $M \equiv x_i$, we have
\[
[x_1 : A_1, \ldots, x_n : A_n \vdash M : A] \circ (t_1 \otimes \cdots \otimes t_n) = \eta^T_{[A]} \circ [A_i : A] \circ t_i.
\]
By (2) in Lem. 6.4, the pair $([A_i < A] \circ t_i, V_i)$ is in $R_A$. Therefore, by (1) in Lem. 6.4 and (45), we see that
\[
([x_1 : A_1, \ldots, x_n : A_n \vdash M : A] \circ (t_1 \otimes \cdots \otimes t_n), \quad V_i)
\]
is in $R_A^\top$. When $M$ is a constant, see Lem. AppendixF.9–AppendixF.13. When $M$ is $\text{letrec} \ f^A x = N \text{ in } L$, the statement follows from Lem. 6.5. Note that the interpretation of $\text{letrec} \ f^A x = N \text{ in } L$ is given by the least upper bound of a sequence of realizers. The other cases are easy.
Corollary 6.7 (Adequacy). For a closed term $\vdash M : \text{bit}$, we have

$$M \uparrow (p, q) \iff M \downarrow (p, q).$$

Proof. We suppose that $M \uparrow (p, q)$ and $M \downarrow (p', q')$. By Thm. 6.3, we have $(p, q) \leq (p', q')$ on the one hand. On the other hand, by Thm. 6.6 (consider its special case where $M$ is closed), we have $([\vdash M : \text{bit}], M) \in R^{\uparrow\downarrow}_{\text{bit}}$. Since $(\eta_{\text{bit}}, [\bot])$ is easily shown to be in $R^\uparrow_{\text{bit}}$, we obtain $[\vdash M : \text{bit}] \leq M$. Hence $(p', q') \leq (p, q)$. \qed

7. Conclusions and Future Work

We presented a concrete denotational model of a quantum $\lambda$-calculus that supports the calculus’ full features including the $!$ modality and recursion. The model’s construction is via known semantical techniques like GoI and realizability. The current work is a demonstration of the generality of these techniques in the sense that, with a suitable choice of a parameter (namely $B = \mathcal{Q}$ in Fig. 1), the known techniques for classical computation apply also to quantum computation (or more precisely “quantum data, classical control”). Our model is also one answer to the question “Quantum GoI?” raised in [65].

Our semantics is based on so-called particle-style GoI and hence on local interaction of agents, passing a token to each other. This is much like in game semantics [22, 23]; our denotational model, therefore, has a strong operational flavor. We are currently working on extracting abstract machines for quantum computation, much like the classical cases in [24, 25, 26, 27, 28]. In doing so, our current use of the continuation monad $T$ (see §5) is a technical burden; endowing realizers with an explicit notion of memory (or state) [66, 28] seems to be a potent alternative.

Acknowledgments

Thanks are due to Peter Selinger for a number of insightful comments; to Kazuyuki Asada, Bart Jacobs, Prakash Panangaden, Phil Scott, Benoît Valiron and the anonymous reviewers for the earlier version [1] for useful discussions and comments; and to Andy Pitts for his ETAPS’07 talk that inspired the current work.

Acknowledgments

Such “compilation” from a quantum program to an abstract machine is presented in [12]. The functional language considered there is a first-order one, drastically easing the challenge of dealing with classical control.
Appendix A. CPO Structures of Density Matrices and Quantum Operations

For the limit properties of density matrices and quantum operations (such as Lem. 2.5) we employ some basic facts from matrix analysis.

There are various notions of norms for matrices but they are known to coincide in finite-dimensional settings. We will be using the following two.

Definition Appendix A.1 (Norms $\| \|_\text{tr}$ and $\| \|_\text{Fr}$). Given a matrix $A \in M_m$, its trace norm $\| \|_\text{tr}$ is defined by

$$\|A\|_\text{tr} := \text{tr}(\sqrt{A^\dagger A}) .$$

Here the matrix $A^\dagger A$ is positive hence its square root is well defined (see e.g. [37, §2.1.8]). In particular, we have

$$\|A\|_\text{tr} = \text{tr}(A) \quad \text{when } A \text{ is positive.} \quad (A.1)$$

The Frobenius norm $\|A\|_\text{Fr}$ of a matrix $A$ is defined by

$$\|A\|_\text{Fr} := \sqrt{\sum_{i,j} |A_{i,j}|^2} = \sqrt{\text{tr}(A^\dagger A)} .$$

Here $A_{i,j}$ is the $(i, j)$-entry of the matrix $A$, hence the Frobenius norm coincides with the standard norm on $M_m \cong \mathbb{C}^{m \times m}$. The latter equality is immediate by a direct calculation.

The metric induced by $\| \|_\text{tr}$ is called the trace distance and heavily used in [37, §9.2].

Lemma Appendix A.2. 1. For each matrix $A \in M_m$ we have $\|A\|_\text{Fr} \leq \|A\|_\text{tr} \leq m \|A\|_\text{Fr}$; therefore the two norms induce the same topology on the set $M_m$.

2. Both norms $\| \|_\text{Fr}$ and $\| \|_\text{tr}$ are complete.

3. The subset $\text{DM}_m \subseteq M_m$ is closed with respect to both norms $\| \|_\text{Fr}$ and $\| \|_\text{tr}$.

Proof. 1. Let $\lambda_1, \ldots, \lambda_m$ be the (nonnegative) eigenvalues of positive $A^\dagger A$. Then the inequality is reduced to

$$\sqrt{\lambda_1 + \cdots + \lambda_m} \leq \sqrt{\lambda_1} + \cdots + \sqrt{\lambda_m} \leq m \cdot \sqrt{\lambda_1 + \cdots + \lambda_m}$$

which is obvious.

2. $\| \|_\text{Fr}$ is complete because so is $\mathbb{C}$. Then one uses 1.

3. Let $(\rho_k)_{k \in \mathbb{N}}$ be a Cauchy sequence in $\text{DM}_m$. We show that $\lim_k \rho_k$ belongs to $\text{DM}_m$. It is positive because the mapping $(v)|v\rangle : \text{DM}_m \to \mathbb{C}$ is continuous with respect to $\| \|_\text{Fr}$ (hence also to $\| \|_\text{tr}$). Similarly, continuity of $\text{tr}(\_)$ yields that $\text{tr}(\lim_k \rho_k) \leq 1$. $\Box$
We shall henceforth assume the topology on $M_m$ that is induced by either of the norms. It is with respect to this topology that we speak, for example, continuity of the function $\text{tr}(\cdot) : DM_m \to \mathbb{C}$. On the one hand, the Frobenius norm $\| \cdot \|_{Fr}$ is useful since many functions—such as $\text{tr}(\cdot)$—are obviously continuous with respect to it. On the other hand, the trace norm $\| \cdot \|_{tr}$ is important for us due to the following property.

**Lemma Appendix A.3.** Let $(\rho_n)_{n \in \mathbb{N}}$ be a sequence in $DM_m$ that is increasing with respect to the Löwner order in Def. 2.4. Then $(\rho_n)_{n \in \mathbb{N}}$ is Cauchy and hence has a limit in $DM_m$.

**Proof.** For any $n, n' \in \mathbb{N}$ with $n \leq n'$, we have

$$\| \rho_{n'} - \rho_n \|_{tr} = \text{tr}(\rho_{n'} - \rho_n) ; \quad (A.2)$$

where $(*)$ holds since $\rho_{n'} - \rho_n$ is positive (see (A.1)). Now observe that the sequence $(\text{tr}(\rho_n))_{n \in \mathbb{N}}$ is an increasing sequence in $[0, 1]$ hence is Cauchy. Combined with (A.2), we conclude that the sequence $(\rho_n)_{n \in \mathbb{N}}$ in $DM_m$ is Cauchy with respect to $\| \cdot \|_{tr}$. By Lem. Appendix A.2, it has a limit $\lim_n \rho_n$ in $DM_m$. □

**Lemma (Lem. 2.5, repeated).** The relation $\sqsubseteq$ in Def. 2.4 is indeed a partial order. Moreover it is an $\omega$-CPO: any increasing $\omega$-chain $\rho_0 \sqsubseteq \rho_1 \sqsubseteq \cdots$ in $DM_m$ has the least upper bound.

**Proof.** Reflexivity holds because 0 is a positive matrix; transitivity is because a sum of positive matrices is again positive. Anti-symmetry is because, if a positive matrix $A$ is such that $-A$ is also positive, all the eigenvalues of $A$ are 0 hence $A$ itself is the zero matrix.

That $\sqsubseteq$ is an $\omega$-CPO is proved in [7, Prop. 3.6] via the translation into quadratic forms. Here we present a proof using norms. By Lem. Appendix A.3, an increasing $\omega$-chain $(\rho_n)_{n \in \mathbb{N}}$ in $DM_m$ has a limit $\lim_n \rho_n$ in $DM_m$. We claim that $\lim_n \rho_n$ is the least upper bound.

To show that $\rho_k \sqsubseteq \lim_n \rho_n$, consider

$$\langle v | (\lim_n \rho_n) - \rho_k | v \rangle = \lim_n \langle v | \rho_n - \rho_k | v \rangle ; \quad (A.3)$$

the equality is due to the continuity of $\langle v | \cdot | v \rangle : DM_m \to \mathbb{C}$. The value $\langle v | \rho_n - \rho_k | v \rangle$ is a nonnegative real for almost all $n$, therefore (A.3) itself is a nonnegative real. This proves $\rho_k \sqsubseteq \lim_n \rho_n$. One can similarly prove that $\lim_n \rho_n$ is the least among the upper bounds of $(\rho_n)_{n \in \mathbb{N}}$. □

**Proposition (Prop. 2.13, repeated).** The order $\sqsubseteq$ on $QO_{m,n}$ (Def. 2.12) is an $\omega$-CPO.

**Proof.** Let $(\mathcal{E}_k)_{k \in \mathbb{N}}$ be an increasing chain in $QO_{m,n}$. We define $\mathcal{E}$ to be its “pointwise supremum”: for each $\rho \in DM_m$,

$$\mathcal{E}(\rho) := \sup_{k \to \infty} \mathcal{E}_k(\rho) \overset{(\ast)}{=} \lim_{k \to \infty} \mathcal{E}_k(\rho) \quad (A.4)$$
where the supremum is taken in the $\omega$-CPO $DM_n$ (Lem. 2.5). In the proof of Lem. 2.5 we exhibited that the supremum is indeed the limit ((*) above). We claim that this $E$ is the supremum of the chain $(E_k)_{k \in \mathbb{N}}$.

We check that (A.4) indeed defines a QO $E$. In Def. 2.6, the trace condition follows from the continuity of $\text{tr}(\_): DM_n \to \mathbb{R}$. For convex linearity we have to show

$$\lim_{k \to \infty} \left( (c_k(x))_{m,n}\left(\sum_j p_j \rho_j\right) \right) = \sum_j p_j \left( \lim_{k \to \infty} \left( (c_k(x))_{m,n}(\rho_j) \right) \right).$$

This follows from the linearity of the limit operation $\lim_{k \to \infty}$, which is straightforward since $\lim_{k \to \infty}$ is with respect to the trace norm $\|\_\|_{\text{tr}}$. To prove complete positivity of $E$, one can use Choi’s characterization of complete positive maps (see [7, Thm. 6.5]). The operations involved in the characterization are all continuous, hence one can conclude complete positivity of $E$ from that of $E_k$.

It remains to show that $E$ is indeed the least upper bound. This is obvious since $\subseteq$ on $QO_{m,n}$ is a pointwise extension of $\subseteq$ on density matrices. \qed

**Appendix B. Proofs for §3.4**

**Appendix B.1. Proof of Lem. 3.8**

**Proof.** We let

- the set of evaluation contexts that is defined in Def. 3.7 denoted by $EV$, and
- that which is defined in Lem. 3.8 denoted by $\overline{EV}$.

We are set out to show $EV = \overline{EV}$. We rely on the following facts:

1. if $E, E' \in EV$ then $E[E'] \in EV$; and
2. if $D, D' \in EV$ then $D[D'] \in \overline{EV}$.

The former is proved by induction on the construction of $E'$; the latter is by induction on the construction of $D$.

One direction $EV \subseteq \overline{EV}$ is proved easily by induction. We present only one case. For $E \equiv E'[\_][M] \in EV$, by the induction hypothesis we have $E' \in \overline{EV}$; moreover $\_[M] \in \overline{EV}$. Therefore by the fact 2 above, $E \equiv E'[\_][M]$ belongs to $\overline{EV}$.

