Semantics of Higher-Order Quantum Computation via Geometry of Interaction☆

Ichiro Hasuo[a,1,*], Naohiko Hoshino[b,2]

[a] *Department of Computer Science, The University of Tokyo, Hongo 7-3-1, Tokyo 113-8656, Japan*
[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

☆An earlier version of this paper [1] has been presented at the Twenty-Sixth Annual IEEE Symposium on Logic in Computer Science (LICS 2011), 2124 June 2011, Toronto, Ontario, Canada.

*Corresponding author
Email addresses: `ichiro@is.s.u-tokyo.ac.jp` (Ichiro Hasuo), `naophiko@kurims.kyoto-u.ac.jp` (Naohiko Hoshino)
URL: `http://www-mmm.is.s.u-tokyo.ac.jp/~ichiro` (Ichiro Hasuo), `http://www.kurims.kyoto-u.ac.jp/~naophiko` (Naohiko Hoshino)

[1] Supported by Grants-in-Aid No. 24680001, 15K11984 & 15KT0012, JSPS; by the JSPS-INRIA Bilateral Joint Research Project "CRECOGI"; and by Aihara Innovative Mathematical Modelling Project, FIRST Program, JSPS/CSTP.
[2] Supported by Grants-in-Aid No. 26730004 & 15K11984, JSPS; and by the JSPS-INRIA Bilateral Joint Research Project "CRECOGI".

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 counter-intuitive 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) λ-calculus with quantum stores.

Another important family—that is most strongly oriented towards mathematical semantics—is those of *quantum λ-calculi* that are very often based on *linear λ-calculus*. While λ-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 λ-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 λ-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 λ-calculus that is essentially a linear λ-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 **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 **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 λ-calculus is presented. It relies on Selinger's category **Q** in [7]—it can be thought of as an extension of **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 λ-calculus. Instead of starting from **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,[3] 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 λ-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 (Figure 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*

[3] "Classical" as opposed to "quantum"; not as the opposite of "intuitionistic".

(GoI) [20], 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.

Monad B for branching

$\downarrow$ *Coalgebraic trace semantics* [29, 33]
Take the Kleisli category $\mathcal{K}\ell(B)$

Traced monoidal category $\mathbb{C}$
(+ other constructs $\rightarrow$ *GoI situation* [21])

$\downarrow$ *Categorical GoI* [21]
Take $\mathbb{C}(U, U)$

Linear combinatory algebra A

$\downarrow$ *Realizability*, e.g. [34, 32, 35]
Take $\mathbf{PER}_A$

Linear category

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* $\mathcal{Q}$ on **Sets**; this is our choice for the monad B in Figure 1. The resulting linear category $\mathbf{PER}_{\mathcal{Q}}$ is described, too. In §5 we interpret Hoq in this category; finally in §6 we prove adequacy of the denotational model.

Some details and proofs are deferred to appendices. They are found in [36].

2. Preliminaries

We denote the syntactic equality by $\equiv$.

2.1. Quantum Computation

We follow Kraus' formulation [37] of quantum mechanics, which is by now standard and is used in e.g. [38, 7]. For proofs and more detailed explanation, our principal reference is the standard textbook [38, Chap. 2 & Chap. 8].

Notation 2.1. $\mathcal{I}_m$ 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 [38, §2.4] for more details; for our developments later, it is crucial that we allow density matrices ρ such that the trace $\mathsf{tr}(\rho)$ 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 the norm $\| |v\rangle \| = 1$ 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*:

$$|\Phi^+\rangle \;=\; \frac{1}{\sqrt{2}}\big(\,|00\rangle + |11\rangle\,\big) \;=\; \frac{1}{\sqrt{2}}\big(\,|0_1 0_2\rangle + |1_1 1_2\rangle\,\big) \;\; . \tag{1}$$

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 $|0_1\rangle$ and $|1_1\rangle$. The outcome is $|0_1\rangle$ or $|1_1\rangle$ with the same probabilities $1/2$; in each case the state vector gets *reduced* and becomes $|0_1 0_2\rangle$ or $|1_1 1_2\rangle$, respectively. In other words, the result of the measurement is a probability distribution

$$\big[\,|0_1 0_2\rangle \mapsto \frac{1}{2} \;\; , \quad |1_1 1_2\rangle \mapsto \frac{1}{2}\,\big]$$

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

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

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

$$|\Phi^+\rangle\langle\Phi^+| \;=\; \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \quad ; \tag{2}$$

the ensemble $\big[\,|0_1 0_2\rangle \mapsto \frac{1}{2} \;,\; |1_1 1_2\rangle \mapsto \frac{1}{2}\,\big]$ that results from the measurement above is represented by

$$\frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad . \tag{3}$$

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 [38, Theorem 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 $\mathsf{tr}(\rho) \in [0,1]$. Here $[0,1]$ denotes the unit interval. The set of all m-dimensional density matrices is denoted by 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 $\big[\,|v_i\rangle \mapsto p_i\,\big]$ 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. One can also recall, as a related phenomenon in program semantics, that the semantics of possibly *diverging* probabilistic program is given by a *sub*distribution where the probabilities can add up to less than 1. The missing probability is then that for divergence.

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 $|0_1 0_2 \ldots 0_N\rangle, |0_1 0_2 \ldots 1_N\rangle, \ldots, |1_1 1_2 \ldots 1_N\rangle$). 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 [38, 7].

Definition 2.4 (Löwner partial order). The order $\sqsubseteq$ on the set 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

- ρ arises from an ensemble $\big[\,|v_i\rangle \mapsto p_i\,\big]_{i\in I}$;
- σ arises from an ensemble $\big[\,|v_i\rangle \mapsto q_i\,\big]_{i\in I}$; and
- $p_i \le q_i$ for each $i \in I$.

That is, when, in comparing ρ and σ thought of as ensembles, ρ has some components missing.

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

Lemma 2.5. *The relation $\sqsubseteq$ in Definition 2.4 is indeed a partial order. Moreover it is an ω-CPO: any increasing ω-chain $\rho_0 \sqsubseteq \rho_1 \sqsubseteq \cdots$ in DM_m has the least upper bound.* □

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 [38, 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 $\mathcal{E} : \mathrm{DM}_m \to \mathrm{DM}_n$ subject to the following axioms.

1. (Trace condition)
$$\frac{\mathsf{tr}\big(\mathcal{E}(\rho)\big)}{\mathsf{tr}(\rho)} \in [0,1] \quad \text{for any } \rho \in \mathrm{DM}_m \text{ such that } \mathsf{tr}(\rho) > 0.$$
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:
$$\mathcal{E}\big(\sum_{i\in I} p_i\rho_i\big) \;=\; \sum_{i\in I} p_i\mathcal{E}(\rho_i) \;\;.$$
Here I is a possibly infinite index set—we can assume that I is at most countable since a discrete probability subdistribution $(p_i)_{i\in I}$ necessarily has a countable support. From this and that the trace of ρ_i is bounded by 1, it easily follows that the infinite sums on both sides are well-defined.
3. (Complete positivity) An arbitrary "extension" of $\mathcal{E}$ of the form $\mathrm{id}_k \otimes \mathcal{E} : M_k \otimes M_m \to M_k \otimes M_n$ carries a positive matrix to a positive one. (Here $\mathrm{id}_k : M_k \to M_k$ is the identity function.) In particular, so does $\mathcal{E}$ itself.

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

The definition slightly differs from the one in [38, §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 [38, §8.2] for further details.

Proposition 2.7 (Operator-sum representation)**.** *A mapping $\mathcal{E} : \mathrm{DM}_m \to \mathrm{DM}_n$ is a QO if and only if it can be represented in the form*
$$\mathcal{E}(\rho) = \sum_{i\in I} E^{(i)}\rho(E^{(i)})^\dagger \;\;, \tag{4}$$

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

$$\sum_{i \in I} \big(E^{(i)}\big)^{\dagger} E^{(i)} \;\sqsubseteq\; \mathcal{I}_m \enspace .$$

Here the order $\sqsubseteq$ refers to the one in Definition 2.4.

Proof. See [38, §8.2.4]. $\square$

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

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

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 [38, Theorem 8.2]: given two operator-sum representations

$$\mathcal{E} = \textstyle\sum_i E^{(i)}(\_)(E^{(i)})^{\dagger} = \sum_j F^{(j)}(\_)(F^{(j)})^{\dagger} \enspace ,$$

there exists a unitary matrix $U = (u_{i,j})_{i,j}$ such that $E^{(i)} = \sum_j u_{i,j} F^{(j)}$. We have

$$\begin{aligned} \textstyle\sum_i (E^{(i)})^{\dagger} E^{(i)} &= \textstyle\sum_i \big(\sum_j u^*_{i,j} (F^{(j)})^{\dagger}\big)\big(\sum_k u_{i,k} F^{(k)}\big) \\ &= \textstyle\sum_{j,k} \big(\sum_i u^*_{i,j} u_{i,k}\big)\, (F^{(j)})^{\dagger} F^{(k)} \\ &= \textstyle\sum_j \, (F^{(j)})^{\dagger} F^{(j)} \enspace , \end{aligned}$$

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

The following property is immediate.

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

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 ρ corresponds to a QO

$$U(\_)U^{\dagger} \;:\; \rho \;\longmapsto\; U \rho U^{\dagger} \enspace .$$

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

$$U|v\rangle\langle v|U^\dagger = U|v\rangle\big(\,|v\rangle\,\big)^\dagger U^\dagger = U|v\rangle\big(\,U|v\rangle\,\big)^\dagger \ ,$$

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

$$\langle 0_1|\,\_\,|0_1\rangle + \langle 1_1|\,\_\,|1_1\rangle \ , \tag{5}$$

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

$$|0_1\rangle = |0\rangle \otimes \mathcal{I}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \ .$$

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.

$$\begin{aligned} &\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} |\Phi^+\rangle\langle\Phi^+| \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} |\Phi^+\rangle\langle\Phi^+| \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \ = \ \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| \ . \end{aligned}$$

This density matrix represents the ensemble $\big[\,|0\rangle \mapsto \frac{1}{2}, \, |1\rangle \mapsto \frac{1}{2}\,\big]$, or $\big[\,|0_2\rangle \mapsto \frac{1}{2}, \, |1_2\rangle \mapsto \frac{1}{2}\,\big]$ 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

$$\frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{that corresponds to the ensemble } \big[\,|0_1 0_2\rangle \mapsto \frac{1}{2}, \, |1_1 1_2\rangle \mapsto \frac{1}{2}\,\big].$$

Our choice above is because of the type $\mathtt{qbit} \multimap \mathtt{bit}$ (rather than $\mathtt{qbit} \multimap \mathtt{bit} \otimes \mathtt{qbit}$) 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

$$\langle 0_1|\_|0_1\rangle \quad \text{and} \quad \langle 1_1|\_|1_1\rangle \ . \tag{6}$$

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 ρ, the probability for observing $|0_1\rangle$ or $|1_1\rangle$ can then be calculated as

$$\mathsf{tr}\big(\langle 0_1|\rho|0_1\rangle\big) \quad \text{or} \quad \mathsf{tr}\big(\langle 1_1|\rho|1_1\rangle\big) \ , \tag{7}$$

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

$$\mathsf{tr}\big(\langle 0_1|\rho|0_1\rangle\big) \ = \ \mathsf{tr}\big(\langle 0_1|v\rangle\big(\langle 0_1|v\rangle\big)^{\dagger}\big) \ = \ \big\|\langle 0_1|v\rangle\big\|^2 \ .$$

See [38, §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 Definition 2.4 in a pointwise manner to obtain an order between QOs. This is done also in [7].

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

Proposition 2.13. *The order $\sqsubseteq$ on $\mathrm{QO}_{m,n}$ is an ω-CPO.*

PROOF. See Appendix A; also [7, Lemma 6.4]. □

For illustration, notice that for any density matrix ρ,

$$\langle 0_1|\rho|0_1\rangle \ \sqsubseteq \ \langle 0_1|\rho|0_1\rangle + \langle 1_1|\rho|1_1\rangle \quad \text{and} \quad \langle 1_1|\rho|1_1\rangle \ \sqsubseteq \ \langle 0_1|\rho|0_1\rangle + \langle 1_1|\rho|1_1\rangle \ ,$$

in the setting of (6). This establishes

$$\langle 0_1|\_|0_1\rangle \ \sqsubseteq \ \langle 0_1|\_|0_1\rangle + \langle 1_1|\_|1_1\rangle \quad \text{and} \quad \langle 1_1|\_|1_1\rangle \ \sqsubseteq \ \langle 0_1|\_|0_1\rangle + \langle 1_1|\_|1_1\rangle$$

where $\sqsubseteq$ is the order of Definition 2.12. This example is prototypical of our use of the Löwner partial order (Definition 2.4 & 2.12): $\mathcal{E} \sqsubseteq \mathcal{F}$ means that $\mathcal{E}$ is a projection (or "partial measurement") that is a "component" of $\mathcal{F}$.

2.2. Monads for Branching

The notion of *monad* is standard in category theory. In computer science, after Moggi [39], 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 $\mathcal{Q}$—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. [40, 41] for introduction); in particular from the use of a monad B on **Sets** for modeling *branching*, e.g. in [29].

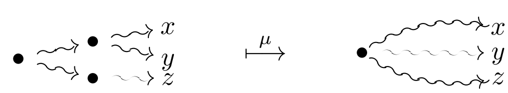
Example 2.14. We list some examples of such "branching monads" B.

- 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*
$$\mathcal{D}X = \bigl\{c : X \to [0,1] \bigm| \sum_x c(x) \leq 1\bigr\}$$
models probabilistic branching. Its unit $\eta^{\mathcal{D}} : X \to \mathcal{D}X$ carries $x \in X$ to the so-called *Dirac distribution* $[x \mapsto 1]$; and its multiplication $\mu^{\mathcal{D}} : \mathcal{D}(\mathcal{D}X) \to \mathcal{D}X$ "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 μ collapses "branching twice" into "branching once," abstracting the internal branching structure. For example, $\mathcal{P}$'s multiplication
$$\mu^{\mathcal{P}}_X : \mathcal{P}\mathcal{P}X \longrightarrow \mathcal{P}X \ , \qquad \text{like} \quad \bigl\{\{x,y\},\{z\}\bigr\} \longmapsto \{x,y,z\}$$
can be understood as follows.



For $\mathcal{D}$, its multiplication
$$\mu^{\mathcal{D}}_X : \mathcal{D}\mathcal{D}X \longrightarrow \mathcal{D}X \ , \qquad \text{like} \quad \begin{bmatrix} \begin{bmatrix} x \mapsto 1/2 \\ y \mapsto 1/2 \end{bmatrix} & \mapsto & 1/3 \\ [z \mapsto 1] & \mapsto & 2/3 \end{bmatrix} \overset{\mu}{\longmapsto} \begin{bmatrix} x \mapsto 1/6 \\ y \mapsto 1/6 \\ z \mapsto 2/3 \end{bmatrix}$$

can be understood as follows.

$$\begin{tikzcd}[row sep=small]
 & \bullet \arrow[r, squiggly, "1/2"] \arrow[dr, squiggly, "1/2"'] & x \\
\bullet \arrow[ur, squiggly, "1/3"] \arrow[dr, squiggly, "2/3"'] & & y \\
 & \bullet \arrow[r, squiggly, "1"'] & z
\end{tikzcd}
\quad \overset{\mu}{\longmapsto} \quad
\begin{tikzcd}[row sep=small]
 & x \\
\bullet \arrow[ur, squiggly, "1/6"] \arrow[r, squiggly, "1/6"] \arrow[dr, squiggly, "2/3"'] & y \\
 & z
\end{tikzcd}
, \tag{8}$$

Furthermore, these monads come with natural order structures which turn out to be ω-CPOs. This is exploited in [29] to prove—using a domain-theoretic technique from [42]—that a final coalgebra in $\mathcal{K}\ell(B)$ coincides with an initial algebra in **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 [43].

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 $\mathbb{C}$ on which one can build a GoI interpretation. Such a category $\mathbb{C}$ (together with some auxiliary data) is called a *GoI situation* in [21]: among other conditions, a crucial one is that $\mathbb{C}$ is a *traced symmetric monoidal category (TSMC)* [44]. Then applying what they call the *GoI construction* $\mathcal{G}$—it is isomorphic to the Int-*construction* in [44]—one is led to a compact closed category $\mathcal{G}(\mathbb{C})$ of "bidirectional computations" or "(stateless) games."

The resulting category $\mathcal{G}(\mathbb{C})$ 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 $\mathcal{G}(\mathbb{C})$. 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 Definition 4.10 later).

A thorough introduction to the rich and deep theory of GoI is certainly out of the current paper's scope. We shall nevertheless provide further intuitions—in a way tailored to categorical GoI and our use of it—later in §4.3.

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 (the last paragraph of) §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 **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 $\mathcal{K}\ell(\mathcal{Q})$ 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 λ-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 $\mathbf{PER}_A$ 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 $\mathbf{PER}_A$ is represented by a *code* $c \in A$.[4]

To turn $\mathbf{PER}_A$ into a model of a typed linear λ-calculus (more specifically into a linear category) one needs type constructors like $\otimes$, $\multimap$ and ! on $\mathbf{PER}_A$. They can be introduced by "programming in untyped linear λ-calculus"—it is much like encoding pairs, natural numbers, coproducts, etc. in the (untyped) λ-calculus ($\langle x, y\rangle := \lambda z.zxy$, 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 λ-calculus—i.e. a model of linear logic. There is a body of literature that seeks for what the latter means exactly—including [45, 46, 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 [47]; 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].

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

3. The Language Hoq

Here we introduce our target calculus. It is a variant of Selinger and Valiron's quantum λ-calculus [9]. The calculus shall be called Hoq—for *higher-order quantum computation*.[5]

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 λ-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-$\mathtt{qbit}$ stand for $\mathtt{qbit}^{\otimes n}$. The difference between n-$\mathtt{qbit} \boxtimes m$-$\mathtt{qbit}$ and $(n+m)$-$\mathtt{qbit}$ is: the former stands for two (n- and m-qubit) quantum states that are *for sure not entangled*; the latter is for the composite system in which two states are *possibly entangled*.

In contrast, in [9] 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 [9].

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 [9]. 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 *types* of Hoq are:

$$\begin{array}{l} A, B \quad ::= \quad n\text{-}\mathtt{qbit} \mid\, !\,A \mid A \multimap B \mid \top \mid A \boxtimes B \mid A + B \;, \\ \text{with conventions} \quad \mathtt{qbit} :\equiv 1\text{-}\mathtt{qbit} \quad \text{and} \quad \mathtt{bit} :\equiv \top + \top \;. \end{array} \tag{9}$$

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

[5] Hoq is a minor modification of the calculus $\mathbf{q}\lambda_\ell$ that we used in the conference version [1] of the current paper.

Definition 3.2 (Terms of Hoq). The *terms* of Hoq are:

$$
\begin{array}{rcl}
M, N, P & ::= & x \mid \lambda x^A.M \mid MN \mid \\
& & \langle M, N\rangle \mid \mathtt{let}\,\langle x^A, y^B\rangle = M\,\mathtt{in}\,N \mid \\
& & * \mid \mathtt{let}\,* = M\,\mathtt{in}\,N \mid \\
& & \mathtt{inj}^B_\ell\,M \mid \mathtt{inj}^A_r\,M \mid \mathtt{match}\,P\,\mathtt{with}\,(x^A \mapsto M \mid y^B \mapsto N) \mid \\
& & \mathtt{letrec}\,f^A x = M\,\mathtt{in}\,N \mid \\
& & \mathtt{new}_\rho \mid \mathtt{meas}^{n+1}_i \mid U \mid \mathtt{cmp}_{m,n}\ ,
\end{array}
$$

$$
\text{with conventions}\quad \mathtt{tt} :\equiv \mathtt{inj}^\top_\ell(*) \quad\text{and}\quad \mathtt{ff} :\equiv \mathtt{inj}^\top_r(*)\ . \tag{10}
$$

Here $m, n \in \mathbb{N}$ and $i \in [1, n+1]$ are natural numbers; $\rho \in \mathrm{DM}_{2^k}$ is a 2^k-dimensional density matrix (corresponding to a k-qubit system); U is a $2^k \times 2^k$ unitary matrix, for some $k \in \mathbb{N}$; and A and B are *type labels*. The terms are almost the same as in [9]; $\mathtt{new}_\rho$ designates *preparation* of a new quantum state—more precisely deployment of some quantum apparatus that is capable of preparing the quantum state ρ. The additional *composition* operator $\mathtt{cmp}$ will have the type m-$\mathtt{qbit} \boxtimes n$-$\mathtt{qbit} \multimap (m+n)$-$\mathtt{qbit}$ and embed nonentangled states as possibly entangled states. For measurements we have operators $\mathtt{meas}^1_1, \mathtt{meas}^2_1, \mathtt{meas}^2_2, \dots$; $\mathtt{meas}^{n+1}_i$ takes an $(n+1)$-qubit system, measures its i-th qubit, and returns the outcome (in the $\mathtt{bit}$ type) as well as the remaining quantum state that consists of n qubits.

The set $\mathrm{FV}(M)$ of *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 [9] and implicitly track the ! modality (see §3.3). The rules that derive <: are as follows.

$$
\begin{array}{c}
\dfrac{n = 0 \Rightarrow m = 0}{!^n\, k\text{-}\mathtt{qbit} <: !^m\, k\text{-}\mathtt{qbit}}\ (k\text{-}\mathtt{qbit}) \qquad \dfrac{n = 0 \Rightarrow m = 0}{!^n \top <: !^m \top}\ (\top) \\[2ex]
\dfrac{A_1 <: B_1 \quad A_2 <: B_2 \quad n = 0 \Rightarrow m = 0}{!^n(A_1 \boxtimes A_2) <: !^m(B_1 \boxtimes B_2)}\ (\boxtimes) \\[2ex]
\dfrac{A_1 <: B_1 \quad A_2 <: B_2 \quad n = 0 \Rightarrow m = 0}{!^n(A_1 + A_2) <: !^m(B_1 + B_2)}\ (+) \\[2ex]
\dfrac{B_1 <: A_1 \quad A_2 <: B_2 \quad n = 0 \Rightarrow m = 0}{!^n(A_1 \multimap A_2) <: !^m(B_1 \multimap B_2)}\ (\multimap)
\end{array}
\tag{11}
$$

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

We introduce the typing rules. They follow the ones in [9] and take subtyping into account. In the rules (Ax.1) and (Ax.2) the variables in the context can be thrown away, making the type system *affine* (weakening is allowed unconditionally while contraction is regulated by !) rather than *linear*. Due to these unconventional features (subtyping and weakening) a derivation of a type judgment is not necessarily unique in Hoq—making it a delicate issue whether the denotational semantics of a derivable type judgment is well-defined (Lemma 5.35).

$$\frac{A <: A'}{\Delta, x : A \vdash x : A'}\ \text{(Ax.1)} \qquad\qquad \frac{!\,\mathrm{DType}(c) <: A}{\Delta \vdash c : A}\ \text{(Ax.2)}$$

$$\frac{x : A, \Delta \vdash M : B \qquad A' <: A}{\Delta \vdash \lambda x^{A}.M : A' \multimap B}\ (\multimap.\mathrm{I}_1)$$

$$\frac{x : A, !\,\Delta, \Gamma \vdash M : B \qquad \mathrm{FV}(M) \subseteq |\Delta| \cup \{x\} \qquad A' <: A}{!\,\Delta, \Gamma \vdash \lambda x^{A}.M : !^{n}(A' \multimap B)}\ (\multimap.\mathrm{I}_2)$$

$$\frac{!\,\Delta, \Gamma_1 \vdash M : A \multimap B \qquad !\,\Delta, \Gamma_2 \vdash N : C \qquad C <: A}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash MN : B}\ (\multimap.\mathrm{E})$$

$$\frac{!\,\Delta, \Gamma_1 \vdash M_1 : !^{n} A_1 \qquad !\,\Delta, \Gamma_2 \vdash M_2 : !^{n} A_2}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash \langle M_1, M_2 \rangle : !^{n}(A_1 \boxtimes A_2)}\ (\boxtimes.\mathrm{I})$$

$$\frac{!\,\Delta, \Gamma_1 \vdash M : !^{n}(A_1 \boxtimes A_2) \qquad !\,\Delta, \Gamma_2, x_1 : !^{n} A_1, x_2 : !^{n} A_2 \vdash N : A}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash \mathtt{let}\, \langle x_1^{!^{n} A_1}, x_2^{!^{n} A_2} \rangle = M \,\mathtt{in}\, N : A}\ (\boxtimes.\mathrm{E})$$

$$\frac{}{\Delta \vdash * : !^{n} \top}\ (\top.\mathrm{I}) \qquad\qquad \frac{!\,\Delta, \Gamma_1 \vdash M : \top \qquad !\,\Delta, \Gamma_2 \vdash N : A}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash \mathtt{let}\, * = M \,\mathtt{in}\, N : A}\ (\top.\mathrm{E})$$

$$\frac{\Delta \vdash M : !^{n} A_1 \qquad A_2 <: A_2'}{\Delta \vdash \mathtt{inj}_{\ell}^{A_2} M : !^{n}(A_1 + A_2')}\ (+.\mathrm{I}_1) \qquad \frac{\Delta \vdash N : !^{n} A_2 \qquad A_1 <: A_1'}{\Delta \vdash \mathtt{inj}_{r}^{A_1} N : !^{n}(A_1' + A_2)}\ (+.\mathrm{I}_2)$$

$$\frac{!\,\Delta, \Gamma \vdash P : !^{n}(A_1 + A_2) \qquad !\,\Delta, \Gamma', x_1 : !^{n} A_1 \vdash M_1 : B \qquad !\,\Delta, \Gamma', x_2 : !^{n} A_2 \vdash M_2 : B}{!\,\Delta, \Gamma, \Gamma' \vdash \mathtt{match}\, P \,\mathtt{with}\, (x_1^{!^{n} A_1} \mapsto M_1 \mid x_2^{!^{n} A_2} \mapsto M_2) : B}\ (+.\mathrm{E})$$

$$\frac{!\,\Delta, f : !(A \multimap B), x : A \vdash M : B \qquad !\,\Delta, \Gamma, f : !(A \multimap B) \vdash N : C}{!\,\Delta, \Gamma \vdash \mathtt{letrec}\, f^{A \multimap B} x = M \,\mathtt{in}\, N : C}\ \text{(rec)}$$

Table 1: Typing rules for Hoq

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

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

When we write Δ, Γ 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* $\mathrm{DType}(c)$

is defined as follows.

$$
\begin{array}{rcll}
\mathrm{DType}(\mathtt{new}_\rho) & :\equiv & k\text{-}\mathtt{qbit} & \text{for a density matrix } \rho \in \mathrm{DM}_{2^k} \\
\mathrm{DType}(\mathtt{meas}_i^{n+1}) & :\equiv & (n+1)\text{-}\mathtt{qbit} \multimap (!\,\mathtt{bit} \boxtimes n\text{-}\mathtt{qbit}) & \text{for } n \geq 1 \\
\mathrm{DType}(\mathtt{meas}_1^1) & :\equiv & \mathtt{qbit} \multimap !\,\mathtt{bit} & \\
\mathrm{DType}(U) & :\equiv & k\text{-}\mathtt{qbit} \multimap k\text{-}\mathtt{qbit} & \text{for a } 2^k \times 2^k \text{ unitary matrix } U \\
\mathrm{DType}(\mathtt{cmp}_{m,n}) & :\equiv & (m\text{-}\mathtt{qbit} \boxtimes n\text{-}\mathtt{qbit}) \multimap (m+n)\text{-}\mathtt{qbit} &
\end{array}
\tag{12}
$$

We shall write $\Pi \Vdash \Delta \vdash M : A$ if a derivation tree Π derives the type judgment. We write $\Vdash \Delta \vdash M : A$ if there exists such Π, 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 (Value, evaluation context). The *values* and *evaluation contexts* of Hoq are defined in the following (mostly standard) way.

$$
\begin{array}{lrl}
\text{Values} & V, V_1, V_2 \ ::= & x \mid \lambda x^A.M \mid \langle V_1, V_2\rangle \mid * \mid \mathtt{inj}_\ell^B\, V \mid \mathtt{inj}_r^A\, V \mid \\
& & \mathtt{new}_\rho \mid \mathtt{meas}_i^{n+1} \mid U \mid \mathtt{cmp}_{m,n} \ ;
\end{array}
$$

Evaluation contexts

$$
\begin{array}{rl}
E \ ::= & [\_] \mid E[\,[\_]M\,] \mid E[\,V[\_]\,] \mid E[\,\langle [\_], M\rangle\,] \mid E[\,\langle V, [\_]\rangle\,] \mid \\
& E[\,\mathtt{let}\,\langle x^A, y^B\rangle = [\_]\,\mathtt{in}\,M\,] \mid E[\,\mathtt{let}\,* = [\_]\,\mathtt{in}\,N\,] \mid \\
& E[\,\mathtt{inj}_\ell^B[\_]\,] \mid E[\,\mathtt{inj}_r^A[\_]\,] \mid E[\,\mathtt{match}\,[\_]\,\mathtt{with}\,(x^A \mapsto M \mid y^B \mapsto N)\,] \ .
\end{array}
$$

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

As usual, all the constants ($\mathtt{new}_\rho$, $\mathtt{meas}_i^{n+1}$, 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 3.6. *The following BNF notation defines the same notion of evaluation context as in Definition 3.5.*

$$
\begin{array}{rl}
D \ ::= & [\_] \mid DM \mid VD \mid \langle D, M\rangle \mid \langle V, D\rangle \mid \\
& \mathtt{let}\,\langle x^A, y^B\rangle = D\,\mathtt{in}\,M \mid \mathtt{let}\,* = D\,\mathtt{in}\,N \mid \\
& \mathtt{inj}_\ell^B D \mid \mathtt{inj}_r^A D \mid \mathtt{match}\,D\,\mathtt{with}\,(x^A \mapsto M \mid y^B \mapsto N) \ .
\end{array}
$$

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

Definition 3.7 (Small-step semantics). The *reduction rules* of Hoq are defined as follows. Each reduction $\longrightarrow$ is labeled with a real number from $[0,1]$.

$$
\begin{array}{lr}
E[\,(\lambda x^A.M)V\,] \longrightarrow_1 E[\,M[V/x]\,] & (\multimap)\\
E[\,\mathtt{let}\,\langle x^A, y^B\rangle = \langle V,W\rangle\,\mathtt{in}\,M\,] \longrightarrow_1 E[\,M[V/x,W/y]\,] & (\boxtimes)\\
E[\,\mathtt{let}\,{*} = {*}\,\mathtt{in}\,M\,] \longrightarrow_1 E[\,M\,] & (\top)\\
E[\,\mathtt{match}\,(\mathtt{inj}^C_\ell V)\,\mathtt{with}\,(x^{!^n A} \mapsto M \mid y^{!^n B} \mapsto N)\,] \longrightarrow_1 E[\,M[V/x]\,] & (+_1)\\
E[\,\mathtt{match}\,(\mathtt{inj}^C_r V)\,\mathtt{with}\,(x^{!^n A} \mapsto M \mid y^{!^n B} \mapsto N)\,] \longrightarrow_1 E[\,N[V/y]\,] & (+_2)\\
E[\,\mathtt{letrec}\,f^{A\multimap B}x = M\,\mathtt{in}\,N\,] \longrightarrow_1 E\big[\,N\big[\,(\lambda x^A.\,\mathtt{letrec}\,f^{A\multimap B}x = M\,\mathtt{in}\,M)/f\,\big]\,\big] & (\mathrm{rec})\\
E[\,U(\mathtt{new}_\rho)\,] \longrightarrow_1 E[\,\mathtt{new}_{U\rho U^\dagger}\,] & (U)\\
E[\,\mathtt{cmp}_{m,n}\langle \mathtt{new}_\rho, \mathtt{new}_\sigma\rangle\,] \longrightarrow_1 E[\,\mathtt{new}_{\rho\otimes\sigma}\,] & (\mathtt{cmp})\\
E[\,\mathtt{meas}^{n+1}_i(\mathtt{new}_\rho)\,] \longrightarrow_1 E[\,\langle\,\mathtt{tt},\,\mathtt{new}_{\langle 0_i|\rho|0_i\rangle}\,\rangle\,] & (\mathtt{meas}_1)\\
E[\,\mathtt{meas}^{n+1}_i(\mathtt{new}_\rho)\,] \longrightarrow_1 E[\,\langle\,\mathtt{ff},\,\mathtt{new}_{\langle 1_i|\rho|1_i\rangle}\,\rangle\,] & (\mathtt{meas}_2)\\
E[\,\mathtt{meas}^1_1(\mathtt{new}_\rho)\,] \longrightarrow_{\langle 0|\rho|0\rangle} E[\,\mathtt{tt}\,] & (\mathtt{meas}_3)\\
E[\,\mathtt{meas}^1_1(\mathtt{new}_\rho)\,] \longrightarrow_{\langle 1|\rho|1\rangle} E[\,\mathtt{ff}\,] & (\mathtt{meas}_4)
\end{array}
$$

Here M,N are terms, V,W are values and $n \geq 1$ is a natural number. The reductions that involve $\mathtt{new}_\rho$—namely $(U), (\mathtt{cmp}), (\mathtt{meas}_1$–$\mathtt{meas}_4)$—occur only when the dimensions match. The last four rules are called *measurement rules*. They always give rise to two reductions in a pair (corresponding to observing $\mathtt{tt}$ or $\mathtt{ff}$); the pair are said to be the *buddy* to each other.

Observe that the label p in reduction $\longrightarrow_p$ is like a probability but not quite: from $\mathtt{meas}^2_i(\mathtt{new}_\rho)$ there are two $\longrightarrow_1$ reductions, to $\mathtt{new}_{\langle 0_i|\rho|0_i\rangle}$ and to $\mathtt{new}_{\langle 1_i|\rho|1_i\rangle}$. We understand that the probabilities are implicitly carried by the trace values of the matrices $\langle 0_i|\rho|0_i\rangle$ and $\langle 1_i|\rho|1_i\rangle$. See (7) and the remarks that follow it.

As is standardly done, we will prove adequacy of our denotational semantics focusing on $\mathtt{bit}$-type closed terms. For that purpose we now introduce big-step semantics for such terms.[6]

Definition 3.8 (Big-step semantics). For each $n \in \mathbb{N}$ we define a relation $\curlyvee^n$ between closed $\mathtt{bit}$-terms M and pairs (p,q) of real numbers. This is by induction on n.

For $n = 0$, we define

$$\mathtt{tt} \curlyvee^0 (1,0)\,,\ \mathtt{ff} \curlyvee^0 (0,1)\,,\ \text{and } M \curlyvee^0 (0,0)\ \text{ for the other } M.$$

For $n+1$, if M has a reduction $M \longrightarrow_1 M'$ caused by a rule other than the measurement rules, we set:

$$M \curlyvee^{n+1} (p,q) \quad \overset{\text{def.}}{\Longleftrightarrow} \quad M' \curlyvee^n (p,q)\ .$$

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

$$M \curlyvee^{n+1} (rp + r'p', rq + r'q') \quad \overset{\text{def.}}{\Longleftrightarrow} \quad N \curlyvee^n (p,q) \text{ and } N' \curlyvee^n (p',q')\ .$$

[6] We swapped the notations $\curlyvee$ and $\Downarrow$ from the previous version [1].

Finally, we define a relation $\curlyvee$ as the supremum of $\curlyvee^n$. That is,

$$M \curlyvee (p,q) \quad \stackrel{\text{def.}}{\Longleftrightarrow} \quad (p,q) \;=\; \sup\big\{\, (p',q') \mid M \curlyvee^n (p',q') \text{ for some } n \,\big\} \;,$$

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 $M \curlyvee^n (p,q)$. The same holds for $\curlyvee$, too.

The intuition of $M \curlyvee (p,q)$ is: the term M (which is closed and of type `bit`) reduces eventually to `tt` with the probability p; to `ff` with the probability q.

Remark 3.9. 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 λ-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 `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 λ-calculi, including the one in [30, 31], typically have explicit syntax for operating on the ! modality. An example is the `derelict` operator in

$$\frac{\Gamma \vdash M \,:\, !A}{\Gamma \vdash \mathtt{derelict}\, M \,:\, A} \quad . \tag{13}$$

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 `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 $\boxtimes$ from $\otimes$ (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 $\mathtt{qbit} \otimes \mathtt{qbit}$; 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 Remark 3.9).

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 $\multimap$ and recursion that go along well with $\boxtimes$.