The other direction $\overline{EV} \subseteq EV$ is similar; we present only one case. For $D \equiv D'[M] \in \overline{EV}$, by the induction hypothesis we have $D' \in EV$; moreover $\_[M] \in EV$. Therefore by the fact 1 above, $D \equiv D'[M] \equiv [D'][M]$ belongs to $EV$. \qed
Appendix B.2. Proof of Lem. 3.15

PROOF. 1. Reflexivity $A <: A$ is easy by induction on the construction of $A$. Transitivity

\[ A <: A' \quad \text{and} \quad A' <: A'' \quad \implies \quad A <: A'' \]

is shown by induction on the derivation. We present one case; the other cases are similar. Assume that $A <: A'$ is derived by the $(\otimes)$ rule:

\[
\frac{B <: B' \quad C <: C' \quad n = 0 \Rightarrow n' = 0}{A \equiv !n(B \otimes C) <: !n'(B' \otimes C') \equiv A'} \quad (\otimes).
\]

The form $A' \equiv !n'(B' \otimes C')$ requires the relation $A' <: A''$ to be derived also by the $(\otimes)$ rule:

\[
\frac{B' <: B'' \quad C' <: C'' \quad n' = 0 \Rightarrow n'' = 0}{A' \equiv !n'(B' \otimes C') <: !n''(B'' \otimes C'') \equiv A''} \quad (\otimes).
\]

Now we apply the induction hypothesis to $B <: B'$ and $B' <: B''$ in (B.1–B.2), and obtain that $B <: B''$ is derivable. Similarly $C <: C''$ is derivable. That $n = 0 \Rightarrow n'' = 0$ follows immediately from (B.1–B.2), too. Using the $(\otimes)$ rule we derive $A <: A''$.

2. By cases on the rule that derives $A <: B$. We present the case $(-\to)$:

\[
\frac{B_1 <: A_1 \quad A_2 <: B_2 \quad n = 0 \Rightarrow m = 0}{A \equiv !n(A_1 \to A_2) <: !m(B_1 \to B_2) \equiv B} \quad (-\to).
\]

Since $n + 1 = 0 \Rightarrow m + 1 = 0$ is trivially true, using $B_1 <: A_1$ and $A_2 <: B_2$ we derive $!n+1(A_1 \to A_2) <: !m+1(B_1 \to B_2)$. The other cases are similar.

3. By cases on the outermost type constructor in $A$ (ignoring $!$). Assume it is $-\to$, with $A \equiv !^k(B \to C)$. Then we have $B <: B$ and $C <: C$ due to the item 1.; and $n = 0 \Rightarrow m = 0$ implies $n + k = 0 \Rightarrow m + k = 0$. Therefore

\[
\frac{B <: B \quad C <: C \quad n + k = 0 \Rightarrow m + k = 0}{!n+k(B \to C) <: !m+k(B \to C)} \quad (-\to)
\]

derives $!^n A <: !^m A$, as required. The other cases are similar.

4. Straightforward, by cases on the rule that derives $!^n A <: !^m B$.

5. We show existence of directed sups and infs by simultaneous induction, on the complexity of upper/lower bounds.

Assume $A_1 <: n\text{-qbit}$ and $A_2 <: n\text{-qbit}$. Then $A_1 \equiv !^{n_1} n\text{-qbit}$ and $A_2 \equiv !^{n_2} n\text{-qbit}$ for some $n_1, n_2 \in \mathbb{N}$. We define

\[
A_0 \equiv \begin{cases} !n\text{-qbit} & \text{if } n_1 \neq 0 \text{ and } n_2 \neq 0, \\ n\text{-qbit} & \text{otherwise.} \end{cases}
\]

This $A_0$ is clearly a supremum of $A_1$ and $A_2$.  

67
Assume \( n \text{-qbit} \triangleleft A_1 \) and \( n \text{-qbit} \triangleleft A_2 \). Then \( A_1 \) and \( A_2 \) must both be \( n \text{-qbit} \), and \( A_0 \equiv n \text{-qbit} \) is the infimum.

In the cases where the given upper (or lower) bound is \( \uparrow^{n+1} n \text{-qbit} \), \( \top \) or \( !^{n+1} \top \), we can similarly compute a supremum (or a infimum).

Assume \( A_1 \triangleleft B \rightarrow C \) and \( A_2 \triangleleft B \rightarrow C \). Then the rules in (11) force that \( A_1 \equiv !^{n_1} (B_1 \rightarrow C_1) \) and \( A_2 \equiv !^{n_2} (B_2 \rightarrow C_2) \), with \( B \triangleleft B_1 \), \( B \triangleleft B_2 \), \( C_1 \triangleleft C \) and \( C_2 \triangleleft C \). Since the complexity of \( B \) or \( C \) is smaller than that of \( B \rightarrow C \) we can use the induction hypothesis, obtaining \( B_0 \) as a infimum of \( B_1 \), \( B_2 \) and \( C_0 \) as a supremum of \( C_1 \), \( C_2 \). Now

\[
A_0 := \begin{cases} 
\uparrow (B_0 \rightarrow C_0) & \text{if } n_1 \neq 0 \text{ and } n_2 \neq 0, \\
B_0 \rightarrow C_0 & \text{otherwise.} 
\end{cases}
\]

is easily shown to be a supremum of \( A_1 \), \( A_2 \). The other cases are similar. \( \Box \)

**Appendix B.3. Proof of Lem. 3.17**

**Proof.** By induction on the derivation of \( \Delta \vdash M : A \). The proof is mostly straightforward; here we only present one case.

Assume that the derivation of \( \Delta \vdash M : A \) looks as follows, with the \((\neg.I_2)\) rule the one applied last, and \( \Delta = (! \Delta_0, \Gamma) \), \( M \equiv \lambda x^B . N \), \( A \equiv !^n (B' \rightarrow C) \).

\[
\begin{array}{c}
x : B, ! \Delta_0, \Gamma \vdash N : C \\
\text{FV}(N) \subseteq | \Delta_0 | \cup \{x \} \\
B' \triangleleft B
\end{array}
\]

\[
\begin{array}{c}
! \Delta_0, \Gamma \vdash \lambda x^B . N : !^n (B' \rightarrow C)
\end{array}
\]

Since \( \Delta' \triangleleft \Delta = (! \Delta_0, \Gamma) \), we have \( \Delta' = (\Delta_0', \Gamma') \) with \( \Delta_0' \triangleleft ! \Delta_0 \) and \( \Gamma' \triangleleft \Gamma \). Furthermore, by Lem. 3.15.4, \( \Delta_0' \) must be of the form \( \Delta_0'' = ! \Delta_0' \) with some \( \Delta_0'' \).

Thus \( \Delta' = (! \Delta_0'', \Gamma') \). Similarly, from the assumption that \( A \equiv !^m (B'' \rightarrow C'') \), \( A' \) we have \( A' \equiv !^m (B'' \rightarrow C'') \) with \( B'' \triangleleft B', C \triangleleft C'' \) and \( m = 0 \Rightarrow m = 0 \).

Now we have \( (x : B, ! \Delta_0'', \Gamma') \triangleleft (x : B, ! \Delta_0, \Gamma) \). Using the induction hypothesis we obtain \( \vdash x : B, ! \Delta_0'', \Gamma' \vdash N : C'' \). Since \( \text{FV}(N) \subseteq | \Delta_0 | \cup \{x \} = | \Delta_0'' | \cup \{x \} \), the \((\neg.I_2)\) rule can be applied.

\[
\begin{array}{c}
x : B, ! \Delta_0'', \Gamma' \vdash N : C'' \\
\text{FV}(N) \subseteq | \Delta_0'' | \cup \{x \} \\
B'' \triangleleft B
\end{array}
\]

\[
\begin{array}{c}
! \Delta_0'', \Gamma' \vdash \lambda x^B . N : !^m (B'' \rightarrow C'')
\end{array}
\]

To obtain \( B'' \triangleleft B \) we used transitivity (Lem. 3.15.1). This derives \( \Delta' \vdash \lambda x^B . N : A' \). \( \Box \)

**Appendix B.4. Proof of Lem. 3.20**

**Proof.** 1. By induction on the construction of a value \( V \).

If \( V \equiv x \), a variable, then the type judgment must be derived by the \((Ax.1)\) rule. Then the claim follows immediately from Lem. 3.15.4.

If \( V \) is some constant (i.e. \texttt{new,meas}^{\ast+1}, \texttt{cmp}^{\text{m,n}, \text{orqstate}_p} \), or if \( V \equiv \ast \), \( \text{FV}(V) \) is empty.
If \( V \equiv \lambda x^B.M \), the type judgment \( \Delta \vdash V : !A \) must be derived by the \( (\neg \circ, I_2) \) rule:

\[
\frac{x : B, !\Delta_0, \Gamma \vdash M : C \quad \text{FV}(M) \subseteq |\Delta_0| \cup \{x\} \quad B' :<! B}{!\Delta_0, \Gamma \vdash \lambda x^B.M : !^n(B' \leadsto C)} \quad (\neg \circ, I_2)
\]

with \( \Delta = (!\Delta_0, \Gamma) \) and \( A \equiv !^n(B \leadsto C) \). Now we have \( \text{FV}(V) = \text{FV}(M) \setminus \{x\} \subseteq (|\Delta_0| \cup \{x\}) \setminus \{x\} = |\Delta_0|, \) from which the claim follows.

If \( V \equiv (V_1, V_2) \), the type judgment \( \Delta \vdash V : !A \) must be derived as follows:

\[
\frac{!\Delta_0, \Gamma_1 \vdash V_1 : !^n A_1 \\ !\Delta_0, \Gamma_2 \vdash V_2 : !^n A_2}{!\Delta_0, \Gamma_1, \Gamma_2 \vdash (V_1, V_2) : !^n (A_1 \# A_2)} \quad (\mid, !)
\]

with \( \Delta = (!\Delta_0, \Gamma_1, \Gamma_2) \). By the induction hypothesis, we have

\[
(!\Delta_0, \Gamma_1)|_{\text{FV}(V_1)} = !\Delta_1, \quad (!\Delta_0, \Gamma_2)|_{\text{FV}(V_2)} = !\Delta_2
\]

for some \( \Delta_1 \) and \( \Delta_2 \) (here \( (!\Delta_0, \Gamma_1)|_{\text{FV}(V_1)} \) denotes the suitable restriction of a context). The claim follows immediately. The cases where \( V \equiv \inj_B V' \) or \( V \equiv \inj_A^\delta V' \) are similar.

2. By induction on the construction of a value \( V \).

If \( V \equiv x \), a variable, then the claim follows easily from Lem. 3.15.4. The cases where \( V \) is a constant or \( V \equiv * \) are similarly easy.

In case \( V \equiv \lambda x^B.M \); if \( A \) is of the form \( A \equiv !^n(B' \leadsto C) \) with \( n \geq 1 \), the type judgment \( !\Delta, \Gamma \vdash V : A \) is derived by the \((\neg \circ, I_2) \) rule and it also derives \(!\Delta, \Gamma \vdash V : !A \). If \( A \) is of the form \( A \equiv B' \leadsto C \), the derivation of \(!\Delta, \Gamma \vdash V : A \) looks as follows.

\[
\frac{x : B, !\Delta, \Gamma \vdash M : C \quad B' :<! B}{!\Delta, \Gamma \vdash \lambda x^B.M : !^n(B' \leadsto C)} \quad (\neg \circ, I_1)
\]

Now the assumption \( \text{FV}(V) \subseteq |\Delta| \) yields \( \text{FV}(M) \subseteq |\Delta| \cup \{x\} \); this can be used in

\[
\frac{x : B, !\Delta, \Gamma \vdash M : C \quad \text{FV}(M) \subseteq |\Delta| \cup \{x\} \quad B' :<! B}{!\Delta, \Gamma \vdash \lambda x^B.M : !^n(B' \leadsto C)} \quad (\neg \circ, I_2).
\]

Thus we have derived \(!\Delta, \Gamma \vdash V : !A \).