The latter is not easily the case with $\otimes$. While the duality $\mathcal{H}^* \cong \mathcal{H}$ gives a compact closed structure (hence the interpretation of $\multimap$) to the category **fdHilb** of *finite-dimensional* Hilbert spaces, such is not available in the category **Hilb** of general Hilbert spaces, on the one hand. On the other hand, **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.[7]

Remark 3.10 (Other categorical models for higher-order quantum computation). 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 $\mathbb{C}$ in a monoidal closed category $[\mathbb{C}^{\mathrm{op}}, \mathbf{Sets}]$ [48]. The base category is $\mathbb{C} = \mathbf{Q}$ 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.

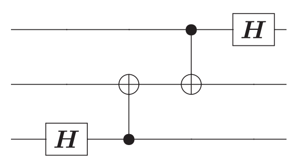
[7] In [6, §1] it is argued—rather on a conceptual level—that the Hilbert space tensor $\otimes$ does not seem quite compatible with a closed structure (i.e. with respect to $\multimap$).

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 categorical 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 3.11 (Other GoI models for higher-order quantum computation). In contrast to the works discussed previously in Remark 3.10, the line of work [49, 50, 51, 11] aims at models for higher-order quantum computation with strong operational flavors—given by token-based presentations of GoI—rather than categorical models. Unlike the current work where we distinguish $\boxtimes$ and $\otimes$, they do use the same tensor $\otimes$ for both. The price to pay is that they in [49, 50, 51, 11] need *multiple* tokens—one for each qubit in entanglement—and they need to *synchronize* from time to time, e.g. when they go through a multi-qubit unitary gate.

Currently it is not clear how the multi-token GoI machines in [49, 50, 51, 11] can be understood as instances of categorical GoI in [21], or how those machines can be organized to form a categorical model. These research questions seem to be important ones, all the more since the significance of multi-token GoI machines seems to go beyond their roles in modeling quantum computation. For example, in [52], it is shown that they successfully capture the difference between CBV and CBN evaluation strategies of PCF, via translations to a linear λ-calculus—much like CPS translation does in the classic work of Plotkin [53].

Remark 3.12 ("Quantum circuits" in Hoq). Sequential and parallel compositions of unitary gates—much like those found in *quantum circuits*—are pervasive in quantum computation.



Many higher-order functional languages for quantum computation (such as [9, 15, 49, 50, 51]) allow to express such compositions in a straightforward manner, exploiting their coincidence of $\otimes$ (for composite quantum systems) and $\boxtimes$ (as a type constructor). For example, in [50] their proof nets (on which they define multi-token GoI machines) are claimed to be one realization of *higher-order quantum circuits*.

This is not the case with Hoq, unfortunately: due to the separation of $\boxtimes$ from $\otimes$, there is no type-theoretic infrastructure that supports the above quantum circuit-like compositions of gates. See also an example in §3.5.

We can however foresee that the kind of (inherently first-order) typing disciplines that would be needed for the above quantum circuit-like compositions are fairly simple. It is not hard to add one of such typing principles to Hoq—whose current type system's job is to take care of higher-order control structures that are much more complicated—as an additional layer. In other words, in such an extension of Hoq we will have a first-order fragment of the language that is devoted to composing quantum gates. We shall not describe such a straightforward extension in the current paper, since our focus here is on the integration of classical and quantum information using the GoI and realizability techniques. We note that, in such an extension, it will suffice to have a finite *universal* set of quantum gates as primitives (any other gate can be approximately expressed in the language).

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 $\boxtimes$ from $\otimes$, 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 (Lemma 5.35)—a delicate issue with Hoq especially because of the subtype relation $<:$.

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 always trivial if the interpretation $[\![\Delta \vdash M : A]\!]$ is well-defined or not.

It is in fact nontrivial already for the simply typed λ-calculus in the Curry-style (i.e. variables' types are not predetermined but are specified in type contexts). 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 [54] a Church-style calculus is used (variables come with their intrinsic types); in [55] its Curry-style calculus has explicit type labels (much like in Hoq).

For the Curry-style simply typed λ-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 [56, 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 linearity tracking (explicit operators like `derelict` in (13), instead of the subtype relation $<:$), 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 Lemma 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 B.

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

Notation 3.16 (Subtyping $<:$ between contexts). We write $\Delta' <: \Delta$ when:

- $|\Delta'| = |\Delta|$, and
- for each $(x : A') \in \Delta'$, there is $(x : A) \in \Delta$ with $A' <: A$.

In Table 1, some rules including $(\multimap.\mathrm{I}_1)$ have *type coercion*: while the term $\lambda x^A.M$ has a type label A, its actual type is $A' \multimap B$ with $A' <: A$. This is so that the following holds.

Lemma 3.17. *The* monotonicity *rule is admissible in Hoq.*

$$\frac{\Delta' <: \Delta \quad \Delta \vdash M : A \quad A <: A'}{\Delta' \vdash M : A'}\ (\mathit{Mon})$$

$\square$

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

$$\frac{\Delta \vdash M : !A}{\Delta \vdash M : A}\ (\mathit{Der}) \qquad \frac{\Delta \vdash M : !A}{\Delta \vdash M : !!A}\ (\mathit{Comult})$$

$\square$

Lemma 3.19. 1. *If* $\Vdash \Delta \vdash M : B$, *then* $\mathrm{FV}(M) \subseteq |\Delta|$.

2. *If* $x \notin \mathrm{FV}(M)$ *and* $\Vdash \Delta, x : A \vdash M : B$, *then* $\Vdash \Delta \vdash M : B$.
3. *The following rule is admissible.*

$$\frac{\Delta \vdash M : A}{\Gamma, \Delta \vdash M : A} \ (\textit{Weakening})$$

PROOF. Straightforward, by induction on derivation. □

Many linear lambda calculi have the *promotion* rule

$$\frac{!\,\Delta \vdash M : A}{!\,\Delta \vdash \texttt{promote}\, M : !\,A} \ (\mathrm{Prom})$$

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* $x \in \mathrm{FV}(V)$, *we have* $(x : !\,B) \in \Delta$ *for some type* B.
2. *Conversely, the following rule is admissible in Hoq.*

$$\frac{!\,\Delta, \Gamma \vdash V : A \qquad \mathrm{FV}(V) \subseteq |\Delta|}{!\,\Delta, \Gamma \vdash V : !\,A} \ (\textit{ValProm}) \ , \quad \textit{where } V \textit{ is a value.}$$

□

Remark 3.21 (No-cloning). We note that all constants are values (Definition 3.5); therefore the last result (or ultimately the rule (Ax.2) in Table 1) implies that any constant can have a type $!\,A$, where A is the default type of the constant (Definition 3.4). This can raise a suspicion that our calculus Hoq does not respect *no-cloning*, one of the most fundamental principles in quantum mechanics: it dictates that no quantum state should be duplicable (unless it is *classical* in a suitable sense); still our primitive $\texttt{new}_\rho$ can have a type $!\,k\text{-}\texttt{qbit}$ and hence is duplicable.

We believe this is not problematic. As discussed briefly after Definition 3.2, we understand the constant $\texttt{new}_\rho$ to stand for "deployment of some quantum apparatus that is capable of preparing the quantum state ρ." In this viewpoint it is no problem that the term

$$\big(\ \lambda x^{!\,\texttt{qbit}}.\, \texttt{cmp}_{1,1}\langle x, x\rangle\ \big)(\texttt{new}_{|0\rangle\langle 0|})$$

is typable—it just denotes that we run an apparatus that prepares the state $|0\rangle$ twice to obtain the state $|00\rangle$.

Still the no-cloning property is discerned in the calculus Hoq. For example, the term

$$\lambda x^{\texttt{qbit}}.\, \texttt{cmp}_{1,1}\langle x, x\rangle \ ,$$

without ! in its argument type, is not typable. This means we cannot duplicate a quantum state whose preparation apparatus we do not have access to.

A general *substitution* rule

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

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

$$x : \mathtt{qbit} \multimap !\,\mathtt{qbit}, y : \mathtt{qbit} \vdash xy : !\,\mathtt{qbit} \qquad w : !\,\mathtt{qbit} \vdash \lambda z^{!\,\mathtt{qbit}}.\, w : !(!\,\mathtt{qbit} \multimap !\,\mathtt{qbit})$$

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

$$x : \mathtt{qbit} \multimap !\,\mathtt{qbit}, y : \mathtt{qbit} \vdash \lambda z^{!\,\mathtt{qbit}}.\, xy : !(!\,\mathtt{qbit} \multimap !\,\mathtt{qbit})$$

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

Lemma 3.22 (Substitution)**.** *The following rules are admissible in Hoq.*

$$\frac{!\,\Delta, \Gamma_1 \vdash M : A \qquad !\,\Delta, \Gamma_2, x : A \vdash N : B \qquad A \not\equiv !\,A' \text{ for any } A'}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash N[M/x] : B}\ (\mathit{Subst}_1)$$

$$\frac{!\,\Delta \vdash M : A \qquad !\,\Delta, \Gamma_2, x : A \vdash N : B}{!\,\Delta, \Gamma_2 \vdash N[M/x] : B}\ (\mathit{Subst}_2)$$

$$\frac{!\,\Delta, \Gamma_1 \vdash V : A \qquad !\,\Delta, \Gamma_2, x : A \vdash N : B \qquad V \text{ is a value}}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash N[V/x] : B}\ (\mathit{Subst}_3)$$

$$\frac{!\,\Delta, \Gamma_1 \vdash M : A \qquad !\,\Delta, \Gamma_2, x : A \vdash E[x] : B \qquad x \text{ does not occur in } E}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash E[M] : B}\ (\mathit{Subst}_4)$$

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 (Definition 3.5); the side condition means that* x *occurs exactly once in the term* $E[x]$*.* $\square$

Lemma 3.23 (Subject reduction)**.** *Assume that* $\Vdash \Delta \vdash M : A$*, and that there is a reduction* $M \longrightarrow_p N$ *(we allow* $p = 0$*). Then* $\Vdash \Delta \vdash N : A$*.* $\square$

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

We note that, given M, the sum of the values p for all possible reductions $M \longrightarrow_p N$ is not necessarily equal to 1. See the remark right after Definition 3.7.

3.5. Hoq Programs: an Example

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. The example, although first-order, will exemplify the expressivity as well as inexpressivity of the calculus Hoq: it fully supports preparation of quantum states and unitary transformations; it also features (classical) branching based on measurement outcomes; however, due to the distinction between $\otimes$ and $\boxtimes$, composition of unitary transformations—one that is much like in quantum circuits—is less straightforward to express (cf. Remark 3.12). See (14) where an explicit use of `cmp` is required, and some discussions that follow it.

Potential use of *higher-order* quantum programs has already been advocated by many authors; see e.g. [9, 2, 6]. They could also be used in formalizing games in *quantum game theory*, a formalism that is attracting increasing attention as a useful presentation of quantum nonlocality (see e.g. [57]). We here use a first-order example of quantum teleportation, however, since it is one of the most well-known quantum procedures.

In the quantum teleportation protocol, Alice and Bob 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`:

$$\mathtt{EPR} \;:\equiv\; U_0\left(\mathtt{cmp}_{1,1}\langle \mathtt{new}_{|0\rangle\langle 0|}, \mathtt{new}_{|0\rangle\langle 0|}\rangle\right) \qquad : \quad \mathtt{2\text{-}qbit} \tag{14}$$

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:

$$U_0 = N(H \otimes \mathcal{I}_2) \;,$$

where $\mathcal{I}_2$ is the 2×2 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. We need to calculate U_0 outside Hoq; see the discussions in Remark 3.12.

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

$$\begin{aligned} \mathtt{Bellmeasure} :\equiv \quad & \lambda w^{\mathtt{3\text{-}qbit}}.\, \mathtt{let}\ \langle b_0^{\mathtt{bit}}, p^{\mathtt{2\text{-}qbit}}\rangle = \mathtt{meas}_1^3\,(U_1 w)\ \mathtt{in} \\ & \qquad \mathtt{let}\ \langle b_1^{\mathtt{bit}}, q^{\mathtt{qbit}}\rangle = \mathtt{meas}_1^2\, p\ \mathtt{in}\ \langle b_0, \langle b_1, q\rangle\rangle \\ & \qquad\qquad : \quad \mathtt{3\text{-}qbit} \multimap \mathtt{bit} \boxtimes (\mathtt{bit} \boxtimes \mathtt{qbit}) \end{aligned} \tag{15}$$

Here U_1 is the following unitary transformation:

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

which is equal to $\big((H \otimes \mathcal{I}_2)N\big) \otimes \mathcal{I}_2$. Although the third qubit has nothing to do with the Bell measurement, we need to include the third qubit in the program (15) because the type $2\text{-}\mathtt{qbit} \boxtimes \mathtt{qbit}$ is different from $3\text{-}\mathtt{qbit}$ 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:

$$\begin{aligned} \mathtt{corr} \ :\equiv \quad & \lambda x^{\mathtt{bit}\boxtimes(\mathtt{bit}\boxtimes\mathtt{qbit})}.\, \mathtt{let}\ \langle b_0^{\mathtt{bit}}, y^{\mathtt{bit}\boxtimes\mathtt{qbit}} \rangle = x \ \mathtt{in}\ \mathtt{let}\ \langle b_1^{\mathtt{bit}}, q^{\mathtt{qbit}} \rangle = y \ \mathtt{in} \\ & \qquad \mathtt{match}\ b_0\ \mathtt{with}\ (\\ & \qquad\quad z_0^{\top} \mapsto \mathtt{match}\ b_1\ \mathtt{with}\ (w_0^{\top} \mapsto U_{00}q \mid w_1^{\top} \mapsto U_{01}q) \\ & \qquad\quad \mid z_1^{\top} \mapsto \mathtt{match}\ b_1\ \mathtt{with}\ (w_0^{\top} \mapsto U_{10}q \mid w_1^{\top} \mapsto U_{11}q)) \\ & \qquad\qquad : \quad \mathtt{bit} \boxtimes (\mathtt{bit} \boxtimes \mathtt{qbit}) \multimap \mathtt{qbit} \end{aligned}$$

where U_{ij} are given as follows.

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

The result is the qubit that Alice wishes to send to Bob.

We combine the above programs into one: we define a closed value $\mathtt{qtel} : \mathtt{qbit} \multimap \mathtt{qbit}$ to be

$$\lambda x^{\mathtt{qbit}}.\, \mathtt{corr}\big(\mathtt{Bellmeasure}\big(\,\mathtt{cmp}_{1,2}\,\langle x, \mathtt{EPR}\rangle\,\big)\big) \ .$$

We can observe that Bob receives Alice's qubit.

Proposition 3.25. *For any unitary transformation U, the reduction tree of* $\mathtt{qtel}$ *is of the following form.*

$$\begin{tikzcd} & & & \mathtt{qtel}\,(U\,(\mathtt{new}_{|0\rangle\langle 0|})) \arrow[dlll, "*"', "1"] \arrow[dl, "*", "1"'] \arrow[dr, "*"', "1"] \arrow[drrr, "*", "1"'] & & & \\ \mathtt{new}_{\frac{1}{4}\rho} & & \mathtt{new}_{\frac{1}{4}\rho} & & \mathtt{new}_{\frac{1}{4}\rho} & & \mathtt{new}_{\frac{1}{4}\rho} \end{tikzcd}$$

Here $\rho \in \mathrm{DM}_2$ is $U\,|0\rangle\langle 0|\,U^{\dagger}$ and $\xrightarrow{*}_1$ is the transitive closure of $\rightarrow_1$. □

3.5.1. Fair Coin Toss

This example is in [9]: it simulates a fair coin toss with quantum primitives.

$$\texttt{fcoin} :\equiv H(\texttt{new}\,\texttt{tt}) \qquad\qquad \texttt{toss} :\equiv \lambda x^{\texttt{qbit}}.\,\texttt{meas}^1_1\, x$$

Proposition 3.26. *The reduction tree of* `toss fcoin` *is the following one.*

$$\begin{tikzcd}[column sep=small] & \texttt{toss fcoin} \arrow[dl, "\frac{1}{2}"'] \arrow[dr, "\frac{1}{2}"] & \\ \texttt{tt} & & \texttt{ff} \end{tikzcd}$$

Here we omitted reductions $\to_1$. □

4. The Quantum Branching Monad $\mathcal{Q}$ and The Category $\mathbf{PER}_{\mathcal{Q}}$

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 Remark 2.15. This relationship establishes the first among the three steps in Figure 1.

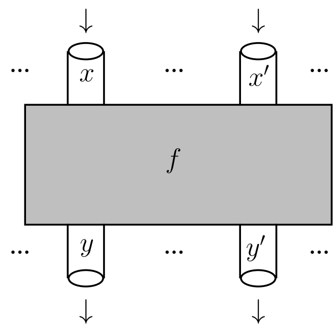
Examples of a traced monoidal category $\mathbb{C}$ used in categorical GoI [21] are divided into two groups: the so-called *wave-style* ones where $\mathbb{C}$'s monoidal structure is given by products $\times$; and the *particle-style* ones where it is given by coproducts $+$. The former are of *static* nature and includes domain-theoretic examples like ω-**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 **Pfn** of sets and partial functions;
- the category $\mathbf{Rel}_+$ 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 **SRel** of measurable spaces and stochastic relations.

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

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

Let us elaborate on such a Kleisli category $\mathcal{K}\ell(B)$. We look at it as a category of *piping*. An arrow[8] $f : X \nrightarrow Y$ in $\mathcal{K}\ell(B)$ is understood as a bunch of pipes, with $|X|$-many entrances and $|Y|$-many exits.[9] The pipes are where a *particle* (or *token*) runs through. See below; here the shaded box f consists of a lot of pipes.



(16)

According to the choice of a monad B (see Example 2.14), different "branching" of such pipes is allowed.

- When $B = \mathcal{L}$ 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 $B = \mathcal{P}$ a pipe can branch, with one entrance x connected to possibly multiple (or zero) exits (namely those in $f(x) \subseteq Y$).
- When $B = \mathcal{D}$ a pipe can branch too, but this time the branching is probabilistic.

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

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

---

[8] We shall use $\nrightarrow$ to denote an arrow in a Kleisli category.

[9] Our piping analogy is not completely faithful: in a Kleisli arrow f the two crossings and are identified, but they are different as physical pipes.

to find a branching monad with a quantum flavor. Our answer is the *quantum branching monad* $\mathcal{Q}$ that we introduce now.

4.2. The Quantum Branching Monad $\mathcal{Q}$

The following formal definition of $\mathcal{Q}$ below is hardly illustrative. The intuition will be explained shortly, using the piping analogy (16) for arrows in the Kleisli category $\mathcal{K}\ell(\mathcal{Q})$.

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

$$\mathcal{Q}X \;=\; \Big\{ c : X \to \prod_{m,n\in\mathbb{N}} \mathrm{QO}_{m,n} \;\Big|\; \text{the trace condition (17)} \Big\}$$

where: $\mathrm{QO}_{m,n}$ is the set of quantum operations of the type $\mathrm{DM}_m \to \mathrm{DM}_n$ (Definition 2.6); $\prod_{m,n\in\mathbb{N}}$ denotes a Cartesian product; and the *trace condition* stands for the following.

$$\sum_{x\in X}\sum_{n\in\mathbb{N}} \mathsf{tr}\Big(\big(c(x)\big)_{m,n}(\rho)\Big) \leq 1 \;\;,\; \forall m \in \mathbb{N},\; \forall \rho \in \mathrm{DM}_m. \tag{17}$$

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

$$\big(\,(\mathcal{Q}f)(c)(y)\,\big)_{m,n} \;:=\; \sum_{x\in f^{-1}(\{y\})} \big(\,c(x)\,\big)_{m,n} \;. \tag{18}$$

Note that the sum on the right-hand side is well-defined, because of the upper bound given by the trace condition (17). As for the monad structure, its unit $\eta_X : X \to \mathcal{Q}X$ is:

$$\big(\,\eta_X(x)(x')\,\big)_{m,n} \;:=\; \begin{cases} \mathrm{id}_m & \text{if } x = x' \text{ and } m = n, \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

Here id_m is the identity map; 0 is the constant QO that maps everything to 0. The multiplication $\mu_X : \mathcal{Q}\mathcal{Q}X \to \mathcal{Q}X$ is defined by:

$$\big(\,\mu_X(\gamma)(x)\,\big)_{m,n} \;:=\; \sum_{c\in\mathcal{Q}X}\sum_{k\in\mathbb{N}} \Big(\big(c(x)\big)_{k,n} \circ \big(\gamma(c)\big)_{m,k}\Big) \;. \tag{20}$$

The QO $\big(c(x)\big)_{k,n} \circ \big(\gamma(c)\big)_{m,k}$ on the RHS is the sequential composition of QOs: given a density matrix $\rho \in \mathrm{DM}_m$ it first applies $(\gamma(c))_{m,k} \in \mathrm{QO}_{m,k}$ and then applies $(c(x))_{k,n} \in \mathrm{QO}_{k,n}$, transforming ρ eventually into an n-dimensional density matrix.

In Appendix C we prove that the sums in (18) and (20) 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:

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

Specifically:

- For $B = \mathcal{L}$ the set W is $2 = \{0, 1\}$ (*stuck* or *through*); the normalizing condition is
$$c(x) = 1 \quad \text{for at most one } x \in X.$$
- 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 $\sum_{x \in X} c(x) \le 1$.
- For $B = \mathcal{Q}$ the weights are a tuple (or a block matrix) of quantum operations and the normalizing condition is the trace condition (17).

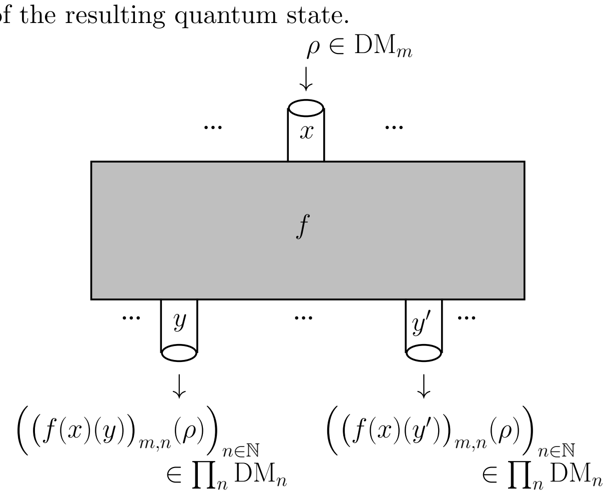
Let us continue (16) and think of an arrow $f : X \nrightarrow Y$ in $\mathcal{K}\ell(\mathcal{Q})$ as piping. The piping analogy is still valid for $\mathcal{Q}$; a crucial difference however is that, for $B = \mathcal{Q}$,

$$\boxed{\begin{array}{c}\text{a token that runs through pipes is no longer a mere particle,}\\ \text{but it } \textit{carries a quantum state.}\end{array}}$$

Each entrance $x \in X$ is ready for an incoming token that carries $\rho \in \mathrm{DM}_m$ 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 $n \in \mathbb{N}$. 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

$$\bigl(\, f(x)(y)\,\bigr)_{m,n} \;\in\; \mathrm{QO}_{m,n} \;,$$

that is determined by: which exit $y \in Y$ the token takes; and what is the dimension n of the resulting quantum state.



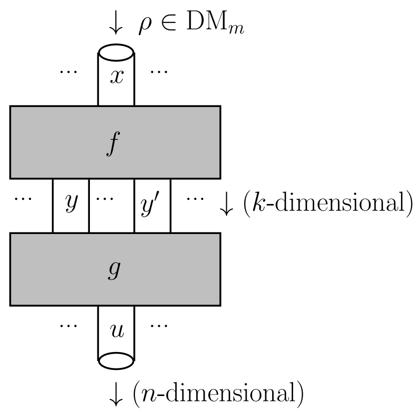
(21)

The trace condition (17) now reads, for an arrow $f : X \rightarrow\!\!\!\!\!\!\rightarrow Y$ in $\mathcal{K}\ell(\mathcal{Q})$:

$$\sum_{y\in Y}\sum_{n\in\mathbb{N}} \mathsf{tr}\Big(\big(f(x)(y)\big)_{m,n}(\rho)\Big) \;\leq\; 1 \;\;, \text{ for each } m\in\mathbb{N},\, \rho\in \mathrm{DM}_m \text{ and } x\in X. \tag{22}$$

The trace value $\mathsf{tr}\Big(\big(f(x)(y)\big)_{m,n}(\rho)\Big)$ is understood as the probability with which a token ρ 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 (22).

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



(23)

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 $\odot$ is described as follows.

Lemma 4.2 (Composition $\odot$ in $\mathcal{K}\ell(\mathcal{Q})$)**.** *Given two successive arrows $f : X \rightarrow\!\!\!\!\!\!\rightarrow Y$ and $g : Y \rightarrow\!\!\!\!\!\!\rightarrow U$ in $\mathcal{K}\ell(\mathcal{Q})$, their composition $g\odot f : X \rightarrow\!\!\!\!\!\!\rightarrow U$ is concretely given as follows.*

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

Proof. See Appendix C.1.

This description of $\odot$ in $\mathcal{K}\ell(\mathcal{Q})$ is ultimately due to the definition (20) of the multiplication operation μ. We notice its similarity to the multiplication operation of the subdistribution monad $\mathcal{D}$. The latter is defined by

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

where $\cdot$ denotes multiplication of real numbers. This notably resembles (20).

Remark 4.3. The reason why $\mathcal{Q}$ is called a quantum branching monad is that a Kleisli arrow $f : X \nrightarrow Y$ in $\mathcal{K}\ell(\mathcal{Q})$—thought of as piping like (21)—is a "quantum branching function." This is in the same sense as an arrow $f : X \nrightarrow Y$ in $\mathcal{K}\ell(\mathcal{P})$ is a "nondeterministically branching function" and an arrow $f : X \nrightarrow Y$ in $\mathcal{K}\ell(\mathcal{D})$ is a "probabilistically branching function."

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

$$f_1 \;:\; \mathbb{N} \longrightarrow\!\!\!\!\!\!\mapsto \mathbb{N} \;,$$
$$\bigl(f_1(k)(l)\bigr)_{m,n}(\rho) := \begin{cases} \langle 0|\rho|0\rangle & \text{if } m = 2,\ n = 1 \text{ and } l = 2k, \\ \langle 1|\rho|1\rangle & \text{if } m = 2,\ n = 1 \text{ and } l = 2k+1, \\ 0 & \text{otherwise.} \end{cases}$$

Imagine a token carrying a quantum state $\rho \in \mathrm{DM}_m$ entering this piping at the entrance $k \in \mathbb{N}$. The token does not come out at all unless ρ is 2-dimensional. If ρ 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 Remark 2.11).

A slightly more complicated example is the following f_2. Here $N \in \mathbb{N}$ is a natural number.

$$f_2 \;:\; \mathbb{N} \longrightarrow\!\!\!\!\!\!\mapsto \mathbb{N} \;,$$
$$\bigl(f_2(k)(l)\bigr)_{m,n}(\rho) := \begin{cases} \langle 0_1|\rho|0_1\rangle & \text{if } m = 2^{N+1},\ n = 2^N \text{ and } l = 2k, \\ \langle 1_1|\rho|1_1\rangle & \text{if } m = 2^{N+1},\ n = 2^N \text{ and } l = 2k+1, \\ 0 & \text{otherwise.} \end{cases}$$

Here an incoming token carries an $(N + 1)$-qubit state $\rho \in \mathrm{DM}_{2^{N+1}}$, 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 outcoming token carries a state

$$\langle 0_1|\rho|0_1\rangle \quad \text{or} \quad \langle 1_1|\rho|1_1\rangle \quad \in \mathrm{DM}_{2^N}$$

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 Remark 2.11.

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

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

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

The monad $\mathcal{Q}$ 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 $\mathcal{K}\ell(\mathcal{Q})$ is a *traced symmetric monoidal category (TSMC)*.

Definition 4.4 (Order $\sqsubseteq$ on $\mathcal{Q}X$). We endow the set $\mathcal{Q}X$ with the pointwise extension of the Löwner partial order in Definition 2.12. Namely: given $c, d \in \mathcal{Q}X$, 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 $\mathcal{K}\ell(\mathcal{Q})$ is partially additive (a notion from [58]). Therefore by [59, Chap. 3], $(\mathcal{K}\ell(\mathcal{Q}), 0, +)$ is a TSMC, with its trace operator given explicitly by Girard's execution formula.*

PROOF. By [33, Proposition 4.8]; see Theorem Appendix C.5 for details. Notably, the Kleisli category $\mathcal{K}\ell(\mathcal{Q})$ is ω-**CPO** enriched: a homset $\mathcal{K}\ell(\mathcal{Q})(X, Y)$ is equipped with the order that is the pointwise extension of that on $\mathcal{Q}Y$ (Definition 4.4). The trace operator will be described later, in the proof of Theorem 4.14. □

Remark 4.6. Continuing Remark 4.3, we note that in the field of quantum programming languages the study of *quantum control* is emerging. With initial observations in [60, 61], this line of work aims at extending the by-now accepted paradigm of *classical control and quantum data*. Under quantum control, the current program counter of a program's execution can be a quantum superposition of multiple program locations. In the token analogue in the current section, this means that the token's position can be superposed—this is not possible with our current monad $\mathcal{Q}$, where a token's position can be a probabilistic ensemble of different positions but is never a quantum superposition.

Some authors use the terminologies *classical* and *quantum branching* to distinguish the choices of classical/quantum control structures. In this view the name "quantum branching monad" of our monad $\mathcal{Q}$ is utterly inappropriate—it is more precisely a "classical control and quantum data monad." We shall stick to this name, however, in view of other branching monads such as $\mathcal{L}, \mathcal{P}$ and $\mathcal{D}$.

4.3. A Linear Combinatory Algebra via Categorical GoI

In the current section (§4.3) we shall elaborate on the high-level description in §2.3 and review the general definitions and results in [21] on categorical GoI, using the prototypical example $\mathbf{Pfn} \cong \mathcal{K}\ell(\mathcal{L})$ for providing some intuitions (where $\mathcal{L}$ is the lift monad given in Example 2.14). The constructions in [21], when applied to $\mathbf{Pfn} \cong \mathcal{K}\ell(\mathcal{L})$, lead to a model that is pretty close to token machines in [62, 24]. Later in §4.4, building on the categorical theory of branching in §4.1–4.2, we will observe that the Kleisli category $\mathcal{K}\ell(\mathcal{Q})$ for the quantum branching monad also constitutes an example of categorical GoI. This fact suggests that the resulting model is a "quantum variant"—one among multiple possibilities, at least—of token machine semantics.

Definition 4.7 (Retraction). Let X and Y be objects of a category $\mathbb{C}$. A *retraction* from X to Y is a pair of arrows $f : X \to Y$ and $g : Y \to X$ such that $g \circ f = \mathrm{id}_X$, that is,

$$\begin{tikzcd} X \arrow[loop left, "\mathrm{id}"] \arrow[r, bend left, "f"] & Y \arrow[l, bend left, "g"] \end{tikzcd} .$$

Such a retraction shall be denoted by $f : X \lhd Y : g$, following [21].

Throughout the current paper all the examples of retractions are in fact isomorphisms, although retractions are sufficient for the axiomatic developments.

Definition 4.8 (GoI situation). A *GoI situation* is a triple $(\mathbb{C}, F, U)$ where

- $\mathbb{C} = (\mathbb{C}, I, \otimes)$ is a traced symmetric monoidal category (TSMC), see e.g. [44, 63]. This means that $\mathbb{C}$ is a monoidal category equipped with a *trace* operator
$$\frac{X \otimes Z \xrightarrow{f} Y \otimes Z \quad \text{in } \mathbb{C}}{X \xrightarrow{\mathsf{tr}^Z_{X,Y}(f)} Y \quad \text{in } \mathbb{C}}$$
that is subject to certain equational axioms.

- $F : \mathbb{C} \to \mathbb{C}$ is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
$$\begin{array}{rll} e \ : \ FF \lhd F \ : \ e' & \qquad & \text{Comultiplication} \\ d \ : \ \mathrm{id} \lhd F \ : \ d' & & \text{Dereliction} \\ c \ : \ F \otimes F \lhd F \ : \ c' & & \text{Contraction} \\ w \ : \ K_I \lhd F \ : \ w' & & \text{Weakening} \end{array}$$
Here K_I is the constant functor into the monoidal unit I;

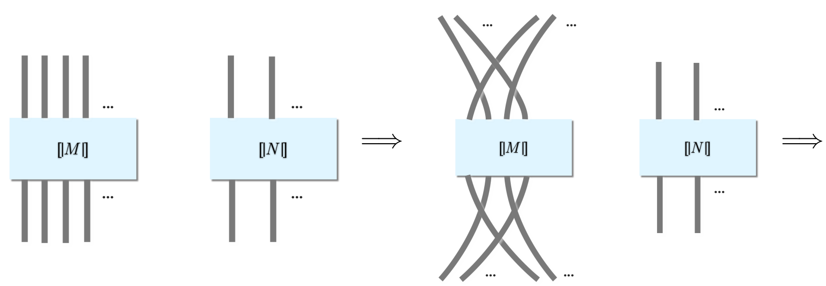
- $U \in \mathbb{C}$ is an object (called *reflexive object*), equipped with the following retractions.
$$\begin{array}{c} j \ : \ U \otimes U \lhd U \ : \ k \\ I \lhd U \\ u \ : \ FU \lhd U \ : \ v \end{array}$$

Let us try to provide some intuitions on the notion of GoI situation; in particular, on how this abstract categorical notion manages to encapsulate the essence of GoI, that is, interaction-based semantics of computation. We shall use a prototypical example of GoI situations from [21].

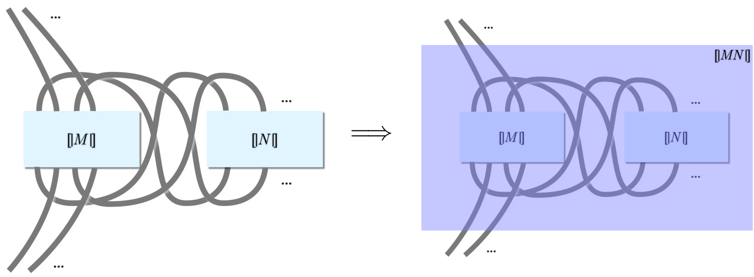
Lemma 4.9 ([21])**.** *The triple* $\bigl((\mathcal{K}\ell(\mathcal{L}), 0, +), \mathbb{N}\cdot\_\,, \mathbb{N}\bigr)$ *forms a GoI situation. Here the functor* $\mathbb{N}\cdot\_ : \mathcal{K}\ell(\mathcal{L}) \to \mathcal{K}\ell(\mathcal{L})$ *carries a set* X *to the coproduct* $\mathbb{N}\cdot X$ *of countably many copies of* X *(i.e. to the* $\mathbb{N}$*-th copower of* X*).* □

In the token-based presentation of GoI like in [62, 24], a term M in a calculus is interpreted as a *partial function* $[\![M]\!]$ from some countable set to another; let us say it is of type $[\![M]\!]\colon \mathbb{N} \nrightarrow \mathbb{N}$. Note here that we used the general notation $\nrightarrow$ for Kleisli arrows, since partial functions are nothing but Kleisli arrows for the lift monad $\mathcal{L}$. Recall the piping analogy from §4.1: in this analogy, the interpretation $[\![M]\!]$ is a piping with countably many entrances and exits. See the figure (16).

What is intriguing about GoI is how a function application MN is interpreted. In this paper we aim at exploiting GoI in deriving a categorical and denotational (hence algebraic and compositional) model—therefore we should be able to derive $[\![MN]\!]\colon \mathbb{N} \nrightarrow \mathbb{N}$ from the interpretations $[\![M]\!]\colon \mathbb{N} \nrightarrow \mathbb{N}$ and $[\![N]\!]\colon \mathbb{N} \nrightarrow \mathbb{N}$ of the constituent parts.[10] This happens as follows. Here we use the *piping* analogy in §4.1, identifying a partial function $f\colon \mathbb{N} \nrightarrow \mathbb{N}$ with a piping with countably many entrances and exits, with possible loops (a token enters and it might not come out).

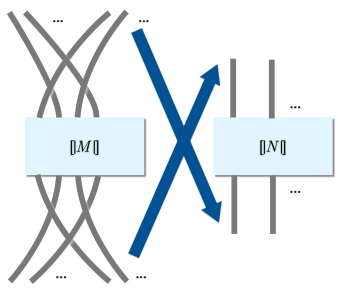


[10]It is often emphasized (e.g. in [20]) that GoI semantics is different from "denotational semantics," in that the former explicitly uses the execution formula as a semantical counterpart of β-reduction, while commonly in denotational semantics the interpretation of terms is unchanged. In this paper we follow categorical GoI [21] and abstract away from this difference.