Finally, the cases where \( V \equiv \inj_B^\delta V' \) or \( V \equiv \inj_A^\delta V' \) are easy using the induction hypothesis. This concludes the proof. \( \square \)

Appendix B.5. Proof of Lem. 3.22

PROOF. The first two rules are straightforward, by induction on the derivation of \(!\Delta, \Gamma_2, x : A \vdash N : B \). Here we make essential use of the monotonicity rule (Lem. 3.17) and the weakening rule (Lem. 3.19.3). The third rule (Subst_3) follows from (Subst_2) via Lem. 3.19.2 and 3.20.1. The last rule (Subst_4) for evaluation contexts is proved by induction on the construction of \( E \), where we employ the “bottom-up” definition (Lem. 3.8) in place of Def. 3.7. \( \square \)
Appendix B.6. Proof of Lem. 3.23

Proof. By induction on the construction of an evaluation context $E$. The step cases where $E \not\equiv \dbrack{}$ are easy, since the local character of the typing rules of $\text{Hoq}$ (for a rule to be applied, the terms in the assumptions can be anything).

For the base case (i.e. $E \equiv \dbrack{}$) we prove by cases according to Def. 3.9.

In the case where $M \rightarrow_p N$ is by the ($\rightarrow$) rule of Def. 3.9, we have $p = 1$, $M \equiv (\lambda x^A.M')V$ and $N \equiv M'[V/x]$. By the assumption we have $\vdash \Delta \vdash (\lambda x^A.M')V : A$; inspection of the typing rules shows that its derivation must look like the following.

\[
\begin{array}{rcl}
\vdash \Delta, \Gamma \vdash M' : A & \vdash \Delta, \Gamma \vdash x : A' & \vdash \Delta, \Gamma \vdash f : B \rightarrow A \\
\vdash \Delta, \Gamma, \Gamma_0 \vdash V' : C & \vdash \Delta, \Gamma, \Gamma_0 \vdash x' : C & \vdash \Delta, \Gamma, \Gamma_0 \vdash f' : \lambda x^A.M' : B \rightarrow A \\
\vdash \Delta, \Gamma_0 \vdash \lambda x^A.M' : B \rightarrow A & \vdash \Delta, \Gamma \vdash \text{letrec} f^{B\rightarrow C} x = M' \in N' & \vdash \Delta, \Gamma \vdash x : A' \\
\end{array}
\]

Here $\Delta = (\Delta', \Gamma_1, \Gamma_2)$. We have $C <; B <; A'$, thus $C <; A'$ (Lem. 3.15.1). Using the monotonicity rule (Lem. 3.17) we have $\vdash ! \Delta', \Gamma_2 \vdash V : A'$. Combining this with the top-left judgment in (B.4) via the (Subst$_{\lambda}$) rule in Lem. 3.22, we obtain $\vdash ! \Delta', \Gamma_1, \Gamma_2 \vdash M'[V/x] : A$, which is our goal.

The cases of the ($\square$), ($\top$), and ($+_\gamma$) rules are similar, where we rely on the substitution rules in Lem. 3.22.

We consider the case of the (rec) rule, where $p = 1$, $M \equiv \text{letrec} f^{B\rightarrow C} x = M' \in N'$ and

\[
\begin{align*}
N & \equiv N'[\lambda x^B. \text{letrec} f^{B\rightarrow C} x = M' \in M'] / f \\
& \equiv N'[\lambda z^B. \text{letrec} f^{B\rightarrow C} x = M' \in M'[z/x] / f] .
\end{align*}
\]

Here $z$ is a fresh variable and we used the $\alpha$-equivalence. By the assumption we have $\vdash ! \Delta \vdash \text{letrec} f^{B\rightarrow C} x = M' \in N' : A$; inspection of the typing rules shows that its derivation must look like the following.

\[
\begin{array}{rcl}
\vdash ! \Delta', \Gamma, \Gamma_0 \vdash f : (B \rightarrow C) & \vdash \Delta, \Gamma, \Gamma_0 \vdash x : B \vdash M' : C & \vdash \Delta, \Gamma, \Gamma_0 \vdash \text{letrec} f^{B\rightarrow C} x = M' \in N' : A \\
\vdash ! \Delta', \Gamma \vdash \text{letrec} f^{B\rightarrow C} x = M' \in N' : A & \vdash ! \Delta', \Gamma \vdash \text{letrec} f^{B\rightarrow C} x = M' \in N' : A \\
\end{array}
\]

Here $\Delta = (\Delta', \Gamma)$. Now by $\alpha$-converging $\vdash ! \Delta', f : (B \rightarrow C), x : B \vdash M' : C$—the top-left judgment in (B.5)—we have $\vdash ! \Delta', f : (B \rightarrow C), z : B \vdash M'[z/x] : C$.

Applying the (rec) rule to the last two judgments, we obtain $\vdash ! \Delta', \Gamma \vdash \text{letrec} f^{B\rightarrow C} x = M' \in M'[z/x] : C$. By the ($\rightarrow$) rule this leads to $\vdash ! \Delta', \Gamma \vdash \lambda z^B. \text{letrec} f^{B\rightarrow C} x = M' \in M'[z/x] : (B \rightarrow C)$.

The last judgment is combined with the second assumption in (B.5) via the substitution rule (Subst$_{\lambda}$) in Lem. 3.22, and yields $\vdash ! \Delta', \Gamma \vdash N'[\lambda z^B. \text{letrec} f^{B\rightarrow C} x = M' \in M'[z/x] / f] : A$. This is our goal.

In the other cases, the reduction $M \rightarrow_p N$ is derived by one of the rules in Def. 3.9 that deal with quantum constants (such as $\text{new}$ and $U$). We do only one case, of the rule ($\text{meas}_1$). The other cases are similar.
By inspecting the typing rules it is easy to see that the type $A$ of the term $\text{meas}_i^{n+1}(\text{qstate}_p)$ must be $A \equiv ! \text{bit} \otimes n\text{-qbit}$. Therefore it suffices to show that the term $tt \equiv \text{inj}_i^\top(*)$ can indeed have the type $! \text{bit} \equiv !(\top + \top)$. This is shown as follows.

\[
\frac{\Delta \vdash ! \top \quad (\top,1)}{\Delta \vdash \text{inj}_i^\top(*) : !(\top + \top) \quad (+,I_1)}
\]

This concludes the proof. □

Appendix B.7. Proof of Lem. 3.24

Proof. By induction on the construction of the term $M$. We only present the case where $M \equiv \text{NL}$. If $N$ is not a value, by the induction hypothesis $N$ has a reduction $N \rightarrow_p N'$; this yields $M \equiv \text{NL} \rightarrow_p N'L$. It is similar when $N$ is a value but $L$ is not.

Now assume that both $N$ and $L$ are values. By the assumption that $M \equiv \text{NL}$ is typable, we must have $\vdash N : B \Rightarrow A$ for some $B$. A value $N$ of the type $B \Rightarrow A$ must be either of the following forms: $\lambda x^B. N'$, $\text{new}$, $\text{meas}_i^n$, $U$ or $\text{cmp}$.

If $N \equiv \lambda x^B. N'$, since $L$ is a value we have $M \equiv \text{NL} \rightarrow_1 N'[L/x]$. If $N \equiv \text{new}$, it is easy to see that a closed value $L$ of type $\text{bit}$ must be either $tt$ or $ff$. Therefore the reduction ($\text{new}_1$) or ($\text{new}_2$) in Def. 3.9 is enabled from $M \equiv \text{NL}$. The other cases are similar. □

Appendix C. The Quantum Branching Monad $\mathcal{Q}$

The following characterization is standard. See e.g. [37].

Lemma Appendix C.1. The trace condition (16) holds if and only if: for each $m \in \mathbb{N}$,

\[
\sum_{x \in X} \sum_{n \in \mathbb{N}} M \left( (c(x))_{m,n} \right) \subseteq I_m \quad . \tag{C.1}
\]

Here $\subseteq$ is the Löwner partial order (Def. 2.4); $M((c(x))_{m,n})$ is the matrix from Def. 2.8. Note that $M((c(x))_{m,n})$ is an $m \times m$ matrix regardless of the choice of $n$, hence the sum in (C.1) makes sense.

Proof. We define a matrix $A$ by

\[
A := I_m - \sum_{x \in X} \sum_{n \in \mathbb{N}} M \left( (c(x))_{m,n} \right) \quad . \tag{C.2}
\]
To prove the ‘if’ part, assume that $A$ is positive. We have, for each $\rho \in \text{DM}_m$,
\[
\text{tr}(A\rho) + \sum_{x \in X} \sum_{n \in \mathbb{N}} \text{tr}\left((c(x))_{m,n}(\rho)\right)
\]
\[
= \text{tr}(A\rho) + \sum_{x \in X} \sum_{n \in \mathbb{N}} \text{tr}\left(\sum_{i \in I, x, m, n} E^{(i)}_{x,m,n} \cdot \rho \cdot (E^{(i)}_{x,m,n})^\dagger\right)
\]
where \{E^{(i)}_{x,m,n}\}_{i \in I, x, m, n} is an operator-sum representation of \((c(x))_{m,n}.
\]
\[
= \text{tr}(A\rho) + \sum_{x \in X} \sum_{n \in \mathbb{N}} \text{tr}\left(\sum_{i \in I, x, m, n} (E^{(i)}_{x,m,n})^\dagger \cdot E^{(i)}_{x,m,n} \cdot \rho\right)
\]
\[
= \text{tr}\left((A + \sum_{x \in X} \sum_{n \in \mathbb{N}} M((c(x))_{m,n})) (\rho)\right)
\]
\[
= \text{tr} (\rho) \leq 1 \text{ by (C.2).}
\]

Hence it suffices to show that $\text{tr}(A\rho) \geq 0$. It is a standard fact that any density matrix $\rho \in \text{DM}_m$ can be written as
\[
\sum_{i \in I} \lambda_i |v_i\rangle \langle v_i| ,
\]
with $|v_i\rangle \in \mathbb{C}^m$, $\|v_i\| = 1$, $\lambda_i \geq 0$ and $\sum_i \lambda_i \leq 1$. Therefore it suffices to show that $\text{tr}(A|v\rangle \langle v|) \geq 0$ if $\|v\| = 1$. Now:
\[
\text{tr}(A\rho) = \text{tr}(A|v\rangle \langle v|) \geq \text{tr}(A|v\rangle \langle v|) \geq 0 ,
\]
where $(\ast)$ is because $\text{tr}(BC) = \text{tr}(CB)$ for any $B, C$; and the last inequality holds because $A$ is positive.

For the ‘only if’ part, we must show that the matrix $A$ in (C.2) is positive. For that purpose it suffices to prove: for any $|v\rangle \in \mathbb{C}^m$ with length 1, $\langle v|A|v\rangle \geq 0$.
\[
\langle v|A|v\rangle = \langle v|I_m - \sum_{x, n} \sum_{i} (E^{(i)}_{x,m,n})^\dagger E^{(i)}_{x,m,n} |v\rangle
\]
where \{E^{(i)}_{x,m,n}\}_{i \in I, x, m, n} is an operator-sum representation of \((c(x))_{m,n}.
\]
\[
= \langle v|v\rangle - \sum_{x, n} \sum_{i} \langle v| (E^{(i)}_{x,m,n})^\dagger E^{(i)}_{x,m,n} |v\rangle
\]
\[
= 1 - \sum_{x, n} \sum_{i} \text{tr}\left((E^{(i)}_{x,m,n})^\dagger \cdot (E^{(i)}_{x,m,n})\right)
\]
using $\text{tr}(BC) = \text{tr}(CB)$ and $\langle v|v\rangle = \|v\|^2 = 1$
\[
= 1 - \sum_{x, n} \text{tr}\left((c(x))_{m,n} (|v\rangle \langle v|)\right) \geq 0 \text{ by (16).}
\]

This concludes the proof. \hfill \Box

**Proposition AppendixC.2.** The construction $Q$ in Def. 4.1 is indeed a functor.
Proof. First we verify that the data (\(Qf\))(c) defined in (17) indeed satisfies the trace condition. This is easy by direct calculations. It remains to be shown that: \(Q(id) = id\) and \(Q(g \circ f) = Qg \circ Qf\). These are easy consequences of the facts that \(id^{-1} = id\) and \((g \circ f)^{-1} = f^{-1} \circ g^{-1}\), respectively. \(\square\)

Lemma AppendixC.3. The sum in the definition (19) of \(\mu\) is well-defined.

Proof. First we show that, for fixed \(\gamma \in QX\), \(m \in \mathbb{N}\) and \(\rho \in DM_m\), there are only countably many pairs \((c, k) \in QX \times \mathbb{N}\) such that

\[
(\gamma(c))_{m,k}(\rho) \neq 0,
\]

equivalently (because the matrix is positive), \(\text{tr}( (\gamma(c))_{m,k}(\rho) ) \neq 0\).

To see this, observe that the trace condition (16) for \(\gamma \in QX\) means \(\sum_{c,k} \text{tr}( (\gamma(c))_{m,k}(\rho) ) \leq 1\). It is a standard fact that a discrete distribution with sum \(\leq 1\) has at most a countable support; from this our claim above follows.

Therefore we can enumerate all such pairs as \(((c_l, k_l))_{l \in \mathbb{N}}\). Then (19) amounts to

\[
(\mu_X(\gamma)(x))_{m,n}(\rho) = \sum_{l \in \mathbb{N}} (c_l(x))_{k_l,n} \circ (\gamma(c_l))_{m,k_l}(\rho).
\]

The right-hand side is the limit of a sequence (over \(l \in \mathbb{N}\)) in \(DM_n\) that satisfies the assumption of Lem. AppendixA.3. Thus it is well-defined. \(\square\)

Proposition AppendixC.4. The construction \(Q\) in Def. 4.1 is indeed a monad.

Proof. First we verify that the data \(\eta_X(x)\) in (18) and \(\mu_X(\gamma)\) in (19) satisfy the trace condition (16) and hence belong indeed to \(QX\). For the unit \(\eta_X(x)\) this is obvious. For the multiplication \(\mu_X(\gamma)\) we shall verify (16). For any \(\rho \in DM_m\), we have

\[
\sum_{x \in X} \sum_{n \in \mathbb{N}} \text{tr}( (\mu_X(\gamma))(x))_{m,n}(\rho)
\]

\[
= \sum_{x \in X} \sum_{n \in \mathbb{N}} \sum_{c \in QX} \sum_{k \in \mathbb{N}} \text{tr}( (c(x))_{k,n} \circ (\gamma(c))_{m,k}(\rho) )
\]

\[
= \sum_{x \in X} \sum_{n \in \mathbb{N}} \sum_{c \in QX} \sum_{k \in \mathbb{N}} \text{tr}( (\gamma(c))_{m,k}(\rho) ) \cdot \text{tr}( (c(x))_{k,n} \circ (\gamma(c))_{m,k}(\rho) )
\]

since \((c(x))_{k,n}\) and \(\text{tr}\) are linear

\[
= \sum_{c \in QX} \sum_{k \in \mathbb{N}} \text{tr}( (\gamma(c))_{m,k}(\rho) ) \cdot \left( \sum_{x \in X} \sum_{n \in \mathbb{N}} \text{tr}( (c(x))_{k,n} \circ (\gamma(c))_{m,k}(\rho) ) \right)
\]

\[
\leq \sum_{c \in QX} \sum_{k \in \mathbb{N}} \text{tr}( (\gamma(c))_{m,k}(\rho) ) \cdot 1 \quad \text{by the trace condition for } c \in QX, (\ast)
\]

\[
\leq 1 \quad \text{by the trace condition for } \gamma \in QQX.
\]
Note that in the above (\(\ast\)), the matrix
\[
\frac{(\gamma(c))_{m,k}(\rho)}{\text{tr}\left((\gamma(c))_{m,k}(\rho)\right)}
\]
has its trace 1 hence is a density matrix.

Next we verify that the maps \(\eta_X\) and \(\mu_X\) in (18–19) are natural in \(X\). For \(\eta_X\) it is obvious. For \(\mu_X\), given \(\gamma \in \mathcal{Q} \mathcal{Q} X\) and \(f : X \to Y\):

\[
\left((\mu_Y \circ \mathcal{Q} f)(\gamma)(y)\right)_{m,n} = \sum_{c' \in \mathcal{Q} Y} \sum_{k \in \mathbb{N}} (c'(y))_{k,n} \circ (\mathcal{Q} f)^{-1}(c')_{m,k} \circ (\gamma(c))_{m,k}
\]

since \((c'(y))_{k,n}\) is linear

\[
= \sum_{c' \in \mathcal{Q} Y} \sum_{c \in \mathcal{Q} f} \sum_{k \in \mathbb{N}} (c'(y))_{k,n} \circ (\gamma(c))_{m,k}
\]

This proves the naturality of \(\mu\).

Finally we verify that \(\eta\) and \(\mu\) indeed satisfy the monad laws, that is, that the following diagrams commute.

\[
\begin{align*}
\mathcal{Q}X & \xrightarrow{\eta_X} \mathcal{Q} \mathcal{Q}X \xrightarrow{\mathcal{Q} \eta_X} \mathcal{Q}X & \mathcal{Q} \mathcal{Q}X & \xrightarrow{\mathcal{Q} \mu_X} \mathcal{Q} \mathcal{Q}X & \mathcal{Q} \mathcal{Q}X & \xrightarrow{\mu_X} \mathcal{Q}X
\end{align*}
\]

The leftmost triangle is obvious; for the other triangle, we first observe

\[
\left((\mathcal{Q} \eta_X)(c)(c')\right)_{m,n} = \begin{cases} (c(x'))_{m,n} & \text{if } c' = \eta_X(x') \text{ for some } x' \in X, \\ 0 & \text{otherwise}. \end{cases}
\]
This is used in the following calculation.

\[
\left( (\mu_X \circ (Q\eta_X))(c)(x) \right)_{m,n} = \sum_{c' \in QX} \sum_{k \in N} (c'(x))_{k,n} \circ \left( (Q\eta_X)(c') \right)_{m,k} \\
= \sum_{x \in X} \sum_{k \in N} \left( (\eta_X(x'))(x) \right)_{k,n} \circ (c(x'))_{m,k}
\]

by (C.4)

\[
= I_n \circ (c(x))_{m,n} = (c(x))_{m,n}.
\]

This proves the commutativity of the triangle in the middle of (C.3). For the square on the right, given \(\Gamma \in QQQ\):

\[
\left( (\mu_X \circ Q\mu_X)(\Gamma)(x) \right)_{m,n} = \sum_{c \in QX} \sum_{l \in N} (c(x))_{l,n} \circ \left( Q\mu_X(\Gamma)(c) \right)_{m,l} \\
= \sum_{c \in QX} \sum_{l \in N} \sum_{\gamma \in \mu^{-1}_X(c)} \left( \gamma(c) \right)_{l,n} \circ \left( Q\mu_X(\Gamma)(c) \right)_{m,l} \\
= \sum_{\gamma \in QX} \sum_{k \in N} \sum_{l \in N} \sum_{\gamma \in \mu^{-1}_X(c)} \left( \gamma(c) \right)_{l,n} \circ \left( \gamma(c) \right)_{k,l} \circ (\Gamma)_{m,k}
\]

This concludes the proof. \(\square\)

**Appendix C.1. The Kleisli Category \(KF(Q)\)**

**Lemma (Lem. 4.2, repeated).** Given two successive arrows \(f : X \to Y\) and \(g : Y \to U\) in \(KF(Q)\), their composition \(g \circ f : X \to U\) is concretely given as follows.

\[
\left( (g \circ f)(x)(u) \right)_{m,n} = \sum_{y \in Y} \sum_{k \in N} (g(y))(u)_{k,n} \circ (f(x)(y))_{m,k}.
\]
Proof. Given $x \in X$, $u \in U$ and $\rho \in \text{DM}_m$:

\[
\begin{aligned}
\left( (g \circ f)(x)(u) \right)_{m,n} &= \left( (\mu_U \circ \mathcal{Q}g \circ f)(x)(u) \right)_{m,n} \\
&= \left( \mu_U \left( (\mathcal{Q}g)(f(x)) \right)(u) \right)_{m,n} \\
&= \sum_{c \in \mathcal{Q}U} \sum_{k \in \mathbb{N}} (c(u))_{k,n} \circ \left( \left( (\mathcal{Q}g)(f(x)) \right)(c) \right)_{m,k} \\
&= \sum_{c \in \mathcal{Q}U} \sum_{k \in \mathbb{N}} \left( \sum_{y \in g^{-1}(\{c\})} (f(x)(y))_{m,k} \right) (c(u))_{k,n} \\
&= \sum_{y \in Y} \sum_{k \in \mathbb{N}} (g(y)(u))_{k,n} \circ (f(x)(y))_{m,k}.
\end{aligned}
\]

Here the equality $(\ast)$ holds because, due to $g : Y \rightarrow \mathcal{Q}U$ being a function, we have $Y = \bigsqcup_{c \in \mathcal{Q}U} \{y \mid g(y) = c\}$. □

Note that $\mathcal{K}(\mathcal{Q})$ has finite coproducts, carried over from $\text{Sets}$ by the Kleisli inclusion functor.

**Theorem AppendixC.5.** The monad $\mathcal{Q}$ on $\text{Sets}$ satisfies the following conditions (from [33, Requirements 4.7]); and therefore by [33, Prop. 4.8], the category $\mathcal{K}(\mathcal{Q})$ is partially additive.

1. $\mathcal{K}(\mathcal{Q})$ is $\omega$-CPO enriched.
2. $\mathcal{K}(\mathcal{Q})$ has monotone cotupling.
3. For each $X, Y \in \mathcal{K}(\mathcal{Q})$, the least element $\bot_{X,Y} \in \mathcal{K}(\mathcal{Q})(X, Y)$ in the homset is preserved by both pre- and post-composition: that is, $f \circ \bot = \bot$ and $\bot \circ g = \bot$.

We note that, under this condition, there exist “projection” maps $p_j : \coprod_{i \in I} X_i \rightarrow X_j$ such that

\[
p_j \circ \kappa_i = \begin{cases} 
\text{id} & \text{if } i = j, \\
\bot & \text{otherwise,}
\end{cases}
\]

where $\kappa_j : X_j \rightarrow \coprod_{i \in I} X_i$ denotes a coprojection.
4. The “bicartesian” maps

\[
\text{bc}_{(X_i)_{i \in I}} := \left( \mathcal{Q} \left( \prod_{i \in I} X_i \right) \xrightarrow{(p^*_i)_{i \in I}} \prod_{i \in I} \mathcal{Q}X_i \right) \quad \text{where} \quad p^*_i := \mu \circ Tp_i
\]

76
form a cartesian natural transformation with monic components. This means that all the naturality squares

\[
\begin{align*}
T(\prod_i X_i) & \xrightarrow{bc} \prod_i TX_i \\
T(\prod_i f_i) & \xrightarrow{bc} \prod_i Tf_i \\
T(\prod_i Y_i) & \xrightarrow{bc} \prod_i TY_i
\end{align*}
\]

are pullback diagrams in \textbf{Sets}, for each \( f_i : X_i \to Y_i \) in \textbf{Sets}.

The original condition [33, Requirements 4.7] is stated in terms of DCPOs instead of \(\omega\)-CPOs. This difference is not important.

\textbf{Proof.} We use the pointwise extension of the order \(\sqsubseteq\) in Def. 4.4 in homsets \(\mathcal{K}\ell(Q)(X,Y)\). It is an \(\omega\)-CPO due to Prop. 2.13. It is easy to see that the bottoms are preserved by pre- and post-composition. To see that suprema are preserved too, one uses the following facts.

- A QO is continuous, since its operator-sum representation is.
- The fact at the beginning of the proof of Lem. AppendixC.3 (that the support of each of the relevant functions is at most countable).
- The limit operator \(\lim_{k \to \infty}\) (for increasing chains) and the countable sum operator \(\sum_{l \in \mathbb{N}}\) are interchangeable: \(\lim_k \sum_l \rho_{k,l} = \sum_l \lim_k \rho_{k,l}\).

Cotupling is monotone since the order in the homsets are pointwise.

To see \(bc\) is monic, assume \(bc(c) = bc(d)\). Then \(p^c_i(c) = p^d_i(d)\) for each \(i \in I\). It is easy to see that \(p^c_i(c) = c \circ \kappa^c_i\), therefore

\[
c = [c \circ \kappa^c_i]_i = [d \circ \kappa^d_i]_i = d .
\]

It is straightforward to see that the naturality squares are pullbacks. \(\square\)

\textbf{Appendix D. Proofs for \S 5.4}

\textbf{Appendix D.1. Proof of Lem. 5.21}

\textbf{Proof.} The set \(A_Q = \mathcal{K}\ell(Q)(\mathbb{N},\mathbb{N})\) is an \(\omega\)-CPO due to the \(\omega\)-\textbf{CPO} enriched structure of the category \(\mathcal{K}\ell(Q)\) (see Thm. 4.5). Therefore the order \(\sqsubseteq\) on \(A_Q\) is essentially the L"owner partial order (Def. 2.4).