(24)

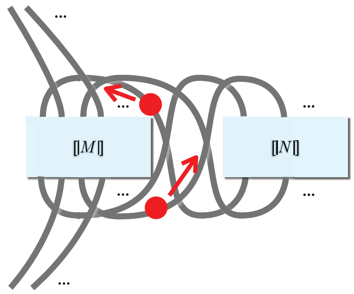
Let us go through the above process by one step after another. In the first step ($\Longrightarrow$), a bundle of countably many pipes (for the entrances of $[\![M]\!]$) is split into two bundles (left and right); and the same happens for the exits of $[\![M]\!]$. This is possible because the set of natural numbers is isomorphic to two copies of it ($\mathbb{N} + \mathbb{N} \cong \mathbb{N}$); axiomatically this is what is required of the reflexive object U in Definition 4.8. In the second step we interconnect the bundles on the right into/from $[\![M]\!]$, and the exits/entrances of $[\![N]\!]$, in the following way.



Notice that, after this second step ($\Longrightarrow$), we are now looking at a piping with countably many entrances (top-left) and countably many exits (bottom-left)—that is, a partial function $\mathbb{N} \rightharpoonup \mathbb{N}$. The third step ($\Longrightarrow$) just means that we take the piping (i.e. a partial function) thus obtained as the interpretation $[\![MN]\!]$ of the function application MN.

One can also think of the process (24) as "parallel composition and hiding," a design principle often heard in game semantics [22, 23]. Specifically, the interpretations $[\![M]\!], [\![N]\!]$ of the constituent terms operate in parallel, communicating with each other by passing a token (information is communicated by the choice of a pipe via which to pass the token, see below); and the interpretation $[\![MN]\!]$ is finally obtained by "hiding" the internal interactions between $[\![M]\!]$ and $[\![N]\!]$

(the third $\Longrightarrow$ in (24)).



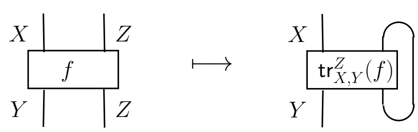
(25)

Let us also note that partiality is crucial in GoI. Even if $[\![M]\!]$ and $[\![N]\!]$ are total functions from $\mathbb{N}$ to $\mathbb{N}$, the piping obtained in (25) does not necessarily correspond to a total function: a token that enters from one of the top-left pipes can be caught in an infinite loop between $[\![M]\!]$ and $[\![N]\!]$, never reaching one of the exits (the bottom-left pipes).

As we already mentioned, a traced symmetric monoidal category (TSMC) is a main component of the notion of *GoI situation* (Def. 4.8), a categorical axiomatization of GoI given in [21]. The reason is simple, in view of the above piping and token intuitions about GoI. Recall that a symmetric monoidal category $\mathbb{C}$ being *traced* means that it is equipped with a trace operator; in our current setting where the tensor product of $\mathbb{C} = \mathcal{K}\ell(\mathcal{L})$ is given by coproduct $+$, the type of the trace operator is

$$\frac{X + Z \xrightarrow{f} Y + Z \quad \text{in } \mathcal{K}\ell(\mathcal{L})}{X \xrightarrow{\mathsf{tr}^{Z}_{X,Y}(f)} Y \quad \text{in } \mathcal{K}\ell(\mathcal{L})}\ .$$

Moreover the operator's action can be depicted as follows, using the string diagram formalism for (arrows in) monoidal categories [64].



A trace operator can be thought of as a *feedback* operator in many examples. This is also the case in our current specific instance of $\mathcal{K}\ell(\mathcal{L}) \cong \mathbf{Pfn}$. Here a trace operator can be concretely given in the following straightforward manner, exploiting partiality. Given $f\colon X + Z \nrightarrow Y + Z$, let $f_{XY}\colon X \nrightarrow Y$ be its "restriction" to X and Y, that is

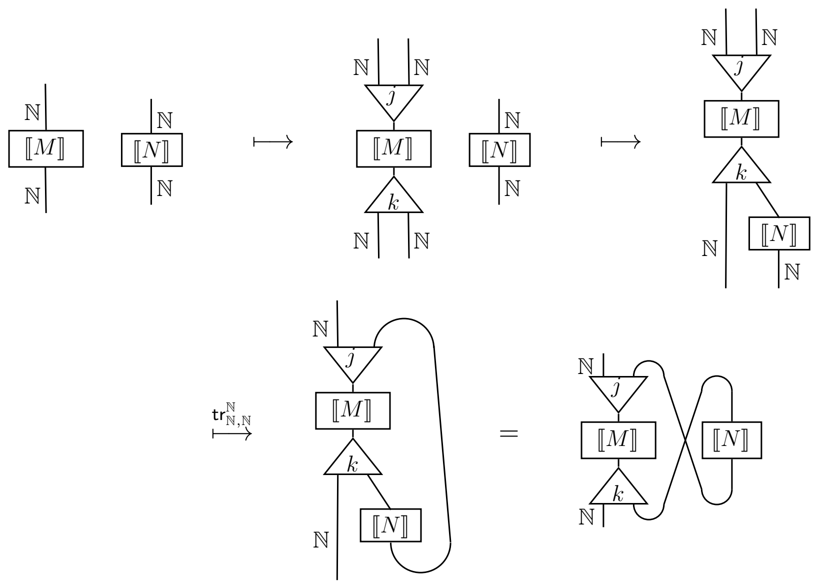
$$f_{XY}(u) \;:=\; \begin{cases} f(u) & \text{if } u \in X \text{ and } f(u) \in Y, \\ \bot \text{ (undefined)} & \text{otherwise.} \end{cases} \tag{26}$$

We define f_{XZ}, f_{ZY} and f_{ZZ} in a similar manner, and we let $\mathsf{tr}^{Z}_{X,Y}(f)\colon X \nrightarrow Y$ be

$$\mathsf{tr}^{Z}_{X,Y}(f)(x) \;:=\; f_{XY}(x) \sqcup \bigsqcup_{n \in \mathbb{N}} \big(f_{ZY} \circ f^{n}_{ZZ} \circ f_{XZ} \big)(x)\ , \tag{27}$$

where $\sqcup$ denotes supremums in the flat cpo $\mathcal{L}Y = \{\bot\} + Y$. The right-hand side informally reads: either the token immediately comes out of Y, or it comes out of Z, in which case it is fed back to the Z-entrance again. It bears notable similarity to Girard's *execution formula*, too.

This trace operator is what allows the series of transformations described in (24). In (24) we used pipes to informally depict partial functions; the same can be described formally, using string diagrams [64] for the (traced) monoidal category $(\mathcal{K}\ell(\mathcal{L}), 0, +)$, as follows.



(28)

In the above, we start with two arrows $[\![M]\!], [\![N]\!] \colon \mathbb{N} \nrightarrow \mathbb{N}$ in $\mathcal{K}\ell(\mathcal{L})$; we compose j and k, from the retraction $\mathbb{N} + \mathbb{N} \lhd \mathbb{N}$ (Def. 4.8), with $[\![M]\!]$; to it we post-compose the arrow

$$\mathbb{N} + [\![N]\!] : \quad \mathbb{N} + \mathbb{N} \nrightarrow \mathbb{N} + \mathbb{N}$$

(where the first $\mathbb{N}$ stands for the identity arrow $\mathrm{id}_{\mathbb{N}} \colon \mathbb{N} \nrightarrow \mathbb{N}$) and obtain the arrow $(\mathbb{N} + [\![N]\!]) \circ k \circ [\![M]\!] \circ j$ (the third string diagram); here we crucially exploit the trace operator tr in the category $\mathcal{K}\ell(\mathcal{L})$ and obtain the arrow

$$\mathsf{tr}^{\mathbb{N}}_{\mathbb{N},\mathbb{N}}\big((\mathbb{N} + [\![N]\!]) \circ k \circ [\![M]\!] \circ j \big) : \quad \mathbb{N} \nrightarrow \mathbb{N} \ ;$$

finally some "topological reasoning" with string diagrams yields the last equality, and the last string diagram corresponds to the last "piping" diagram in (24). This way, a trace operator plays an essential role in axiomatizing "two processes talking to each other (i.e. feeding one's answer back to the other)."

As we already discussed in §2.3, the workflow of categorical GoI [21] turns a GoI situation (Def. 4.8) into an LCA, a model of *untyped* linear λ-calculus.

Definition 4.10 (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*) $\mathsf{B}, \mathsf{C}, \mathsf{I}, \mathsf{K}, \mathsf{W}, \mathsf{D}, \delta$, and F, of A. These are required to satisfy the following equalities.

$\mathsf{B}xyz = x(yz)$	Composition, Cut
$\mathsf{C}xyz = (xz)y$	Exchange
$\mathsf{I}x = x$	Identity
$\mathsf{K}x\,!\,y = x$	Weakening
$\mathsf{W}x\,!\,y = x\,!\,y\,!\,y$	Contraction
$\mathsf{D}\,!\,x = x$	Dereliction
$\delta\,!\,x = !\,!\,x$	Comultiplication
$\mathsf{F}\,!\,x\,!\,y = !(xy)$	Monoidal functoriality

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

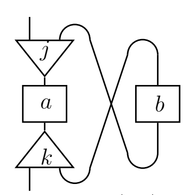
Theorem 4.11 (From GoI situations to LCAs [21])**.** *Let $(\mathbb{C}, F, U)$ be a GoI situation (Definition 4.8). Then the homset*

$$\mathbb{C}(U, U)$$

is a linear combinatory algebra (LCA). □

We shall review the proof of the previous result (from [21]) and describe the LCA structure of $\mathbb{C}(U, U)$ in some detail. Its application operator $\cdot$ is defined in the same way as what we already described for the special case of $(\mathbb{C}, F, U) = (\mathcal{K}\ell(\mathcal{L}), \mathbb{N} \cdot \_, \mathbb{N})$. That is, for each $a, b \colon U \nrightarrow U$,

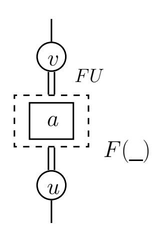
$$a{\cdot}b \;:=\; \mathsf{tr}^{U}_{U,U}\Big(\, U \otimes U \xrightarrow{j} U \xrightarrow{a} U \xrightarrow{k} U \otimes U \xrightarrow{U \otimes b} U \otimes U \,\Big) \;=$$



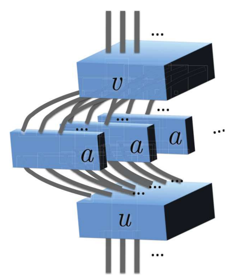
(29)

Let us now describe the $!$ operator. Its string diagram presentation, using dashed boxes and double lines for denoting application of the functor F (like in [65, 47]), is as follows.

$$!\,a \;:=\; \Big(\, U \xrightarrow{v} FU \xrightarrow{Fa} FU \xrightarrow{u} U \,\Big) \;= \qquad (30)$$

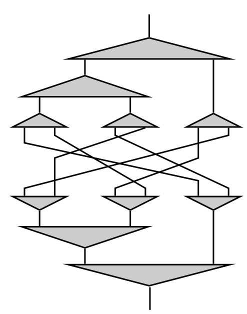


Now, for some intuition, let us instantiate the general definition to the leading example of $(\mathbb{C}, F, U) = (\mathcal{K}\ell(\mathcal{L}), \mathbb{N}\cdot\_, \mathbb{N})$. Since we have an isomorphism $\mathbb{N}\cdot\mathbb{N} \cong \mathbb{N}$, a retraction $u : \mathbb{N}\cdot\mathbb{N} \lhd \mathbb{N} : v$ is readily available; let us fix one such pair. In the string diagram in (30) a plain string represents $\mathbb{N}$ and a double line represents $\mathbb{N}\cdot\mathbb{N}$; the dashed box then represents making countably many copies of its content. Using pipes in place of strings (i.e. thinking of a string of type $\mathbb{N}$ as a bunch of countably many pipes), the diagram in (30) comes to look like the following.



Here a token moves from the top towards the bottom; the piping v divides countably many pipes into countably many bunches (each of which consists of countably many pipes), according to a fixed isomorphism $\mathbb{N}\cdot\mathbb{N} \cong \mathbb{N}$; in the middle we have a copy of a for each bunch; and finally the bunches of pipes are unified by u into a single bunch.

Let us now go on to describing LCA combinators, like B, C and I, in the LCA $\mathbb{C}(U,U)$ in Theorem 4.11. Their definitions are given in [21, §4], uniformly for any GoI situation (whether it is particle-style or wave-style), by certain string diagrams. For example, the B combinator is given by the following element of $\mathbb{C}(U,U)$.



(31)

This diagram is a string diagram in $\mathbb{C}$; the triangles denote the isomorphisms $j : U \otimes U \cong U : k$. By expanding the application operation $\cdot$ according to (29), it is not hard to see that the equation $\mathsf{B}xyz = x(yz)$ holds. See Figure 2. In fact, it is a nice (and easy) puzzle to recover the string diagram (29) from the specification $\mathsf{B}xyz = x(yz)$ of the combinator B—the (seemingly complicated and arbitrary) wiring in the middle of (31) can be deduced by working out which wire should be connected, in the end, to which wire. An important point here

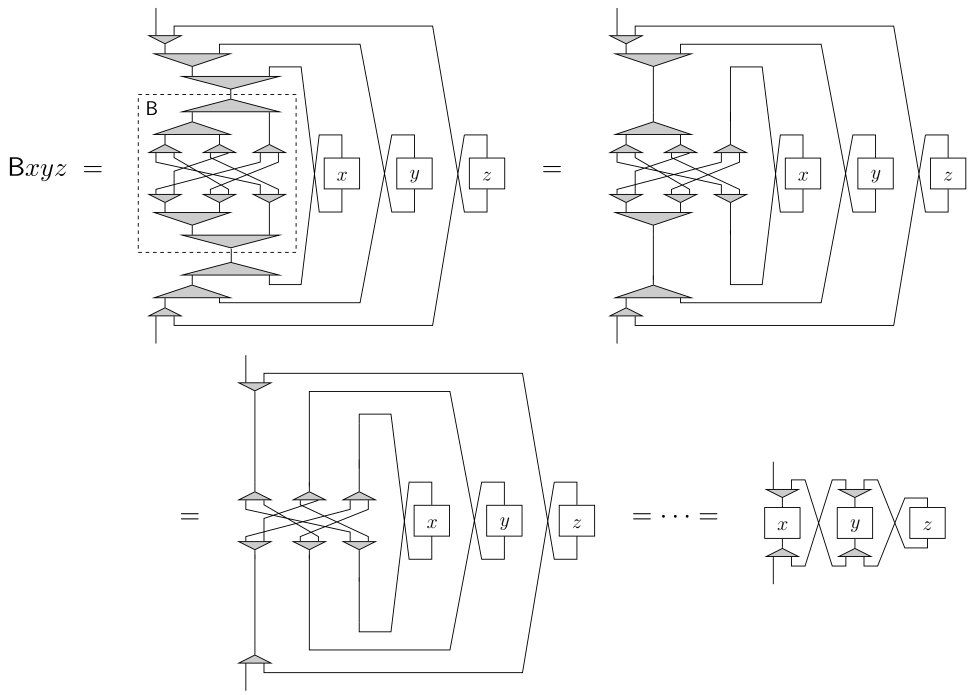
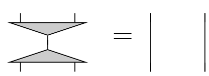


Figure 2: Proof of $\mathsf{B}xyz = x(yz)$

is that two triangles pointing to each other cancel out, that is,



since $k \circ j = \mathrm{id}$.

Remark 4.12. The definition (31) can be derived by working backwards in the reasoning in Figure 2. A more "logical" derivation is possible, too; it works as follows. We first turn the TSMC $\mathbb{C}$ into a compact closed (hence symmetric monoidal closed) category $\mathrm{Int}(\mathbb{C})$, applying the Int-construction [44] (or the GoI construction, see §2.3). The reflexive object U will then give rise to an object (U, U) in $\mathrm{Int}(\mathbb{C})$ that is equipped with a retraction $(U, U) \multimap (U, U) \lhd (U, U)$. This retraction—of the function space $(U, U) \multimap (U, U)$ in (U, U) itself—will allow the interpretation of untyped linear λ-terms over it. We then interpret $\lambda xyz.\, x(yz)$, the λ-term for the combinator B.

Remark 4.13 (wave-style GoI). In §4.3 we have relied on *particle-style* examples of GoI situations for intuitions. This leaves *wave-style* examples—where tensor products are given by products instead of coproducts, see §4.1—untouched. In fact we have preliminary observations that suggest the following correspondence: particle-style GoI situations naturally model *forward*, *state-based* description of tokens' dynamics, while wave-style ones model *backward*, *predicate-based* description of it. Here the contrast is really the one between *state-transformer* semantics and *predicate-transformer* semantics—a classic topic in theoretical computer science [66, 67] that has also proved to be relevant in quantum dynamics (the Schrödinger picture and the Heisenberg one; see e.g. [68]). Details are yet to be worked out.

4.4. Categorical GoI Instantiated to a Quantum Setting

In §4.1 we observed that "particle-style" examples of GoI situations in [21] allow a uniform treatment as Kleisli categories. This generalizes to the quantum branching monad $\mathcal{Q}$ in §4.2.

Theorem 4.14. *The triple* $\big((\mathcal{K}\ell(\mathcal{Q}), 0, +), \mathbb{N}\cdot\_\,, \mathbb{N}\big)$ *forms a GoI situation. Here the functor* $\mathbb{N}\cdot\_ : \mathcal{K}\ell(\mathcal{Q}) \to \mathcal{K}\ell(Q)$ *carries* X *to the coproduct* $\mathbb{N}\cdot X$ *of* $\mathbb{N}$*-many copies of* X *(i.e.* $\mathbb{N}$*-th copower of* X*).*

Proof. The main challenge—namely, if $(\mathcal{K}\ell(\mathcal{Q}), 0, +)$ is traced or not—is already answered in Theorem 4.5. Its trace operator is given much like for the category $\mathcal{K}\ell(\mathcal{L}) \cong \mathbf{Pfn}$ of sets and partial functions. Namely, given $f\colon X+Z \nrightarrow Y+Z$ in $\mathcal{K}\ell(\mathcal{Q})$: its "restrictions" f_{XY}, f_{XZ}, f_{ZY} and f_{ZZ} are defined much like in (26); and we use Girard's execution formula

$$\mathsf{tr}^{Z}_{X,Y}(f)(x) \;:=\; f_{XY}(x) + \sum_{n\in\mathbb{N}} \big(\, f_{ZY} \odot f^{n}_{ZZ} \odot f_{XZ}\,\big)(x)$$

to define the trace operator. Notice similarity to the formula (27).