To show the item 1, we use the fact that composition \(\odot\) of arrows and the trace operator \(\text{tr}\) are both continuous in the Kleisli category \(\mathcal{K}\ell(Q)\). Indeed, the former is part of the fact that \(\mathcal{K}\ell(Q)\) is \(\omega\)-\textbf{CPO} enriched (Thm. AppendixC.5). The proof for the latter is not hard either, exploiting the explicit presentation of \(\text{tr}\) by Girard’s execution formula (see [52, Chap. 3]). In the proof the following Fubini-like result is essential: if \((x_{n,m})_{n,m \in \mathbb{N}}\) is increasing both in \(n\) and \(m\), then \(\sup_n \sup_m x_{n,m} = \sup_n x_{n,n}\). The item 1 then follows immediately from the definitions of \(\cdot\) and \(!\) in (23).

The item 2 is proved using the presentation of \(\text{tr}\) by Girard’s execution formula and the fact that composition \(\odot\) in \(\mathcal{K}\ell(Q)\) is (left and right) strict. \(\square\)
Appendix D.2. Proof of Lem. 5.23

Proof. For inductiveness of $U \times V$, assume $x_0 \subseteq x_1 \subseteq \ldots \subseteq x'_i \subseteq \ldots$ and $(x_i, x'_i) \in U \times V$ for each $i$. By the definition of $U \times V$, we find $k_i, l_i, u_i, k'_i, l'_i, u'_i$ such that $x_i = \hat{P}k_i(\hat{P}l_i u_i)$, $x'_i = \hat{P}k'_i(\hat{P}l'_i u'_i)$, $(k_i, u_i, k'_i, u'_i) \in U$ and $(l_i, u_i, l'_i, u'_i) \in V$. Since $k_i = \hat{P}x_i$, by continuity of $\cdot$ we have that $(k_i)$ is an increasing chain. So are $(l_i), (u_i)$, $(k'_i), (l'_i)$, $(u'_i)$; therefore $(k_i, u_i, l_i, u_i), (k'_i, u'_i, l'_i, u'_i)$, are increasing, too. By the admissibility of $U$ and $V$ we have

$$(\sup_i k_i u_i, \sup_i k'_i u'_i) \in U \quad \text{and} \quad (\sup_i l_i u_i, \sup_i l'_i u'_i) \in V.$$  

Again by continuity of $\cdot$ we have

$$\sup_i x_i = \hat{P}(\sup_i k_i)(\hat{P}(\sup_i l_i)(\sup_i u_i)) \quad \text{and} \quad \sup_i x'_i = \hat{P}(\sup_i k'_i)(\hat{P}(\sup_i l'_i)(\sup_i u'_i)) ;$$

since $\sup_i k_i u_i = (\sup_i k_i)(\sup_i u_i)$ (and so on) we conclude that $(\sup_i x_i, \sup_i x'_i) \in U \times V$.

Strictness is the reason we use $\times$ instead of $\times$. We have $\hat{P} \bot (\hat{P} \bot \bot) = \bot$: this is because $\hat{P} x y = j \circ (x + y) \circ k$ (see (31)) and that $\circ$ is (left and right) strict. This shows $(\bot, \bot) \in U \times V$.

Inductiveness of $X \rightarrow U$ is easily shown by similar arguments. Finally, strictness of $X \rightarrow U$ is because for each $(x, x') \in X$, $(\bot x, \bot x') = (\bot, \bot) \in U$. Here the left strictness of $\cdot$ is crucial. \qed

Appendix D.3. Proof of Lem. 5.24

Proof. The PER $[\text{0-qbit}]$ (Def. 5.13) is admissible. Indeed, $(\bot, \bot) = (Q_0, Q_0) \in [\text{0-qbit}]$ and an increasing chain in $[\text{0-qbit}]$ is precisely an increasing chain in $[0, 1]$. Therefore Lem. 5.23 shows that the functor $F_{\text{pbt}} = \text{[bit]} \rightarrow ([\text{0-qbit}] \times \_)$ preserves admissibility. Since $\text{Bit} = \{(\bot, \bot)\}$ is admissible, each object $F_{\text{pbt}} \text{Bit}$ in the final sequence is admissible.

We prove strictness of $R$. By Def. 5.6 we have $\bot = (\bot, \_), \bot = \bot$ by (31). Thus it suffices to show that for any $j$ and any $i$ such that $j \leq i$, $(c_{i,j}, \bot, \bot) \in F_{\text{pbt}} \text{Bit}. \quad \text{This is by cases: we distinguish } j = 0 \text{ and } j > 0. \quad \text{If } j = 0, (c_{i,j}) : F_{\text{pbt}} \text{Bit} \rightarrow \text{Bit} \text{ is the unique map to } \text{Bit} = \{(\bot, \bot)\} \text{ and hence } c_{i,j} \bot = \bot. \quad \text{Therefore } (c_{i,j}, \bot, \bot) \in F_{\text{pbt}} \text{Bit} \text{ for any } i.

Assume $j > 0$. We first note the functor $[\text{bit}] \rightarrow \_ \cdot$’s action on arrows: it carries

$$[c] : X \rightarrow Y \quad \text{to} \quad [\lambda b. c(\text{th})] : [\text{bit}] \rightarrow X \rightarrow [\text{bit}] \rightarrow Y . \quad \text{(D.1)}$$

The functor $[\text{0-qbit}] \times \_$ carries $[c] : X \rightarrow Y$ to

$$[\lambda v.(\lambda k_1 w. w(\lambda k_2 u. \hat{P}k_1(\hat{P}(\lambda z. c(k_2 z))u)))] : [\text{0-qbit}] \times X \rightarrow [\text{0-qbit}] \times Y ; \quad \text{(D.2)}$$

after suitable insertion of the conversion combinators $C_{\text{pbt}}$ and $C_{\text{pbt}}$, it describes the functor $[\text{0-qbit}] \times \_$’s action on arrows.
Our aim now is to show
\[(c_{i,j} \perp, \perp) \in F_{\text{pbt}}^j Bt = \llbracket \text{bit} \rrbracket \twoheadrightarrow (\llbracket \text{0-qbit} \rrbracket \times F_{\text{pbt}}^{j-1} Bt) ;\]
by (24) it suffices to show
\[(c_{i,j} \perp b, \perp b') \in \llbracket \text{0-qbit} \rrbracket \times F_{\text{pbt}}^{j-1} Bt \quad \text{for each } (b, b') \in \llbracket \text{bit} \rrbracket. \quad \text{(D.3)}\]
By (D.1) we have
\[(c_{i,j}, \lambda b. d_{i-1,j-1}(b)) \in F_{\text{pbt}}^i Bt \twoheadrightarrow F_{\text{pbt}}^j Bt ,\]
where \(d_{i-1,j-1}\) is the realizer of the arrow
\[
[\text{0-qbit}] \times [c_{i-1,j-1}] : [\text{0-qbit}] \times F_{\text{pbt}}^{i-1} Bt \twoheadrightarrow [\text{0-qbit}] \times F_{\text{pbt}}^{j-1} Bt
\]
described as in (D.2). Therefore
\[(c_{i,j} \perp b, d_{i-1,j-1}(\perp b)) \in [\text{0-qbit}] \times F_{\text{pbt}}^{j-1} Bt . \quad \text{(D.4)}\]
Now using that \(\cdot\) is left strict,
\[d_{i-1,j-1}(\perp b) = d_{i-1,j-1} \perp = \perp = \perp b' ,\]
where for the second equality we also used the concrete description (D.2) of \(d_{i-1,j-1}\). Therefore (D.4) proves (D.3).

Inductiveness of \(R\) is proved much like the proof of Lem. 5.23. □

Appendix E. Well-Definedness of Interpretation of Well-Typed Terms

Towards our goal of proving Lem. 5.32, we introduce another set of typing rules, and we call them the principal typing rules. The system is a restriction of the Hoq typing rules (Table 1).

Definition Appendix E.1 (Principal typing in Hoq). The principal typing rules of Hoq are in Table E.2.

In the rules, a square-bracketed entry like \([x : A]\) in a context means that it can be absent. The contexts in the (+.E)P rule are complicated: they form the following partition of the free variables of the term in the conclusion. Notice that \(\Gamma'\) need not be of the form \(\Gamma''\).
\[ \frac{\Delta \vdash x : A}{\Delta \vdash \lambda x. A : A \rightarrow B} \quad \Delta \neq \Delta' \text{ for any } \Delta' \quad (-\circ.1) \]
\[ \frac{\Delta \vdash x : A, \Delta : B}{\Delta \vdash \lambda x.A : !A \rightarrow B} \quad \Delta \equiv \Delta' \text{ for some } \Delta' \quad (-\circ.2) \]

\[ \frac{!\Delta, \Gamma_1 \vdash M : !^m(A \rightarrow B) \quad !\Delta, \Gamma_2 \vdash N : C \quad C <: A}{!\Delta, \Gamma_1, \Gamma_2 \vdash MN : B} \quad (\rightarrow) \]

\[ \frac{\Gamma \vdash x : !^m A_1 \quad \Gamma \vdash \Gamma_2 : !^m A_2 \quad (m = 0 \Leftrightarrow n = 0) \quad (m = 1 \Leftrightarrow n \geq 0) \quad \text{At least one of } A_1 \text{ and } A_2 \text{ is not of the form } !B}{!\Delta, \Gamma_1, \Gamma_2 \vdash \text{let } (x_1 \vdash A_1, x_2 \vdash A_2) = M \text{ in } N : A} \quad (\exists.1) \]

\[ \frac{\Delta \vdash * : !^m \top}{!\Delta, \Gamma_1 \vdash M : !^m \top} \quad !\Delta, \Gamma_2 \vdash N : A \quad (\top.E) \]

\[ \frac{\Delta \vdash M : !^m A_1 \quad (m = 0 \Leftrightarrow n = 0) \quad (m = 1 \Leftrightarrow n \geq 0) \quad A_1 \text{ is not of the form } !B}{\Delta \vdash \text{inj}^{A_1}_x : !^m(A_1 + !^m A_2)} \quad (+.I_1) \]

\[ \frac{\Delta \vdash N : !^m A_2 \quad (m = 0 \Leftrightarrow n = 0) \quad (m = 1 \Leftrightarrow n \geq 0) \quad A_2 \text{ is not of the form } !B}{\Delta \vdash \text{inj}^{A_2}_y : !^m(A_1 + !^m A_2)} \quad (+.I_2) \]

\[ \frac{!\Delta, !\Delta_1, !\Delta_2 \vdash P : !^m(C_1 + C_2) \quad m = 0 \Rightarrow n = 0}{!\Delta, !\Delta_1, !\Delta_2, \Gamma', \Gamma_1, \Gamma_2 \vdash \text{match } P \text{ with } (x_1, x_2) : M_1 \quad x_1 \vdash !A_1 \quad x_2 \vdash !A_2 \quad x_1 \notin [\Gamma, \Delta_2, \Gamma_2], x_2 \notin [\Gamma, \Delta_1, \Gamma_1] \quad (+.E)} \]

\[ \frac{!\Delta \vdash [f : !(A \rightarrow B)], [x : A] : M : B' \quad !\Delta, \Gamma \vdash N : C \quad B'' <: B}{!\Delta, \Gamma \vdash \text{letrec } f^{A \rightarrow B} = M \text{ in } N : C} \quad (\text{rec}) \]

---

Table E.2: Principal typing rules for Hoq
We shall write \( \Pi \vdash_P \Delta \vdash M : A \) if a derivation tree \( \Pi \), according to these rules, derives the type judgment. We write \( \vdash_P \Delta \vdash M : A \) if there exists such \( \Pi \), that is, the type judgment is derivable.

**Lemma AppendixE.2.** \( \vdash_P \Delta \vdash M : A \) implies \( \vdash \Delta \vdash M : A \).

**Proof.** By induction on the principal type derivation of \( \vdash_P \Delta \vdash M : A \). We only present some cases.