The only nontrivial part that remains is to show that $\mathbb{N}\cdot\_$ preserves traces. Since the trace operator in $\mathcal{K}\ell(\mathcal{Q})$ can be described using Girard's execution formula (much like in (26) for $\mathcal{K}\ell(\mathcal{L})$, due to the results in [33, 59]), we can use a lemma that is similar to [21, Lemma 5.1]. $\square$

By Theorem 4.11 (that is from [21]) we obtain the following LCA. It will be denoted by $A_{\mathcal{Q}}$ and used in the rest of the paper.

Theorem 4.15 (The quantum LCA $A_{\mathcal{Q}}$)**.** *The homset*

$$A_{\mathcal{Q}} \;:=\; \mathcal{K}\ell(\mathcal{Q})(\mathbb{N},\mathbb{N})$$

is a linear combinatory algebra (LCA). $\square$

The LCA structure of $A_{\mathcal{Q}}$ (the operators $\cdot, !$ and combinators like B) can be described very much like for $\mathcal{K}\ell(\mathcal{L})$; we already described the latter in §4.3. In particular the piping analogy is still valid, except for the difference in the notion of "branching" (possibly diverging vs. based on quantum operations) we explained in §4.2.

The following special property is shared by the LCAs that arise from particle-style GoI situations. For a proof see [32, §2.2].

Proposition 4.16. *The LCA* $A_{\mathcal{Q}}$ *in Theorem 4.15 is* affine*: it has the full* K *combinator such that* $\mathsf{K}xy = x$. $\square$

4.5. A Linear Category via Realizability

According to our workflow in Figure 1, the next step is to employ the (linear) *realizability* technique [32, 35] and turn an LCA (an *untyped* model) into a linear category (a *typed* model). Here in §4.5 we describe how that happens, focusing on the constructions and observations already described in [32, 35]. Our specific linear category $\mathbf{PER}_{\mathcal{Q}}$ has some additional properties that result from the way we construct it and, at the same time, are exploited for interpreting some features of Hoq that go beyond standard linear λ-calculi. These additional features will be described separately, later in §5.1.

Although a very brief introduction to realizability is found in §2.4, the current paper would hardly be enough in providing good intuitions behind the technical constructions. Unfamiliar readers are referred to [34, 69] for realizability in general in categorical settings, and to [32, 35] for linear realizability in particular. In what follows, for intuitions, it can be helpful to imagine Kleene's first combinatory algebra—where elements are natural numbers and application $a \cdot b$ is defined by the outcome of the a-th recursive function applied to the b-th tuple of natural numbers—in place of the LCA $A_{\mathcal{Q}}$.

Definition 4.17 (PER). A *partial equivalence relation (PER)* over $A_{\mathcal{Q}}$ is a symmetric and transitive relation X on the set $A_{\mathcal{Q}}$. The *domain* of a PER $|X|$ is defined by

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

where the last equality follows from symmetry and transitivity of X. When restricted to its domain $|X|$, X is an equivalence relation; therefore X can be thought of as a subset $|X| \subseteq A_{\mathcal{Q}}$, suitably quotiented.

Intuitively: X is a "datatype," each element of which is represented by some elements of $A_{\mathcal{Q}}$; not every element of $A_{\mathcal{Q}}$ represent an entity of X; the set of those elements which do is the domain $|X| \subseteq A_{\mathcal{Q}}$; and finally the equivalence relation X (restricted to the domain $|X|$) designates which elements of $A_{\mathcal{Q}}$ represent the same entity of X.

Definition 4.18 (The category $\mathbf{PER}_{\mathcal{Q}}$). PERs over the LCA $A_{\mathcal{Q}}$ form a category; it is denoted by $\mathbf{PER}_{\mathcal{Q}}$. Its object X is a PER over $A_{\mathcal{Q}}$. Its arrow $X \to Y$ is defined to be an equivalence class of the PER

$$X \multimap Y := \bigl\{\, (c,c') \mid (x,x') \in X \Rightarrow (cx, c'x') \in Y \,\bigr\} \;\;, \tag{32}$$

where $cx = c \cdot x$ denotes $c \in A_{\mathcal{Q}}$ applied to $x \in A_{\mathcal{Q}}$ via the applicative structure $\cdot$ of $A_{\mathcal{Q}}$. We denote by $[c]$ the equivalence class in $X \multimap Y$ to which $c \in A_{\mathcal{Q}}$ belongs. That is, $[c]$ is an arrow that is "realized by the code c."

Identity arrows and composition of arrows in $\mathbf{PER}_{\mathcal{Q}}$ are defined as usual (see e.g. [32, 35]). Explicitly,

$$\mathrm{id}_X \;:=\; [\mathsf{I}] \;;\quad \text{and} \quad [d] \circ [c] \;:=\; [\mathsf{B}dc] \quad \text{for } X \xrightarrow{[c]} Y \xrightarrow{[d]} Z \text{ in } \mathbf{PER}_{\mathcal{Q}}.$$

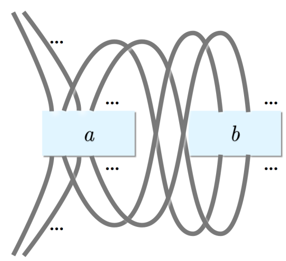
Observe that, for the latter, we indeed have

$$\big(\text{the code of } [d] \circ [c]\big) \cdot x \;=\; \mathsf{B}dcx \;=\; d(cx) \quad \in |Z| \ .$$

Note also that we use $\circ$ for composition of arrows in $\mathbf{PER}_{\mathcal{Q}}$. This is to be distinguished from $\cdot$ (for application in the LCA $A_{\mathcal{Q}}$, that is often omitted); and from $\odot$ (for composition of arrows in $\mathcal{K}\ell(\mathcal{Q})$, like in (23).)

We elaborate further on the definition (32). Its domain $|X \multimap Y|$ is easily seen to be the set of $c \in A_{\mathcal{Q}}$ such that $(x, x') \in X$ implies $(cx, cx') \in Y$. This requirement is that the function $[c] : |X|/X \to |Y|/Y, [x] \mapsto [cx]$ is *well-defined*: if $(x, x') \in X$, that is, if $x, x' \in A_{\mathcal{Q}}$ "represent" the same entity in the PER X, then applying the code c to both elements must result in the elements $cx, cx' \in A_{\mathcal{Q}}$ that again represent the same entity in the PER Y. Furthermore, the PER $X \multimap Y$ identifies c and c' such that $(cx, c'x) \in Y$ for each $|X|$. This is the *extensionality* of the functions of the type $|X|/X \to |Y|/Y$. The situation is much like in recursion theory, where different natural numbers can be "codes" of the same recursive function.

In (32) cx and $c'x'$ are short for $c \cdot x$ and $c' \cdot x'$, respectively. Recall that $\cdot$ here is the application operator in the LCA $A_{\mathcal{Q}} = \mathcal{K}\ell(\mathcal{Q})(\mathbb{N}, \mathbb{N})$, that is defined by

$a \cdot b \;:=$ 

in terms of pipes. See §4.3; to repeat some of the intuitions provided there, we imagine a token that takes one of the entrances (top-left pipes) and comes out of one of the exits (bottom-left pipes). The token exhibits certain branching: in the example of $\mathcal{K}\ell(\mathcal{L}) \cong \mathbf{Pfn}$ it was simply possible nontermination; in the current setting of quantum branching, a token carries a quantum state and quantum measurements on the quantum state give rise to probabilistic branching over different exits that the token takes (§4.2).

From now on we describe some properties and features of the category $\mathbf{PER}_{\mathcal{Q}}$. It is a linear category, making it a categorical model for usual typed linear λ-calculi. This fact, with some concrete constructions of the linear category structure, is described in short. The category $\mathbf{PER}_{\mathcal{Q}}$ also exhibits some additional properties. This is for two principal reasons: 1) due to its construction via realizability (Definition 4.18); and 2) because the underlying LCA $A_{\mathcal{Q}}$ is affine (rather than linear) and supports full weakening (Proposition 4.16). These additional features, separately presented later in §5.1, will be exploited for interpreting various features of the language Hoq.

Let us begin with some preparations.

Notation 4.19. In what follows, an element of the LCA $A_{\mathcal{Q}}$ is often designated by an untyped linear λ-term. This is justified by combinatory completeness of LCAs. It is claimed e.g. in [32, 70] and not hard to establish; an explicit proof is found in [71].

Definition 4.20. We introduce the following additional combinators in $A_{\mathcal{Q}}$; via combinatory completeness, they stand for certain terms composed of the basic combinators in Definition 4.10, together with the full K combinator in Proposition 4.16.

$$\begin{array}{ll}
\mathsf{P} := \lambda xyz.zxy & \text{Pairing} \\
\bar{\mathsf{K}} := \mathsf{KI} & \text{Weakening, } \bar{\mathsf{K}}xy = y \\
\mathsf{P_l} := \lambda w.w\mathsf{K} & \text{Left Projection, } \mathsf{P_l}(\mathsf{P}xy) = x \\
\mathsf{P_r} := \lambda w.w\bar{\mathsf{K}} & \text{Right Projection, } \mathsf{P_r}(\mathsf{P}xy) = y
\end{array}$$

It is easy to see that the pairing in $\mathbf{PER}_{\mathcal{Q}}$ is *extensional*: $\mathsf{P}xy = \mathsf{P}x'y'$ implies $x = x'$ and $y = y'$.

Theorem 4.21 ($\mathbf{PER}_{\mathcal{Q}}$ as a linear category)**.** *The category* $\mathbf{PER}_{\mathcal{Q}}$ *is a* linear category *[30, 31], equipped with a symmetric monoidal structure* $(\mathrm{I}, \boxtimes)$ *and a so-called* linear exponential comonad $!$. *The latter means that* $!$ *is a symmetric monoidal comonad, with natural transformations*

$$\begin{array}{rlrl}
\mathsf{der}: & !X \to X \ , & \delta: & !X \to !!X \ , \\
\varphi: & !X \boxtimes !Y \to !(X \boxtimes Y) \ , & \varphi': & \mathrm{I} \to !\mathrm{I} \ ,
\end{array} \tag{33}$$

that is further equipped with monoidal natural transformations

$$\mathsf{weak}: \ !X \to \mathrm{I} \quad \textit{and} \quad \mathsf{con}: \ !X \to !X \boxtimes !X \ , \tag{34}$$

subject to certain additional conditions (see [30, 31]).

We use the symbol $\boxtimes$ for the monoidal product in $\mathbf{PER}_{\mathcal{Q}}$; it is distinguished from the tensor product of quantum states denoted by $\otimes$. See §3.3 for the discussion on this issue.

Proof. The result is due to [32, Theorem 2.1]. For later use, we shall explicitly describe some of the structures of $\mathbf{PER}_{\mathcal{Q}}$.

The category $\mathbf{PER}_{\mathcal{Q}}$ is a symmetric monoidal closed category with respect to the following operations.

$$\begin{array}{l}
X \boxtimes Y := \left\{ (\mathsf{P}xy, \mathsf{P}x'y') \,\middle|\, (x,x') \in X \wedge (y,y') \in Y \right\} \ , \\
\mathrm{I} := \{(\mathsf{I},\mathsf{I})\} \ , \qquad X \multimap Y := (\text{the same as (32)}) \ .
\end{array}$$

Here P and I are combinators from Definitions 4.10 and 4.20. The operations' action on arrows is defined in a straightforward manner. For example, given $[c_1] : X_1 \to Y_1$ and $[c_2] : X_2 \to Y_2$ in $\mathbf{PER}_{\mathcal{Q}}$,

$$[c_1] \boxtimes [c_2] := \big[\lambda w.w\big(\lambda uv.\mathsf{P}(c_1 u)(c_2 v)\big)\big] \ : \quad X_1 \boxtimes X_2 \longrightarrow Y_1 \boxtimes Y_2 \ .$$

With this definition we indeed have, for $\mathsf{P}x_1x_2 \in |X_1 \boxtimes X_2|$,

$$\big(\text{the code of } [c_1] \boxtimes [c_2]\big) \cdot (\mathsf{P}x_1x_2) \;=\; \big[\,\mathsf{P}(c_1x_1)(c_2x_2)\big] \quad \in\; Y_1 \boxtimes Y_2 \ .$$

The linear exponential comonad ! (see [32]) is given as follows, via the ! operation on the LCA $A_{\mathcal{Q}}$:

$$!\,X := \{(!\,x, !\,x') \mid (x, x') \in X\} \ , \quad ![c] := [\mathsf{F}(!\,c)] \ . \tag{35}$$

In particular, the use of the combinator F in the latter ensures the type $![c]\colon\ !\,X \to\ !\,Y$ (in $\mathbf{PER}_{\mathcal{Q}}$) for $c\colon X \to Y$ (in $\mathbf{PER}_{\mathcal{Q}}$): indeed, for any $x \in |X|$,

$$\big(\text{the code of } ![c]\big) \cdot (!\,x) \;=\; \mathsf{F}(!\,c)(!\,x) \;=\; !(cx) \quad \in\; Y \ .$$

The natural transformations $\mathsf{der}, \delta, \varphi, \varphi', \mathsf{weak}$ and con that accompany a linear exponential comonad (see (33–34)) are concretely given as follows.

$$\begin{array}{rll} \mathsf{der} & := [\mathsf{D}] & :\ !\,X \to X \\ \delta & := [\updelta] & :\ !\,X \to\, !\,!\,X \\ \varphi & := \big[\lambda w.w(\lambda uv.\mathsf{F}(\mathsf{F}(!\,\mathsf{P})u)v)\big] & :\ !\,X \boxtimes !\,Y \to !(X \boxtimes Y) \\ \varphi' & := \big[\lambda w.w(!\,\mathsf{I})\big] & :\ \mathrm{I} \to\, !\,\mathrm{I} \\ \mathsf{weak} & := [\mathsf{KI}] & :\ !\,X \to \mathrm{I} \\ \mathsf{con} & := [\mathsf{WP}] & :\ !\,X \to\, !\,X \boxtimes !\,X \end{array} \tag{36}$$

Recall $[c]$ denotes an arrow in $\mathbf{PER}_{\mathcal{Q}}$ that is realized by the code $c \in A_{\mathcal{Q}}$. □

Note that the constructions in the previous proof are the standard ones from [32] that work for any LCA.

5. Interpretation of Hoq

We now present our interpretation of Hoq in the category $\mathbf{PER}_{\mathcal{Q}}$. We have seen in Theorem 4.21 that the category is a linear category, hence models (standard) linear λ-calculi. See e.g. [47].

However, the specific calculus Hoq calls for some extra features. Firstly, the linear exponential comonad ! on $\mathbf{PER}_{\mathcal{Q}}$ should be idempotent ($!\,!\,X \overset{\cong}{\Rightarrow}\ !\,X$). This is because in Hoq we chose to implicitly track linearity by subtyping <:—in contrast to explicit tracking by constructs like `derelict` in standard linear λ-calculi. See §3.3. This issue is addressed in §5.1. Secondly, recursion in Hoq requires suitable cpo structures in the model. In the current style of realizability models the notion of *admissibility* is a standard vehicle; this and some related issues are addressed in §5.2. Thirdly we introduce some quantum mechanical constructs in $\mathbf{PER}_{\mathcal{Q}}$ for interpreting constants of Hoq; see §5.3. Finally, we need a strong monad T on $\mathbf{PER}_{\mathcal{Q}}$ for the probabilistic effect that arises inevitably in quantum computation (more specifically through measurements). In fact we will use for T a continuation monad $(\_ \multimap R) \multimap R$ with the result type R described as a final coalgebra; see §5.4. The actual interpretation of Hoq in $\mathbf{PER}_{\mathcal{Q}}$ is presented in §5.5.

The proofs for §5 are deferred to Appendix D.

5.1. Additional Structures of $\mathbf{PER}_\mathcal{Q}$

We go on to study some additional structures that are available in $\mathbf{PER}_\mathcal{Q}$. These will be exploited later for interpreting various features of Hoq, such as subtyping and coproduct types.

Recall that in $A_\mathcal{Q}$ (that is an *affine* LCA) a full weakening combinator K is available (Proposition 4.16).

Lemma 5.1 (I is terminal)**.** *The monoidal unit* I *is terminal (i.e. final) in* $\mathbf{PER}_\mathcal{Q}$*, with a unique arrow* $\mathsf{weak} : X \to \mathrm{I}$ *given by*[11]

$$\mathsf{weak} \;:=\; [\mathsf{KI}] \;. \qquad \square$$

Lemma 5.2 (Binary (co)products in $\mathbf{PER}_\mathcal{Q}$)**.** *The category* $\mathbf{PER}_\mathcal{Q}$ *has binary products* $\times$ *and binary coproducts* $+$*. Products are realized by a CPS-like encoding.*

$$\begin{aligned}
X \times Y &:= \left\{ \left(\mathsf{P}k_1(\mathsf{P}k_2 u),\, \mathsf{P}k_1'(\mathsf{P}k_2' u')\right) \,\middle|\, (k_1 u, k_1' u') \in X \wedge (k_2 u, k_2' u') \in Y \right\} \;, \\
X + Y &:= \left\{ (\mathsf{PK}x, \mathsf{PK}x') \,\middle|\, (x, x') \in X \right\} \cup \left\{ (\mathsf{P}\bar{\mathsf{K}}y, \mathsf{P}\bar{\mathsf{K}}y') \,\middle|\, (y, y') \in Y \right\} \;.
\end{aligned}$$

Their accompanying structures are defined in a straightforward manner. For example, the projection maps are concretely as follows.

$$\begin{array}{rcllcl}
\pi_1 & = & \left[\lambda w.w(\lambda kv.v(\lambda lu.ku))\right] & : & X \times Y \longrightarrow X \;; \\
\pi_2 & = & \left[\lambda w.w(\lambda kv.v(\lambda lu.lu))\right] & : & X \times Y \longrightarrow Y \;.
\end{array} \qquad (37)$$

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

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

Lemma 5.3. *The following canonical isomorphisms are available in any linear category with binary (co)products, hence in* $\mathbf{PER}_\mathcal{Q}$*.*

$$\begin{array}{rcl}
!(X \times Y) & \cong & !\,X \boxtimes !\,Y \;, \\
(X + Y) \boxtimes Z & \cong & X \boxtimes Z + Y \boxtimes Z \;, \\
\mathrm{I} \multimap X & \cong & X \;, \\
(X + Y) \multimap Z & \cong & (X \multimap Z) \times (Y \multimap Z) \;.
\end{array} \qquad (38)$$

Additionally, in $\mathbf{PER}_\mathcal{Q}$*, we have the following canonical isomorphisms.*

$$\begin{array}{rclcrcl}
!(X + Y) & \xrightarrow{\cong} & !\,X + !\,Y & & & & \\
\mathsf{der}_{!\,X} :\ !\,!\,X & \xrightarrow{\cong} & !\,X \ : \delta & & !\,\mathsf{der}_X :\ !\,!\,X & \xrightarrow{\cong} & !\,X \ : \delta \\
!(X \boxtimes Y) & \xrightarrow{\cong} & !\,X \boxtimes !\,Y \ : \varphi & & \mathsf{weak} :\ !\,\mathrm{I} & \xrightarrow{\cong} & \mathrm{I} \ : \varphi'
\end{array} \qquad (39)$$

Here the arrows $\mathsf{der}, \delta, \varphi, \varphi'$ *are from Theorem 4.21. Therefore* ! *on* $\mathbf{PER}_\mathcal{Q}$ *is idempotent and strong monoidal; it also preserves coproducts.*

[11] We shall use the same notation, weak, for both the unique arrow $X \to \mathrm{I}$ and a structure morphism for a linear exponential comonad $!\,X \to \mathrm{I}$ (Theorem 4.21). They are indeed the same arrow $[\mathsf{KI}]$.