When the last rule applied is \((-\circ,E)\)P, that is

\[
\frac{\Delta', \Gamma_1 \vdash M' : !^m (A' \rightarrow A) \quad \Delta', \Gamma_2 \vdash N' : C \quad C <: A'}{\Delta', \Gamma_1, \Gamma_2 \vdash M'N' : A} (-\circ,E)P
\]

with \( \Delta = (\Delta', \Gamma_1, \Gamma_2) \) and \( M \equiv M'N' \), by the induction hypothesis we have

\( \vdash ! \Delta', \Gamma_1 \vdash M' : !^m (A' \rightarrow A) \) and \( \vdash ! \Delta', \Gamma_2 \vdash N' : C' \).

Applying Cor. 3.18 to the former yields \( \vdash ! \Delta', \Gamma_1 \vdash M' : A' \rightarrow A \). Then, together with \( C <: A' \), we can use the \((-\circ,E)\) rule (of the original type system) to derive \( ! \Delta', \Gamma_1, \Gamma_2 \vdash M'N' : A \). The case \((\otimes,I)\)P is similar using Cor. 3.18.

For the case \((\otimes,E)\)P we additionally use Lem. 3.17 to show that

\( \vdash ! \Delta, \Gamma_2, x_1 : !^n A_1, x_2 : !^n A_2 \vdash N : A \) implies \( \vdash ! \Delta, \Gamma_2, x_1 : !^n C_1, x_2 : !^n C_2 \vdash N : A \).

The case \((+E)\)P and \((\text{rec})\)P are similar. \( \square \)

**Lemma AppendixE.3.**

1. \( \vdash_P \Delta \vdash M : A \) implies \( |\Delta| = \text{FV}(M) \).

2. Principal typing is unique in the following sense:

   \( \vdash_P \Delta \vdash M : A \) and \( \vdash_P \Delta' \vdash M : A' \) imply \( A \equiv A' \).

3. Derivation in principal typing is unique: if \( \Pi \vdash_P \Delta \vdash M : A \) and \( \Pi' \vdash_P \Delta \vdash M : A \), then \( \Pi \equiv \Pi' \).

**Proof.** 1. Straightforward by induction.

2. By induction on the construction of \( M \). We only present one case; the other cases are similar.

   Assume \( M \) is of the form \( \text{let} \ (x_1^n A_1, x_2^n A_2) = M' \in N \). Then its principal type derivation must end with the \((\otimes,E)\)P rule, as below.

\[
\frac{\Delta, \Gamma_1 \vdash M' : !^m (C_1 \otimes C_2) \quad m = 0 \Rightarrow n = 0 \quad \Delta, \Gamma_2, [x_1 : !^n A_1], [x_2 : !^n A_2] \vdash N : A \quad C_1 <: A_1 \quad C_2 <: A_2}{\Delta, \Gamma_1, \Gamma_2 \vdash \text{let} \ (x_1^n A_1, x_2^n A_2) = M' \in N : A} (\otimes,E)P
\]

The context \( \Delta, \Gamma_2, [x_1 : !^n A_1], [x_2 : !^n A_2] \) is determined by the given context \( \Delta, \Gamma_1, \Gamma_2 \) in particular we can read off the types \( !^n A_i \) of the variables \( x_i \) from the explicit type labels in \( M \). Therefore by the induction hypothesis, the principal type \( A \) of \( N \) is determined; hence so is the principal type of \( M \), too.

3. Straightforward by induction on the construction of a term \( M \). In many cases (including \( M \equiv NL \), where the rule \((-\circ,E)\)P is involved) the items 1–2 play an essential role. \( \square \)
Definition AppendixE.4 (Interpretation of principal type judgments).
For each derivation $\Pi \vdash \Delta \vdash M : A$ by the rules in Table E.2, we assign an arrow

$$[[\Pi]]^\gg : [[\Delta]] \longrightarrow T[[A]]$$

in the way that is a straightforward adaptation of Def. 5.31.

Lemma AppendixE.5. Assume $\Pi \vdash \Delta \vdash M : A$ (in the original rules in Table 1). Then there exist a type $A^o$ and a derivation $\Pi^o$ (in the principal typing rules in Table E.2) such that:

1. $\Pi^o \vdash \Delta|_{\text{FV}(M)} \vdash M : A^o$,
2. $A^o \prec A$, and
3. the following diagram commutes.

$$\begin{array}{c}
\Delta \\
\downarrow \text{weak} \\
\Delta|_{\text{FV}(M)} \\
\downarrow \Pi \\
\Pi^o \\
\downarrow \text{weak} \\
\Pi^o|_{\text{FV}(M)} \\
\downarrow \text{E.2} \\
\Pi^o|_{\Pi} \\
\downarrow T[A] \\
\Pi^o|_{A^o \prec A} \\
\downarrow T[A^o < A] \\
\end{array}$$

PROOF. The diagram in (E.2) can be refined into the following one; we shall prove that the triangle therein commutes.

$$\begin{array}{c}
\Delta \\
\downarrow \text{weak} \\
\Delta|_{\text{FV}(M)} \\
\downarrow \Pi \\
\Pi^o \\
\downarrow \text{weak} \\
\Pi^o|_{\text{FV}(M)} \\
\downarrow \text{E.2} \\
\Pi^o|_{\Pi} \\
\downarrow T[A] \\
\Pi^o|_{A^o \prec A} \\
\downarrow T[A^o < A] \\
\end{array}$$

Here $[[\Pi]]_{\text{FV}}$ is as in Def. 5.31. The proof is by induction on $\Pi$. We present one case; the others are similar.

Assume $\Pi$ is in the following form, with the last rule applied being $(\neg \circ \text{E})$.

$$\Pi \equiv \left[ \begin{array}{c} ! \Delta, \Gamma_1 \vdash M : A \rightarrow \circ B \\
\Pi_1 \\
! \Delta, \Gamma_2 \vdash N : C \\
\Pi_2 \\
\end{array} \right] \quad (\neg \circ \text{E})$$

By the induction hypothesis, there exist types $D, E$ and derivations $\Pi_1^o, \Pi_2^o$ such that

$$\Pi_1^o \vdash ! \Delta, \Gamma_1 \vdash M : D , \quad \Pi_2^o \vdash ! \Delta, \Gamma_2 \vdash N : E ;$$

$$D < : A \rightarrow \circ B , \quad E < : C ;$$

$$[[\Pi_1]]_{\Pi} = T[D < : A \rightarrow \circ B] \circ [[\Pi_2]]^\gg , \quad \text{and}$$

$$[[\Pi_2]]_{\Pi} = T[E < : C] \circ [[\Pi_2]]^\gg .$$

Since $D < : A \rightarrow \circ B$, the type $D$ must be of the form $D \equiv !^m(A' \rightarrow \circ B')$ with $A < : A'$ and $B' < : B$. Now consider the following derivation $\Pi^o$.

$$\Pi^o \equiv \left[ \begin{array}{c} ! \Delta, \Gamma_1 \vdash M : !^m(A' \rightarrow \circ B') \\
\Pi_1^o \\
! \Delta, \Gamma_2 \vdash N : E \\
\Pi_2^o \\
\end{array} \right] \quad (\neg \circ \text{E})$$

$$\Pi^o|_{\Pi} = T[A' \rightarrow \circ B'] \circ [[\Pi_2]]^\gg , \quad \text{and}$$

$$[[\Pi_2]]_{\Pi} = T[B' < : B] \circ [[\Pi_2]]^\gg .$$

82
Here the side condition $E <: A'$ holds since $E <: C <: A : A'$. Thus we obtain
$\Pi^\circ \vdash \Gamma_1, \Gamma_2 \vdash MN : B'$ with $B' <: B$.

It remains to show that $[[\Pi]]_{FV} = (T[B' <: B]) \circ [[\Pi^\circ]]^\circ$. This is, however, an immediate consequence of

- the induction hypothesis,
- $[[A <: C]] = [[B <: C]] \circ [[A <: B]]$ (Lem. 5.28),
- bifunctoriality of $\otimes$ and $\to$,
- and naturality of $\str, \str', \ev$ and $\mu$.

This can be checked by straightforward diagram chasing.

We are ready to prove Lem. 5.32.

**Proof.** (Of Lem. 5.32) Assume $\Pi \vdash \Delta \vdash M : A$ and $\Pi' \vdash \Delta \vdash M : A$. We apply Lem. AppendixE.5 to obtain $\Pi^\circ, \Pi^\circ, (A')^\circ, (\Pi')^\circ$ such that

$$
\Pi^\circ \vdash [\Pi^\circ]_{FV(M)} \vdash M : A^\circ,
\Pi' = T[A^\circ <: A] \circ [\Pi^\circ]^\circ \text{ weak and } [\Pi'] = T[(A')^\circ <: A] \circ [(\Pi')^\circ]^\circ \text{ weak}.
$$

Applying Lem. AppendixE.3 to the first line we have $A^\circ \equiv (A')^\circ$, and moreover $\Pi^\circ \equiv (\Pi')^\circ$. This is used in the second line to conclude $[[\Pi]] = [[\Pi']]$.

**AppendixF. Proofs for §6**

**Lemma AppendixF.1.** Let $M$ be a term such that $\Gamma \vdash M : A$ is derivable. Assume a subtype relation $A <: B$. Then we have

$$
T[A <: B] \circ [[\Gamma \vdash M : A]] = [[\Gamma \vdash M : B]] .
$$

**Proof.** By induction on the term $M$. We only present two cases. When $M \equiv x$, the composition $T[A <: B] \circ [[\Gamma \vdash M : A]]$ is equal to

$$
[[\Gamma]] \overset{\text{weak}}{\longrightarrow} [[A']] \overset{[A'] \vdash [A]}{\longrightarrow} \overset{[[A] < [B]]}{\longrightarrow} \overset{[[B]]}{\longrightarrow} \overset{[[x]]}{\longrightarrow} T[[B]]. \tag{F.1}
$$

Since $[[A <: B] \circ [[A' <: A]]$ is equal to $[[A' <: B]$ by Lem. 5.28, the arrow (F.1) is equal to $[[\Gamma \vdash x : B]]$.

When $M \equiv M_1 M_2$, the term environment $\Gamma$ is of the form $! \Delta, \Gamma_1, \Gamma_2$, and there is a type $C$ such that $! \Delta, \Gamma_1 \vdash M_1 : C \to A$ and $! \Delta, \Gamma_2 \vdash M_2 : C$ are derivable. The interpretation $[[\Gamma \vdash M_1 M_2 : A]]$ is

$$
\mu_{[A]} \circ \mu_{[A]} \circ TT\text{ev}[C, \tau_{[A]}] \circ T\text{str}[\tau_{[C]} \to \tau_{[A]}, [C]] \circ \text{str}_{\tau_{[C]} \to \tau_{[A]}, \tau_{[C]}} \circ \mu_{[\Delta, \Gamma_1 \vdash M_1 : C \to A]} \overset{\otimes}{\longrightarrow} [[! \Delta, \Gamma_2 \vdash M_2 : C]] .
$$

83
where \(c : \llbracket \Delta, \Gamma_1, \Gamma_2 \rrbracket \rightarrow \llbracket \Delta, \Gamma_1 \rrbracket \boxtimes \llbracket \Delta, \Gamma_2 \rrbracket\) is a suitable permutation followed by contractions. By naturality of \(\mu, \text{str}, \text{str}^t\) and \(\text{ev}\), the composition \(\Gamma \vdash A <: B\) \(\circ [\Gamma \vdash M_1 M_2 : A]\) is equal to

\[
\mu_B [\circ \mu_T [\circ T \text{Teu}_C, T_B \circ T \text{str}_C \rightarrow \text{r}[B], [C] \circ \text{str}^t_C \rightarrow \text{r}[B], T_C] \circ (\llbracket \Delta, \Gamma_1 \vdash M_1 : C \rightarrow A \rrbracket \boxtimes [\llbracket \Delta, \Gamma_2 \vdash M_2 : C \rrbracket] \circ \varepsilon),
\]

where the arrow

\[
T(\llbracket C \rrbracket \rightarrow T[A <: B]) : T(\llbracket C \rrbracket \rightarrow T[A]) \rightarrow T(\llbracket C \rrbracket \rightarrow T[B])
\]

is obtained by applying suitable functors to the arrow \([A <: B] : [A] \rightarrow [B]\).

Since \(T(\llbracket C \rrbracket \rightarrow T[A <: B]) \circ [\llbracket \Delta, \Gamma_1 \vdash M_1 : C \rightarrow A \rrbracket] = \llbracket \llbracket \Delta, \Gamma_1 \vdash M_1 : C \rightarrow B \rrbracket\), we see that \(T[A <: B] \circ [\Gamma \vdash M_1 M_2 : A]\) is equal to \([\Gamma \vdash M_1 M_2 : B]\).