Proof. The ones in (38) are standard; see [72, §2.1.2]. The ones in (39) also hold in any $\mathbf{PER}_A$ with an affine LCA A. For the first (! distributes over +), from right to left one takes $[!\,\kappa_\ell, !\,\kappa_r]$ where κ_ℓ, κ_r are coprojections;[12] from left to right one can take

$$[a] \;:\; !(X+Y) \longrightarrow !\,X + !\,Y \quad \text{with} \quad a := \lambda w.\mathsf{WP}w\big(\lambda uv.\mathsf{P}(\mathsf{D}u\mathsf{K})(\mathsf{F}(!\,\mathsf{P_r})v)\big) \;;$$

indeed it is straightforward to see that $a(!(\mathsf{PK}x)) = \mathsf{PK}(!\,x)$ and $a(!(\mathsf{P\bar{K}}x)) = \mathsf{P\bar{K}}(!\,x)$ for the above a. The second line of (39) (! is idempotent) follows immediately from the definitions of der and δ in (36). For example,

$$\begin{aligned}(!\,\mathsf{der}_X)(!\,!\,x) &= (![\mathsf{D}])(!\,!\,x)\\ &= \mathsf{F}(!\,\mathsf{D})(!\,!\,x) \quad \text{by (35), def. of !'s action on arrows}\\ &= !(\mathsf{D}\,!\,x) = !\,x \;\;.\end{aligned}$$

For the third line (! distributes over $\boxtimes$), an inverse of φ can be given by the following composite, exploiting that I is terminal (Lemma 5.1).

$$!(X\boxtimes Y) \xrightarrow{\mathsf{con}} !(X\boxtimes Y)\boxtimes !(X\boxtimes Y) \xrightarrow{\mathsf{weak}} !(X\boxtimes \mathrm{I})\boxtimes !(\mathrm{I}\boxtimes Y) \xrightarrow{\cong} !\,X\boxtimes !\,Y$$

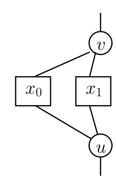
This concludes the proof. □

One consequence of the last result is that we have $!(X\boxtimes Y) \cong !(X\times Y)$ in $\mathbf{PER}_{\mathcal{Q}}$. As stated in the proof, this is true in $\mathbf{PER}_A$ for any *affine* LCA A.

We go ahead and show that the category $\mathbf{PER}_{\mathcal{Q}}$ has countable limits and colimits.

Definition 5.4 (Combinators $(x_i)_{i\in\mathbb{N}}$, D_i). Let $x_0, x_1, \ldots \in A_{\mathcal{Q}}$. We define the element $(x_i)_{i\in\mathbb{N}} \in A_{\mathcal{Q}}$ by:

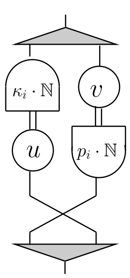
$$(x_i)_{i\in\mathbb{N}} \;:=\; \Big(\,\mathbb{N} \overset{v}{\longrightarrow\!\!\!\!\!\!\!+}\; \mathbb{N}\cdot\mathbb{N} \overset{\coprod_i x_i}{\longrightarrow\!\!\!\!\!\!\!+}\; \mathbb{N}\cdot\mathbb{N} \overset{u}{\longrightarrow\!\!\!\!\!\!\!+}\; \mathbb{N}\,\Big) \quad = $$



where $u : \mathbb{N}\cdot\mathbb{N} \cong \mathbb{N} : v$ are (fixed) isomorphisms in Theorem 4.15.

For each $i \in \mathbb{N}$, we define an element $\mathsf{D}_i \in A_{\mathcal{Q}}$ by

$$\mathsf{D}_i \;:=\; \left(\begin{array}{l}\mathbb{N} \overset{k}{\longrightarrow\!\!\!\!\!\!\!+}\; \mathbb{N}+\mathbb{N} \overset{\kappa_i\cdot\mathbb{N}+v}{\longrightarrow\!\!\!\!\!\!\!+}\; \mathbb{N}\cdot\mathbb{N}+\mathbb{N}\cdot\mathbb{N}\\ \qquad \overset{u+p_i\cdot\mathbb{N}}{\longrightarrow\!\!\!\!\!\!\!+}\; \mathbb{N}+\mathbb{N} \overset{[\kappa_r,\kappa_\ell]}{\longrightarrow\!\!\!\!\!\!\!+}\; \mathbb{N}+\mathbb{N} \overset{j}{\longrightarrow\!\!\!\!\!\!\!+}\; \mathbb{N}\end{array}\right) \quad = $$



.

[12] Note that we are using square brackets [_] to denote both: equivalence classes modulo PERs; and cotupling of arrows (like in $[!\,\kappa_\ell, !\,\kappa_r]$). We hope this does not lead to much confusion: the two usages have different arities.

Here κ_i (for $i \in \mathbb{N}$), κ_ℓ, κ_r (for "left" and "right") are coprojections; and $p_i : \mathbb{N} \nrightarrow 1$ is a "projection" map in [33] (see also Theorem Appendix C.5) that satisfies

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

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

Lemma 5.5. *We have $\mathsf{D}_j \cdot (x_i)_{i\in\mathbb{N}} = x_j$. Here $\cdot$ denotes the applicative structure of $A_{\mathcal{Q}}$ (Theorem 4.15).*

Proof. By easy manipulation of string diagrams, the claim boils down to the equality

$$\Bigl(\ \mathbb{N} \overset{\kappa_j\cdot\mathbb{N}}{\nrightarrow} \mathbb{N}\cdot\mathbb{N} \overset{\coprod_i x_i}{\nrightarrow} \mathbb{N}\cdot\mathbb{N} \overset{p_j\cdot\mathbb{N}}{\nrightarrow} \mathbb{N}\ \Bigr) \;=\; x_j \ .$$

This follows from the fact that

$$(\coprod_i x_i) \odot (\kappa_j \cdot \mathbb{N}) = \bigl[\, (\kappa_i \cdot \mathbb{N}) \odot x_i \,\bigr]_i \odot (\kappa_j \cdot \mathbb{N}) = \kappa_j \cdot \mathbb{N} \odot x_j$$

and that $p_j \odot \kappa_j = \mathrm{id}_1$. □

Proposition 5.6. *The category $\mathbf{PER}_{\mathcal{Q}}$ 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 $[c], [d] : X \rightrightarrows Y$ in $\mathbf{PER}_{\mathcal{Q}}$, let

$$\begin{aligned} E \ &:= \ \bigl\{\, (x, x') \,\big|\, (x, x') \in X \wedge (cx, dx') \in Y \,\bigr\} \ ; \text{ and} \\ C \ &:= \ \bigl(\text{the symmetric and transitive closure of } \bigl\{\, (y, y') \,\big|\, (y, y') \in Y \,\bigr\} \cup \bigl\{\, (cx, dx') \,\big|\, (x, x') \in X \,\bigr\}\bigr) \ . \end{aligned}$$

Then it is straightforward to see that $[\mathsf{I}] : E \to X$ and $[\mathsf{I}] : Y \to C$—where I is the identity combinator (Definition 4.10)—are an equalizer and a coequalizer of $[c]$ and $[d]$, respectively.

It suffices to show that $\mathbf{PER}_{\mathcal{Q}}$ has countable products and coproducts (since (co)products and (co)equalizers give all (co)limits). Given a countable family $(X_i)_{i\in\mathbb{N}}$ of objects of $\mathbf{PER}_{\mathcal{Q}}$, we use the constructs in Definition 5.4 and define

$$\begin{aligned} \prod_{i\in\mathbb{N}} X_i \ &:= \ \bigl\{\, \bigl(\mathsf{P}(k_i)_{i\in\mathbb{N}} u,\ \mathsf{P}(k'_i)_{i\in\mathbb{N}} u'\bigr) \,\big|\, (k_i u, k'_i u') \in X_i \text{ for each } i \in \mathbb{N} \,\bigr\} \ ; \\ \pi_i \ &:= \ \bigl[\, \lambda w.w(\lambda vu.(\mathsf{D}_i v)u) \,\bigr] \quad : \quad \prod_{i\in\mathbb{N}} X_i \longrightarrow X_i \ . \end{aligned}$$

Then, given a family $\bigl([c_i] : Y \to X_i\bigr)_{i\in\mathbb{N}}$ of arrows, its tupling can be given by

$$\bigl\langle [c_i] \bigr\rangle_{i\in\mathbb{N}} \ := \ \bigl[\lambda y.\mathsf{P}(c_i)_{i\in\mathbb{N}}\, y\bigr] \quad : \quad Y \longrightarrow \prod_i X_i \ .$$

The uniqueness of such a tupling directly follows from the definition of the PER $\prod_{i\in\mathbb{N}} X_i$.

On coproducts, we define

$$\coprod_{i\in\mathbb{N}} X_i \;:=\; \left\{\, (\mathsf{PD}_i x_i, \mathsf{PD}_i x_i') \,\middle|\, i\in\mathbb{N}, (x_i, x_i')\in X_i \,\right\} \;;$$

$$\kappa_i \;:=\; \left[\,\mathsf{PD}_i\,\right] \quad : \quad X_i \longrightarrow \coprod_{i\in\mathbb{N}} X_i \;.$$

Given a family $\big([c_i] : X_i \to Y \big)_{i\in\mathbb{N}}$ of arrows, its cotupling can be given by

$$\big[\,[c_i]\,\big]_{i\in\mathbb{N}} \;:=\; \big[\lambda w.w(\lambda dx.d(c_i)_{i\in\mathbb{N}}\, x)\big] \quad : \quad \coprod_{i} X_i \longrightarrow Y \;.$$

This concludes the proof. □

Remark 5.7. Although we have focused on the specific linear category $\mathbf{PER}_{\mathcal{Q}}$, 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 (39) hold in any $\mathbf{PER}_A$ with an affine LCA A. This makes such categories $\mathbf{PER}_A$ suitable for modeling linear λ-calculi with implicit linearity tracking.

Another point is about the constructions in Definition 5.4, Lemma 5.5 and Proposition 5.6. It is not hard to see that these are all possible in the category $\mathbf{PER}_{A_B}$, where the LCA A_B is obtained via categorical GoI [21, Proposition 4.2] from the Kleisli category $\mathcal{K}\ell(B)$ for any "branching monad" like $B = \mathcal{L}, \mathcal{P}, \mathcal{D}$ and $\mathcal{Q}$ (see §2.2).

5.2. Order Enrichment and Further Additional Constructs in $\mathbf{PER}_{\mathcal{Q}}$

Here in §5.2 we describe another additional construct—namely an alternative pairing combinator $\dot{\mathsf{P}}$—in our categorical model $\mathbf{PER}_{\mathcal{Q}}$. What we describe here are mostly concerned about order/cpo structures, which we exploit for the purpose of interpreting recursion in Hoq.

Let us first note that (the underlying set of) the LCA $A_{\mathcal{Q}}$ is equipped with an ω-CPO structure $\sqsubseteq$. This is because: $A_{\mathcal{Q}} = \mathcal{K}\ell(\mathcal{Q})(\mathbb{N},\mathbb{N})$ (Theorem 4.15); and the category $\mathcal{K}\ell(\mathcal{Q})$ is ω-$\mathbf{CPO}$ enriched (Theorem 4.5). Recalling from Definition 4.4, the order on $A_{\mathcal{Q}}$ is (a suitable pointwise extension of) Löwner partial order (Definition 2.4). For the record:

Lemma 5.8 ($A_{\mathcal{Q}}$ is an ω-CPO)**.** *The set $A_{\mathcal{Q}}$ is an ω-CPO with the smallest element $\perp$. Furthermore:*

1. *the application operator $\cdot : A_{\mathcal{Q}}^2 \to A_{\mathcal{Q}}$ and the $!$ operator are continuous; and*
2. *application $\cdot$ is left strict, that is, $\perp \cdot a = \perp$ for each $a \in A_{\mathcal{Q}}$.* □

This order structure on $A_{\mathcal{Q}}$, unfortunately, does *not* give rise to orders on $\mathbf{PER}_{\mathcal{Q}}$ in a categorically structured manner. Certainly not so nicely as for the Kleisli category $\mathcal{K}\ell(\mathcal{Q})$—we do not see any straightforward way to make the category $\mathbf{PER}_{\mathcal{Q}}$ ω-$\mathbf{CPO}$ enriched. To see it, consider the following notion of *admissibility* of PERs—this is a standard vehicle for interpreting recursion in realizability models (see e.g. [73, 74]).

Definition 5.9 (Admissible PER). A PER $U \in \mathbf{PER}_{\mathcal{Q}}$ is said to be *admissible* if:

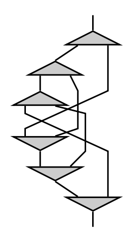
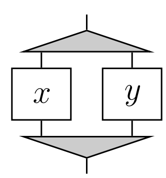
- (strictness) $(\bot, \bot) \in U$ for the least element $\bot \in A_{\mathcal{Q}}$; and
- (inductiveness) $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$, $y_0 \sqsubseteq y_1 \sqsubseteq \cdots$ and $(x_i, y_i) \in U$ for each $i \in \mathbb{N}$ imply $(\sup_i x_i, \sup_i y_i) \in U$.

The trickiness of order structures on $\mathbf{PER}_{\mathcal{Q}}$ is exemplified by the fact that admissibility is not preserved by isomorphisms in $\mathbf{PER}_{\mathcal{Q}}$.

Example 5.10 (Admissibility not preserved by isomorphisms). It is easy to see that we have an isomorphism $\mathrm{I} = \{(\mathsf{I}, \mathsf{I})\} \cong \{(\bot, \bot)\} =: \mathsf{Bt}$ in $\mathbf{PER}_{\mathcal{Q}}$; here Bt is admissible while I is not.

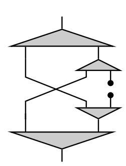
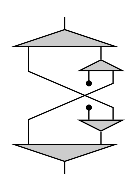
It is for this trickiness of the order structures on $\mathbf{PER}_{\mathcal{Q}}$ that we are introducing an alternative $\dot{\mathsf{P}}$ to the pairing combinator P (Definition 4.20)—the former leads to a different "implementation" $\dot{\times}$ of products. The merit of $\dot{\times}$ is that it exhibits a better order-theoretic property (namely it preserves admissibility, Lemma 5.13); we will need them for recursion. In contrast, P enjoys a useful combinatorial property: $(\mathsf{P}xy)z = zxy$.

Definition 5.11 (Combinator $\dot{\mathsf{P}}$, binary product $X \dot{\times} Y$). We define an element $\dot{\mathsf{P}} \in A_{\mathcal{Q}}$ by the string diagram in $\mathcal{K}\ell(\mathcal{Q})$ shown below on the left. The triangles denote $j : \mathbb{N} + \mathbb{N} \cong \mathbb{N} : k$ in Theorem 4.15. Then $\dot{\mathsf{P}}xy$ becomes as shown bottom on the right.

$\dot{\mathsf{P}}$:=

$\dot{\mathsf{P}}xy$ =


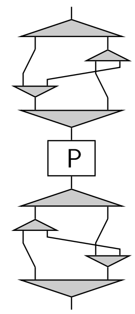
(40)

Let $\dot{\mathsf{P}}_{\mathsf{l}}$ and $\dot{\mathsf{P}}_{\mathsf{r}}$ be the following elements of $A_{\mathcal{Q}}$.

$\dot{\mathsf{P}}_{\mathsf{l}}$:=

$\dot{\mathsf{P}}_{\mathsf{r}}$:=


Here the nodes $\mathord{\text{⫯}}$ and $\mathord{\text{⫰}}$ denote the unique arrows $\mathbb{N} \nrightarrow 0$ and $0 \nrightarrow \mathbb{N}$ in $\mathcal{K}\ell(\mathcal{Q})$, respectively. Furthermore, let us introduce the following conversion combinators.

$$\mathsf{C}_{\mathsf{P}\mapsto\dot{\mathsf{P}}} := \lambda w.w\dot{\mathsf{P}} \qquad \mathsf{C}_{\dot{\mathsf{P}}\mapsto\mathsf{P}} :=$$



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

Lemma 5.12. *We have, for each $x, y \in A_{\mathcal{Q}}$,*

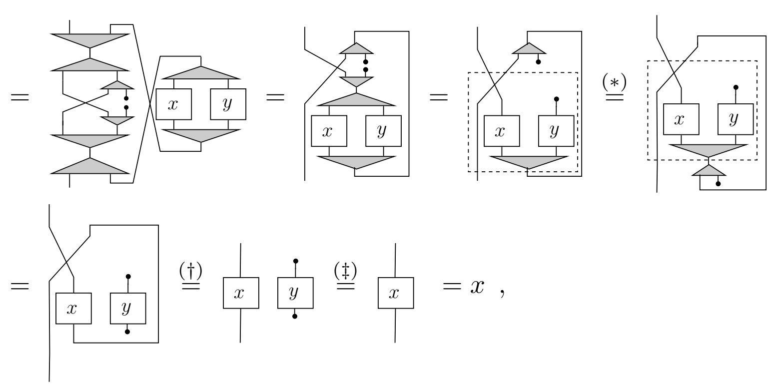
$$\dot{\mathsf{P}}_{\mathsf{l}}(\dot{\mathsf{P}}xy) = x \ , \qquad \dot{\mathsf{P}}_{\mathsf{r}}(\dot{\mathsf{P}}xy) = y \ ; \qquad \mathsf{C}_{\mathsf{P}\mapsto\dot{\mathsf{P}}}(\mathsf{P}xy) = \dot{\mathsf{P}}xy \ , \qquad \mathsf{C}_{\dot{\mathsf{P}}\mapsto\mathsf{P}}(\dot{\mathsf{P}}xy) = \mathsf{P}xy \ .$$

The latter two result in a canonical natural isomorphism $X \times Y \overset{\cong}{\Rightarrow} X \dot{\times} Y$ in $\mathbf{PER}_{\mathcal{Q}}$. Therefore in what follows we shall use $\times$ and $\dot{\times}$ interchangeably. That is, we suppress use of the conversion combinators $\mathsf{C}_{\mathsf{P}\mapsto\dot{\mathsf{P}}}$ and $\mathsf{C}_{\dot{\mathsf{P}}\mapsto\mathsf{P}}$.