We can prove the other cases in the same way. \(\square\)

As is usual with the categorical interpretation of call-by-value languages in Kleisli categories, the interpretation of a value \(\Gamma \vdash V : A\) in Hoq factorizes through the monad unit \(\eta^T\) and is the form \(\eta^T_{[A]} \circ f\) for some arrow \(f : [\Gamma] \rightarrow [A]\) (see Def. 5.31). We write \([\Gamma \vdash V : A]_{ev}\) for the arrow such that \([\Gamma \vdash V : A]\) = \(\eta^T_{[A]} \circ [\Gamma \vdash V : A]_{ev}\) given in Def. 5.31. It is easy to see that \([\Gamma \vdash V : A]_{ev}\) is of the form \(f \circ \varphi'\) for some \(f : 1 \rightarrow [A]\). Here \(\varphi' : 1 \xrightarrow{\cong} !1\) is from Thm. 4.13.

**Lemma AppendixF.2.** Assume that \(\Gamma, x : A \vdash M : B\) and \(\Gamma \vdash V : A\) are derivable for a term \(M\) and a closed value \(V\). Then the composition \(\Gamma, x : A \vdash M : B\) \(\circ (\text{id}_{[\Gamma]} \boxtimes [\Gamma \vdash V : A])\) is equal to \([\Gamma \vdash M[V/x] : B]\).

In Lem. AppendixF.2, we assume that \(x\) is the largest in \([\Gamma] \cup \{x\}\) with respect to the linear order \(\prec\) in Def. 5.30. It is straightforward to generalize the statement to an arbitrary variable \(x\) in a term context.

**Proof.** By induction on the term \(M\). We only treat the case when \(M \equiv x\). When \(M \equiv x\), the composition \(\Gamma, x : A \vdash x : B\) \(\circ (\text{id}_{[\Gamma]} \boxtimes [\Gamma \vdash V : A])\) is equal to \(\eta^T_{[B]} \circ [A <: B] \circ [\Gamma \vdash V : A]_{ev}\), which is equal to \([\Gamma \vdash V : B]\) by Lem. AppendixF.1.

When \(M \equiv M_1 M_2\), the term environment \(\Gamma, x : A\) is of the form \(!\Delta, \Gamma_1, \Gamma_2\) and there is a type \(C\) such that \(!\Delta, \Gamma_1 \vdash M_1 : C \rightarrow B\) and \(!\Delta, \Gamma_2 \vdash M_2 : C\) are derivable. The interpretation \(\llbracket \Gamma, x : A \vdash M_1 M_2 : B \rrbracket\) is

\[
\mu_B [\circ \mu_T [\circ T \text{Teu}_C, T_B \circ T \text{str}_C \rightarrow \text{r}[B], [C] \circ \text{str}^t_C \rightarrow \text{r}[B], T_C] \circ (\llbracket \Delta, \Gamma_1 \vdash M_1 : C \rightarrow B \rrbracket \boxtimes \llbracket \Delta, \Gamma_2 \vdash M_2 : C \rrbracket) \circ c
\]

where \(c : \llbracket \Delta, \Gamma_1, \Gamma_2 \rrbracket \rightarrow \llbracket \Delta, \Gamma_1 \rrbracket \boxtimes \llbracket \Delta, \Gamma_2 \rrbracket\) is a suitable permutation followed by contractions. By the induction hypothesis and naturality of the permutation

84
and contractions, the composition \([\Gamma, x : A \vdash M_1 M_2 : B] \circ (\text{id}_V) \otimes [[V : A]]_v\) is equal to

\[
\begin{align*}
\mu_B & \circ \mu_T[B] \circ TTev[c, \tau[B]] \circ TStr[c] \circ \tau[B], [c] \circ \text{str}_c[c] \circ \tau[B], \tau[c] \\
& \circ ([! \Delta', \Gamma' \vdash M_1[V/x] : C \rightarrow B] \otimes [[! \Delta', \Gamma' \vdash M_2[V/x] : C]]) \circ c'
\end{align*}
\]

where \(! \Delta', \Gamma_1', \Gamma_2'\) is the term environment obtained by removing \(x : A\) from \(! \Delta, \Gamma_1, \Gamma_2\) and \(c' : [! \Delta', \Gamma_1', \Gamma_2'] \rightarrow [! \Delta', \Gamma_1'] \otimes [[\Delta, \Gamma_2']\) is a suitable permutation followed by contractions. Hence, \([\Gamma, x : A \vdash M_1 M_2 : B] \circ (\text{id}_V) \otimes [[V : A]]_v\) is equal to \([\Gamma \vdash (M_1 M_2)[V/x] : B]\).

We can prove the other cases in the same way. \(\square\)

**Lemma AppendixF.3.** If \(\vdash E[M] : A\) is derivable, then there exist a type \(B\) such that \(\vdash M : B\) and \(x : B \vdash E[x] : A\).

**Proof.** By induction on the evaluation context \(E\). \(\square\)

**Lemma AppendixF.4** (Lem. 6.1, repeated). Let \(E\) be an evaluation context, and \(x\) be a variable that does not occur in \(E\). Assume that \(x : A \vdash E[x] : B\) is derivable. Then for any term \(M\) such that \(\vdash \Gamma \vdash M : A\), the interpretation \([\Gamma \vdash E[M] : B] : [\Gamma] \rightarrow [T[B]]\) is calculated by

\[
[\Gamma \vdash E[M] : B] = \mu_B \circ T[x : A \vdash E[x] : B] \circ [\Gamma \vdash M : A].
\]

**Proof.** By induction on the evaluation context \(E\), where we use the characterization in Lem. 3.8. We only present two cases. When \(E \equiv []\), since \(x : A \vdash E[x] : B\) is derivable, we must have \(A \prec B\). By Lem. AppendixF.1, \([\Gamma \vdash E[M] : B]\) is equal to \([T[A \prec B] \circ [\Gamma \vdash M : A]]\), which is nothing but \([\mu_B \circ T[x : A \vdash E[x] : B] \circ [\Gamma \vdash M : A]]\).

When \(E \equiv E'N\), there exists a type \(C\) such that \(x : A \vdash E'[x] : C \rightarrow B\) and \(N : C\) are derivable. The interpretation \([\Gamma \vdash (E'[M])[N] : B]\) is given, by Def. 5.31, by

\[
\mu_B \circ \mu_T[B] \circ TTev[c, \tau[B]] \circ TStr[c] \circ \tau[B], [c] \circ \text{str}_c[c] \circ \tau[B], \tau[c] \\
\circ ([\Gamma \vdash E'[M] : C \rightarrow B] \otimes [[N : C]]). \quad (F.2)
\]

By the induction hypothesis we have

\[
[\Gamma \vdash E'[M] : C \rightarrow B] = \mu_C \circ [T[x : A \vdash E'[x] : C \rightarrow B] \circ [\Gamma \vdash M : A]] .
\]

Using this we see that \((F.2)\) is equal to

\[
\mu_B \circ T[f \circ [\Gamma \vdash M : A]], \quad \text{where}
\]

\[
f := \left[ \mu_B \circ \mu_T[B] \circ TTev[c, \tau[B]] \circ TStr[c] \circ \tau[B], [c] \circ \text{str}_c[c] \circ \tau[B], \tau[c] \circ \left( [x : A \vdash E'[x] : C \rightarrow B] \otimes [[N : C]] \right) \right] : [[A]] \rightarrow [T[B]] .
\]

Here we notice that \(f\) coincides with the interpretation of \(x : A \vdash E'[x]N : B\). This concludes the case when \(E \equiv E'N\).

We can prove the other cases in the same way. \(\square\)
Lemma AppendixF.5 (Lem. 6.2, repeated). For a closed term $M$ such that $\vdash N : A$, if there is a reduction $M \rightarrow_1 N$ that is not due to a measurement rule $(\text{meas}_1 \text{meas}_4)$ in Def. 3.9, then

$$\llbracket \vdash M : A \rrbracket = \llbracket \vdash N : A \rrbracket .$$

Note that $\vdash N : A$ is derivable by Lem. 3.23.

Proof. Assume $M \equiv E[(\lambda x : C')V]$ and $N \equiv E[L[V/x]]$. By Lem. AppendixF.3, there exists a type $B$ such that $y : B \vdash E[y] : A$ and $\llbracket (\lambda x : C')V : B \rrbracket$ are derivable. By Lem. 3.23, we also have $\vdash L[V/x] : B$. Since

$$\llbracket \vdash (\lambda x : C')V : B \rrbracket \quad \text{and} \quad \llbracket \vdash L[V/x] : B \rrbracket \quad \text{by Lem. 6.1, it is enough to show that} \quad \llbracket \vdash L[V/x] : B \rrbracket$$

by unfolding the definition of the interpretation (Def. 3.31), we obtain

$$\quad \llbracket \vdash (\lambda x : C')V : B \rrbracket = [x : A \vdash L : B] \circ \llbracket \vdash V : A \rrbracket_v ;$$

this coincides with $\llbracket \vdash L[V/x] : B \rrbracket$ by Lem. AppendixF.2.

For the other reduction rules, we can prove the statement in the same way.

Lemma AppendixF.6 (Lem. 6.4 (1), repeated). If $(t, V)$ is in $R_A$, then $(\eta^T_A) \circ t, V)$ is in $R_A^\top$.

Proof. For $(k, E) \in R_A$, since $\mu_{[bt]} \circ T k \circ \eta^T_A \circ t = k \circ t$, we have

$$\mu_{[bt]} \circ T k \circ \eta^T_A \circ t < E[V] .$$

Therefore $(\eta^T_A) \circ t, V)$ is in $R_A^\top$.

Lemma AppendixF.7 (Lem. 6.4 (2), repeated). If $(t, V) \in R_A$ and $A :<: A'$, then $(\llbracket A :<: A' \rrbracket \circ t, V) \in R_A$.

Proof. By induction on $A$. When $A$ is $\top$ or $n\text{-qbit}$, the type $A'$ is equal to $A$, and the statement is straightforward.

When $A$ is $B \sqsupset C$, by the definition of subtyping relation, $A'$ must be of the form $B' \sqsupset C'$ for some $B' :> B$ and $C' :> C$. For $(t \sqsupset s, (V, W)) \in R_{B \sqsupset C}$, the composition $[B \sqsupset C :<: B' \sqsupset C'] \circ (t \sqsupset s)$ is equal to $(\llbracket B :<: B' \rrbracket \circ t) \sqsupset (\llbracket C :<: C' \rrbracket \circ s)$. By the induction hypothesis, $(\llbracket B :<: B' \rrbracket \circ t, V) \in R_{B'}$, and $(\llbracket C :<: C' \rrbracket \circ s, W) \in R_{C'}$, and therefore $(\llbracket B \sqsupset C \llbracket :<: B' \sqsupset C' \rrbracket \circ (t \sqsupset s), (V, W))$ is in $R_{B \sqsupset C'}$. We can similarly show the statement for $A \equiv B + C$.

When $A$ is $B \rightarrow C$, the type $A'$ is of the form $B' \rightarrow C'$ for some $B' :<: B$ and $C :<: C'$. For $(t, V) \in R_{B \rightarrow C}$ and $(s, W) \in R_{B'}$, by naturality of $\text{ev}$ we have

$$\quad \text{ev}_{[B'], [C'] \circ t} \circ ((\llbracket B :<: C \llbracket :<: B' \rightarrow C' \rrbracket \circ t) \sqsupset s)$$

$$\quad = T[C :<: C'] \circ \text{ev}_{[B'] \circ [C]} \circ (t \sqsupset (\llbracket B' :<: B \rrbracket \circ s)). \quad \text{(F.3)}$$

86
By the induction hypothesis, \( ([B' <: B] \circ s, W) \) is in \( R_B \). Since \((t, V)\) is in \( R_{B \rightarrow C} \), we see that \( \langle \text{ev}_{[B'][C]} \circ (t \# ([B' <: B] \circ s)), V W \rangle \) is in \( R_C^{\top} \). By the induction hypothesis, \( (k \circ [C <: C'], E) \) is easily seen to be in \( R_C^{\top} \) for any \((k, E) \in R_C^{\top} \). Therefore, \[
\mu_{\text{bit}} \circ Tk \circ T[C <: C'] \circ \text{ev}_{[B'][C]} \circ (t \# ([B' <: B] \circ s)) \ll V W \]
for any \((k, E) \in R_C^{\top} \). By the definition of \( R_C^{\top} \) and (F.3), we obtain \[
\left( \text{ev}_{[B'][C]} \circ ([B \rightarrow C <: B' \rightarrow C'] \circ t) \# s \right) \ll V W \ \in R_C^{\top} \]
for any \((s, W) \in R_B \), which implies that \( ([B \rightarrow C <: B' \rightarrow C'] \circ t, V) \) is in \( R_{B' \rightarrow C'} \).

When \( A \) is \( B \), the type \( A' \) is of the form \( !^n B' \) for some \( n \geq 0 \) and \( B <: B' \). If \((!t \circ \#', V)\) is in \( R_A \), then \([A <: A'] \circ !t \circ \#' = !^n([B <: B'] \circ t) \circ !^{n-1} \#' \circ \cdots \circ ! \#' \circ \#' \). By the induction hypothesis, \( ([B <: B'] \circ t, V) \) is in \( R_{B'} \), and therefore by (44), \([A <: A'] \circ !t \circ \#', V) \) is in \( R_{A'} \).

**Lemma AppendixF.8.** Let \( M \rightarrow_N \) be a reduction that is not due to a measurement rule (\( (\text{meas}_1 - \text{meas}_4) \) in Def. 3.9). Then \( (t, M) \in R_A^{\top} \iff (t, N) \in R_A^{\top} \).

**Proof.** Let \((k, E)\) be an element in \( R_A^{\top} \). Assume \((t, M) \in R_A^{\top} \); then we have \( \mu_{\text{bit}} \circ Tk \circ t \ll E[M] \). By the definition of big-step semantics, \( E[M] \upharpoonright (p, q) \) if and only if \( E[N] \upharpoonright (p, q) \). Therefore, \( \mu_{\text{bit}} \circ Tk \circ t \ll E[N] \). The other direction is similar.

**Lemma AppendixF.9.** For a type \( A \) such that \( ![\text{bit} \rightarrow \text{qbit}] <: A \), the pair \( ([\ll \text{new} : A], \text{new}) \) is in \( R_A^{\top} \).

**Proof.** When \( A = \text{bit} \rightarrow \text{qbit} \), since \( ([\text{qstate}_{\rho}]_{\text{const}}, \text{qstate}_{\rho}) \) is in \( R_{\text{qbit}} \) for any \( \rho \in \text{DM}_2 \), both \( ([\ll \text{qstate}_{\rho}](0)(0) : \text{qbit}, \text{new}) \) and \( ([\ll \text{qstate}_{\rho}](1) : \text{qbit}, \text{new}) \) are in \( R_{\text{qbit}}^{\top} \), by Lem. AppendixF.6 and Lem. AppendixF.8. Therefore, \( ([\ll \text{new}]_{\text{const}}, \text{new}) \) is in \( R_{\text{bit} \rightarrow \text{qbit}} \), and by Lem. AppendixF.6, \( ([\ll \text{new} : A], \text{new}) \) is in \( R_{\text{bit} \rightarrow \text{qbit}}^{\top} \). When \( A = !(\text{bit} \rightarrow \text{qbit}) \), we have \( \eta[A] \circ ![\ll \text{new}]_{\text{const}} \circ \#' = ![\ll \text{new} : A] \). Since \( ![\ll \text{new}]_{\text{const}}, \text{new} \) is in \( R_{\text{bit} \rightarrow \text{qbit}} \), the pair \( ([\ll \text{new}]_{\text{const}}, \text{new}) \) is in \( R_{(\text{bit} \rightarrow \text{qbit})^{\top}}^{\top} \). Therefore, by Lem. AppendixF.6, \( ([\ll \text{new} : A], \text{new}) \) is in \( R_A^{\top} \). When \( A \) satisfies \( !(\text{bit} \rightarrow \text{qbit}) <: A \), the statement follows from Lem. AppendixF.7, Lem. AppendixF.6 and that \( ![\ll \text{new}]_{\text{const}} \circ \#', \text{new} \) is in \( R_{(\text{bit} \rightarrow \text{qbit})^{\top}}^{\top} \).

**Lemma AppendixF.10.** For a type \( A \) such that \( !(n \cdot \text{qbit} \rightarrow n \cdot \text{qbit}) <: A \), the pair \( ([\ll \text{cmp}_n : A], \text{cmp}_{n, n}) \) is in \( R_A^{\top} \).

**Proof.** Similar to the proof of Lem. AppendixF.9.

**Lemma AppendixF.11.** For a type \( A \) such that \( !(n \cdot \text{qbit} \# m \cdot \text{qbit} \rightarrow (n + m) \cdot \text{qbit}) <: A \), the pair \( ([\ll \text{cmp}_{n, m} : A], \text{cmp}_{n, m}) \) is in \( R_A^{\top} \).
Lemma AppendixF.12. For a type \( A \) such that \(!((n + 1)\text{-qbit}) \rightarrow !\text{bit }\otimes n\text{-qbit} \) \(<:\ A\), the pair \(([\text{meas}^{n+1}_i : A],\text{meas}^{n+1}_i)\) is in \( R_A^{n \otimes \text{qbit}} \).

Proof. First we shall prove that, when \( A \equiv (n + 1)\text{-qbit} \rightarrow !\text{bit }\otimes n\text{-qbit}, \) we have \(((\text{meas}^{n+1}_i)_{\text{const}},\text{meas}^{n+1}_i)\) is in \( R_A \). It is easy to show that for any \( \rho \in DM_{2^{n+1}}, \) the pair \(([\text{meas}^{n+1}_i \text{qstate}_\rho : !\text{bit }\otimes n\text{-qbit}],\text{meas}^{n+1}_i \text{qstate}_\rho)\) is in \( R_{[\text{bit }\otimes \text{qbit}]}^{n+1}. \) Let \((k, E)\) be an element in \( R^{[\text{bit }\otimes \text{qbit}]}_{n+1}. \) We define \( k_0, k_1 : [n\text{-qbit}] \rightarrow T[[\text{bit}]] \) to be the following arrows.

\[
k_0 = k \circ ((! \kappa_\ell \circ \varphi') \otimes \text{id}_{[n\text{-qbit}]}), \quad k_1 = k \circ ((! \kappa_\ell \circ \varphi') \otimes \text{id}_{[n\text{-qbit}]}).
\]

Then \( \text{prob}(\text{tree}(\mu_{[\text{bit}]} \circ T k \circ [\text{meas}^{n+1}_i \text{qstate}_\rho : !\text{bit }\otimes n\text{-qbit}]))) \) is equal to \((0, 0) + \text{prob}(\text{tree}(k_0 \circ [\text{qstate}_{(0, \varphi(0))}]_{\text{const}}) + \text{prob}(\text{tree}(k_1 \circ [\text{qstate}_{(1, \varphi(1))}]_{\text{const}})) ; \)

which is seen much like in the proof of Thm. 6.3. Since

\[
k_0 \circ [\text{qstate}_{(0, \varphi(0))}]_{\text{const}} \leq E([tt, qstate_{(0, \varphi(0))}]) \quad \text{and} \quad k_1 \circ [\text{qstate}_{(1, \varphi(1))}]_{\text{const}} \leq E([ff, qstate_{(1, \varphi(1))}] ;
\]

it follows that \( \mu_{[\text{bit}]} \circ T k \circ [\text{meas}^{n+1}_i \text{qstate}_\rho : !\text{bit }\otimes n\text{-qbit}] \leq E[\text{meas}^{n+1}_i \text{qstate}_\rho], \) and therefore, \(([\text{meas}^{n+1}_i \text{qstate}_\rho : !\text{bit }\otimes n\text{-qbit}],\text{meas}^{n+1}_i \text{qstate}_\rho)\) is in \( R^{[\text{bit }\otimes \text{qbit}]}_{n+1}. \)

When \( A \equiv !(n + 1)\text{-qbit} \rightarrow !\text{bit }\otimes n\text{-qbit}, \) since \(([\text{meas}^{n+1}_i]_{\text{const}},\text{meas}^{n+1}_i)\) is in \( R_{(n + 1)\text{-qbit} \rightarrow [\text{bit }\otimes n\text{-qbit}]} \), the pair \(((\text{meas}^{n+1}_i)_{\text{const}} \circ \varphi',\text{meas}^{n+1}_i)\) is in \( R_A \).

Therefore, by Lem. AppendixF.6, \(([\text{meas}^{n+1}_i : A],\text{meas}^{n+1}_i)\) is in \( R_{[\text{qbit}]}^{n \otimes \text{qbit}} \).

Finally, when \( A \) satisfies \(!((n + 1)\text{-qbit}) \rightarrow !\text{bit }\otimes n\text{-qbit} \) \(<:\ A\), the statement follows from Lem. AppendixF.7, Lem. AppendixF.6 and that \(((\text{meas}^{n+1}_i)_{\text{const}} \circ \varphi',\text{meas}^{n+1}_i)\) is in \( R_{(n + 1)\text{-qbit} \rightarrow [\text{bit }\otimes n\text{-qbit}]} \).

Lemma AppendixF.13. For a type \( A \) such that \(!\text{qbit} \rightarrow !\text{bit} \) \(<:\ A\), the pair \(([\text{meas} : A],\text{meas})\) is in \( R_A^{n \otimes \text{qbit}} \).

Proof. Similar to the proof of Lem. AppendixF.12.

Lemma AppendixF.14 (Lem. 6.5(1), repeated). For any type \( A \) and \( M \in \text{ClTerm}(A), \) we have \(([\bot],M) \in R_A^{T^{+}} \).

Proof. Let \((k, E) \in R_A^{T^{+}} \). We claim

\[
\mu_{[\text{bit}]} \circ T k \circ [\bot] = [\bot],
\]

where \([\bot]\) denotes the arrow \( I \rightarrow T[A] \) in \( \text{PER}_Q \) that is realized by \( \perp \in A_Q \) (cf. Lem. 5.24, 5.21.2). Indeed, by the definition of \( T \), it is straightforward to see that the arrow \( \mu_{[\text{bit}]} \circ T k : T[[\text{bit}]] \rightarrow T[[\text{bit}]] \) is realized by

\[
\lambda vy. v(\lambda a. c_k a y).
\]
where \( c_k \) is a choice of a realizer of \( k \). Therefore the left-hand side of (F.4) is

\[
[\lambda x. \lambda y. \bot (\lambda a. c_k ay)]
\]

that is nothing but \([\bot]\) due to the left strictness of application of \( A_Q \).

From (F.4) it easily follows that \( \text{prob}(\text{tree}(\mu_{\llbracket \text{bit} \rrbracket} \circ Tk \circ [\bot])) = (0,0) \). Therefore we always have \( \mu_{\llbracket \text{bit} \rrbracket} \circ Tk \circ [\bot] \ll E[M] \). \( \square \)

**Lemma AppendixF.15 (Lem. 6.5 (2))**. For any type \( A \) and \( M \in \text{ClTerm}(A) \), if there exists a sequence of realizers \( a_1 \sqsubseteq a_2 \sqsubseteq \cdots \) of arrows in \( \text{PER}_Q(I, T[\llbracket A \rrbracket]) \) such that \( ([a_n], M) \in R_A^\top \), then we have \( ([\bigvee_{n \geq 1} a_n], M) \in R_A^\top \).

**Proof**. Let \((k, E)\) be an element in \( R_A^\top \). Since the application of \( A_Q \) is continuous, the value assigned to any edge of the tree

\[
\text{tree}(\mu_{\llbracket \text{bit} \rrbracket} \circ Tk \circ [\bigvee_{n \geq 1} a_n])
\]

is the least upper bound of the value on the corresponding edge of the trees

\[
\text{tree}(\mu_{\llbracket \text{bit} \rrbracket} \circ Tk \circ [a_n]).
\]

Therefore, if we have \( \mu_{\llbracket \text{bit} \rrbracket} \circ Tk \circ [a_n] \ll E[M] \) for every \( n \geq 1 \), then it follows that \( \mu_{\llbracket \text{bit} \rrbracket} \circ Tk \circ [\bigvee_{n \geq 1} a_n] \ll E[M] \). \( \square \)


URL http://www.sciencedirect.com/science/article/B6V0K-3XV2JD6-2/2/2826c60de2a7d05eb7e804434ce2d376