Proof. In the proof we shall rely heavily on the reasoning in string diagrams in the traced monoidal category $\mathcal{K}\ell(\mathcal{Q})$.

For the first equality, the proof goes as follows.

$\dot{\mathsf{P}}_{\mathsf{l}}(\dot{\mathsf{P}}xy) =$



$= x$,

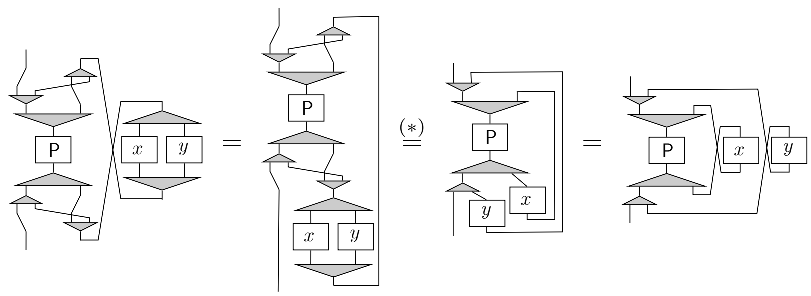
where $(*)$ and $(\dagger)$ hold because of the *dinaturality* (also called *sliding*) and *yanking* axioms of traced monoidal categories, respectively (see [44, 21]); and $(\ddagger)$ holds by a direct calculation in $\mathcal{K}\ell(\mathcal{Q})$. The second equality $\dot{\mathsf{P}}_{\mathsf{r}}(\dot{\mathsf{P}}xy) = y$ is similar.

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

$$\mathsf{C}_{\mathsf{P}\mapsto\dot{\mathsf{P}}}(\mathsf{P}xy) = (\mathsf{P}xy)\dot{\mathsf{P}} = \dot{\mathsf{P}}xy \ .$$

The last equality is shown as follows, where $(*)$ holds due to the dinaturality axiom.

$$\mathsf{C}_{\dot{\mathsf{P}}\mapsto\mathsf{P}}(\dot{\mathsf{P}}xy) =$$



$$= \mathsf{P}xy \ . \qquad \square$$

Lemma 5.13. *For admissible U, V and any X, we have $U \dot{\times} V$ and $X \multimap U$ admissible.* $\square$

As we observed earlier in Example 5.10, admissibility is not necessarily preserved by isomorphisms in $\mathbf{PER}_{\mathcal{Q}}$. Therefore replacing $\dot{\times}$ with $\times$ (that are isomorphic, see Lemma 5.12) would make the last result (Lemma 5.13) fail.

Admissibility of a PER (Definition 5.9) gives rise to a fixed-point construction, in the following sense.

Definition 5.14 (Fixed point operator). Let $U, X \in \mathbf{PER}_{\mathcal{Q}}$; assume that U is admissible. We introduce a *fixed point operator* (denoted by fix) that carries

$$f \ : \ !\,U \boxtimes !\,X \longrightarrow U \qquad \text{to} \qquad \mathrm{fix}(f) \ : \ !\,X \longrightarrow U$$

in the following way. Let c be a code of f. We define $c_0, c_1, \ldots \in |\,!\,X \multimap U|$ by

$$c_0 := \bot \ ;$$

$$c_{n+1} := \text{ the canonical code of } \big(\,!\,X \xrightarrow{\mathsf{con}} !\,X \boxtimes !\,X \xrightarrow{\delta\boxtimes\mathrm{id}} !\,!\,X \boxtimes !\,X \xrightarrow{![c_n]\boxtimes\mathrm{id}} !\,U \boxtimes !\,X \xrightarrow{[c]} U\,\big) \ .$$

A concrete description of c_{n+1} in terms of c_n can be easily given using in particular (36). 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 $!\,X \multimap U$ is admissible its supremum $\sup_i c_i$ belongs to the domain $|\,!\,X \multimap U|$. Finally, we define

$$\mathrm{fix}(f) \ := \ [\,\sup_i c_i\,] \ .$$

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

5.3. Quantum Mechanical Constructs in $\mathbf{PER}_{\mathcal{Q}}$

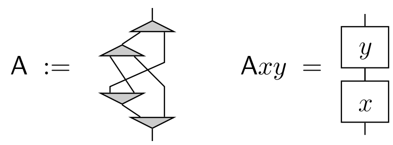
Here we introduce some constructs in $\mathbf{PER}_{\mathcal{Q}}$ that we use for interpreting quantum primitives in Hoq.

We start with the following combinator A that allows to "juxtapose" piping; see (23).

Definition 5.15 (Combinator A). We define $\mathsf{A} \in A_{\mathcal{Q}}$ by the string diagram in $\mathcal{K}\ell(\mathcal{Q})$ shown below. The triangles are $j : \mathbb{N} + \mathbb{N} \cong \mathbb{N} : k$ in Theorem 4.15. It is easily seen to satisfy the equation

$$\mathsf{A}xy = x \odot y \ , \tag{41}$$

where $\odot$ denotes composition of arrows in $\mathcal{K}\ell(Q)$ (see §4.2).



Definition 5.16 (Combinators Q_ρ, Q_U, $\mathsf{Q}^{N+1}_{|0_i\rangle}$, $\mathsf{Q}^{N+1}_{|1_i\rangle}$). We define elements $\mathsf{Q}_\rho, \mathsf{Q}_U, \mathsf{Q}^{N+1}_{|0_i\rangle}, \mathsf{Q}^{N+1}_{|1_i\rangle} \in A_{\mathcal{Q}}$ as follows. Here $N \in \mathbb{N}$, $\rho \in \mathrm{DM}_{2^N}$, U is an $2^N \times 2^N$ unitary matrix, and $i \in [1, N+1]$.

$\mathsf{Q}_\rho, \mathsf{Q}_U, \mathsf{Q}^{N+1}_{|0_i\rangle}, \mathsf{Q}^{N+1}_{|1_i\rangle} : \ \mathbb{N} \longrightarrow\!\!\!\!\!\!\!\!+\ \ \mathbb{N}$ in $\mathcal{K}\ell(\mathcal{Q})$; given $\sigma \in \mathrm{DM}_m$,

$$\big(\mathsf{Q}_\rho(k)(l)\big)_{m,n}(\sigma) := \begin{cases} \rho \otimes \sigma & \text{if } k = l \wedge n = 2^N \cdot m \\ 0 & \text{otherwise,} \end{cases}$$

$$\big(\mathsf{Q}_U(k)(l)\big)_{m,n}(\sigma) := \begin{cases} (U(\_)U^\dagger \otimes \mathrm{id}_j)\sigma & \text{if } k = l \text{ and } \exists j.\,(n = m = 2^N \cdot j) \\ 0 & \text{otherwise,} \end{cases}$$

$$\big(\mathsf{Q}^{N+1}_{|0_i\rangle}(k)(l)\big)_{m,n}(\sigma) := \begin{cases} \big(\langle 0_i|\_|0_i\rangle \otimes \mathrm{id}_j\big)\sigma & \text{if } k = l \text{ and } \exists j.\,(m = 2^{N+1} \cdot j \wedge n = 2^N \cdot j) \\ 0 & \text{otherwise.} \end{cases}$$

$$\big(\mathsf{Q}^{N+1}_{|1_i\rangle}(k)(l)\big)_{m,n}(\sigma) := \begin{cases} \big(\langle 1_i|\_|1_i\rangle \otimes \mathrm{id}_j\big)\sigma & \text{if } k = l \text{ and } \exists j.\,(m = 2^{N+1} \cdot j \wedge n = 2^N \cdot j) \\ 0 & \text{otherwise.} \end{cases}$$

In the definition of Q_ρ, $\otimes$ denotes tensor product of matrices. The combinator Q_ρ "adjoins an auxiliary state ρ": an incoming token carrying σ comes out of the same pipe, with its state composed with ρ. In particular the state $1 \in \mathrm{DM}_1$ comes out as ρ. Similarly, the combinator Q_U applies the unitary transformation U to (the first N qubits of) the incoming quantum state; and $\mathsf{Q}^{N+1}_{|0_i\rangle}$ and $\mathsf{Q}^{N+1}_{|1_i\rangle}$ apply suitable projections (cf. Remark 2.11), focusing on the first $N+1$ qubits of the incoming quantum state σ and projecting its i-th qubit. Indeed:

Lemma 5.17. *We have the following equalities.*

$$\mathsf{Q}_\rho \odot \mathsf{Q}_\sigma = \mathsf{Q}_{\rho\otimes\sigma} \quad \mathsf{Q}_U \odot \mathsf{Q}_\rho = \mathsf{Q}_{U\rho U^\dagger} \quad \mathsf{Q}^{N+1}_{|0_i\rangle} \odot \mathsf{Q}_\sigma = \mathsf{Q}_{\langle 0_i|\sigma|0_i\rangle} \quad \mathsf{Q}^{N+1}_{|1_i\rangle} \odot \mathsf{Q}_\sigma = \mathsf{Q}_{\langle 1_i|\sigma|1_i\rangle}$$

□

Definition 5.18 ($[\![N\text{-}\mathtt{qbit}]\!]$,$[\![\mathtt{bit}]\!]$). For each $N \in \mathbb{N}$ we define a PER $[\![N\text{-}\mathtt{qbit}]\!]$ over $A_{\mathcal{Q}}$ by:

$$[\![N\text{-}\mathtt{qbit}]\!] \ := \ \big\{ (\mathsf{Q}_\rho, \mathsf{Q}_\rho) \mid \rho \in \mathrm{DM}_{2^N} \big\} \ .$$

In particular, $[\![0\text{-}\mathtt{qbit}]\!] = \{ (\mathsf{Q}_p, \mathsf{Q}_p) \mid p \in [0,1] \}$ (cf. Definition 2.3). This type can be thought of as the unit interval $[0,1]$.

A PER $[\![\mathtt{bit}]\!]$ is defined to be $\mathrm{I} + \mathrm{I}$ (see Lemma 5.2).

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

Lemma 5.19. *There is a canonical isomorphism* $[\![\texttt{bit}]\!] \xrightarrow{\cong} ![\![\texttt{bit}]\!]$ *in* $\mathbf{PER}_{\mathcal{Q}}$.

Proof. Use the isomorphisms (39) in Lemma 5.3. □

The quantum combinators in Definition 5.16 are combined with the A combinator in Definition 5.15 and yield the following combinators for quantum operations.

Definition 5.20 (Combinators $\mathsf{U}_U, \mathsf{Pr}^{N+1}_{|0_i\rangle}, \mathsf{Pr}^{N+1}_{|1_i\rangle}$). We define

$$\mathsf{U}_U := \mathsf{A}\mathsf{Q}_U \ , \quad \mathsf{Pr}^{N+1}_{|0_i\rangle} := \mathsf{A}\mathsf{Q}^{N+1}_{|0_i\rangle} \ , \quad \mathsf{Pr}^{N+1}_{|1_i\rangle} := \mathsf{A}\mathsf{Q}^{N+1}_{|1_i\rangle} \ .$$

Lemma 5.21. *For combinators in Definition 5.16 and 5.20, and* ρ, σ, U *of suitable dimensions, we have*

$$\mathsf{A}\mathsf{Q}_\rho \mathsf{Q}_\sigma = \mathsf{Q}_{\rho\otimes\sigma} \ , \qquad \mathsf{U}_U \mathsf{Q}_\rho = \mathsf{Q}_{U\rho U^\dagger} \ ,$$
$$\mathsf{Pr}^{N+1}_{|0_i\rangle} \mathsf{Q}_\sigma = \mathsf{Q}_{\langle 0_i|\sigma|0_i\rangle} \ , \qquad \mathsf{Pr}^{N+1}_{|1_i\rangle} \mathsf{Q}_\sigma = \mathsf{Q}_{\langle 1_i|\sigma|1_i\rangle} \ .$$

Proof. Obvious from Lemma 5.17. □

Remark 5.22 (No-cloning in the category $\mathbf{PER}_{\mathcal{Q}}$). We noted in Remark 3.21 that: the "preparation" primitive $\texttt{new}_\rho$ in Hoq can be typable with the type $!\,k$-$\texttt{qbit}$; hence the constant $\texttt{new}_\rho$ is duplicable; nevertheless "no-cloning" is enforced by the linear typing discipline in Hoq, in the sense that a given quantum state—whose preparation apparatus we do not have access to—cannot be duplicated. Here we shall discuss how these design choices are reflected in our model $\mathbf{PER}_{\mathcal{Q}}$.

It is important to note that, in Hoq and in its model $\mathbf{PER}_{\mathcal{Q}}$ alike, the quantum tensor $\otimes$ (for composed and entangled systems) and the linear-logic tensor $\boxtimes$ are distinguished. Let us speak in the piping analogy in (21): an arrow in $\mathbf{PER}_{\mathcal{Q}}$ is "realized" by a code $c \in A_{\mathcal{Q}}$ (Definition 4.18); and each element c of $A_{\mathcal{Q}}$ is a Kleisli arrow $c\colon \mathbb{N} \nrightarrow \mathbb{N}$ in $\mathcal{K}\ell(\mathcal{Q})$ (Theorem 4.15), that is, piping like in (21) with countably infinite numbers of entrances and exits. Then the quantum tensor $\otimes$ resides solely in the quantum states carried by tokens; importantly it has nothing to do with *how tokens move around* (except for the indirect relationship in which measurements on quantum states result in branching of tokens). In contrast, the type constructors derived from linear logic—namely $\boxtimes$, $\multimap$ and !—are all concerned about how pipes are connected.

Concretely, the denotation of $\texttt{new}_{|0\rangle\langle 0|} \colon \ !\,\texttt{qbit}$ will look like the piping in (4.3), with $a = \mathsf{Q}_{|0\rangle\langle 0|}$ where the latter is from Definition 5.16. Therefore ! in the type $!\,\texttt{qbit}$ here merely makes infinitely many copies $a = \mathsf{Q}_{|0\rangle\langle 0|}$; it does nothing like duplicating quantum states (that are carried by tokens that go through the pipes). It is easy to see that $!\,\mathsf{Q}_{|0\rangle\langle 0|} = \mathsf{Q}_{|0\rangle\langle 0|}$ as arrows $\mathbb{N} \nrightarrow \mathbb{N}$ in $\mathcal{K}\ell(\mathcal{Q})$, as a matter of fact, since $\mathsf{Q}_{|0\rangle\langle 0|}$ does not alter the path taken by a token but only adjoins the quantum state $|0\rangle\langle 0|$ to the one carried by the token.

In order to capture probabilistic branching in Hoq, we use a strong monad T on $\mathbf{PER}_{\mathcal{Q}}$ following Moggi's idea [39]: the interpretation of a type judgment $\Delta \vdash M : A$ will be an arrow $[\![\Delta]\!] \to T[\![A]\!]$ in the category $\mathbf{PER}_{\mathcal{Q}}$. The monad T is in fact a continuation monad $T = (\_ \multimap R) \multimap R$ with a suitable result type R; hence our semantics is in the *continuation-passing style (CPS)*. The resulting CPS model is fairly complex as 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 $\mathtt{meas}^1_1$ that measures one qubit; for the purpose of case-distinction based on the outcome, it is desired that $\mathtt{meas}^1_1$ is of the type $\mathtt{qbit} \multimap \mathtt{bit}$. Therefore it is natural to use *monadic semantics*: we use a monad T—with a probabilistic flavor—so that we have $[\![\mathtt{meas}^1_1]\!] : [\![\mathtt{qbit}]\!] \to T[\![\mathtt{bit}]\!]$.

For our GoI semantics based on local interaction, however, a simple "probability distribution" monad (something like $\mathcal{D}$ in §2.2) would not do. One explanation is as follows. Think of the construct $\mathtt{meas}^2_1 : 2\text{-}\mathtt{qbit} \multimap \mathtt{bit} \boxtimes \mathtt{qbit}$: it takes a state ρ of a 2-qubit system; measures the first qubit; and returns its outcome ($\mathtt{tt}$ or $\mathtt{ff}$) together with the remaining qubit. The probability of observing $\mathtt{tt}$ is calculated by $\mathtt{tr}\Big(\big(\langle 0_1 | \_ | 0_1 \rangle \otimes \mathrm{id}_2\big)\rho\Big)$, 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 ρ implicitly carry a probability in the form of its trace value $\mathtt{tr}(\rho)$, a naive interpretation of $\mathtt{meas}^2_1$ would have the codomain $\mathtt{bit} \otimes \mathtt{qbit}$—with entanglement—rather than the desired codomain $\mathtt{bit} \boxtimes \mathtt{qbit}$ 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 infinite complete binary trees with each edge labeled with a real number $p \in [0, 1]$—obtained as a final coalgebra.

Definition 5.23 (The functor F_{pbt}). We define an endofunctor $F_{\mathrm{pbt}} : \mathbf{PER}_{\mathcal{Q}} \to \mathbf{PER}_{\mathcal{Q}}$ by

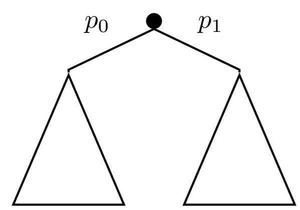
$$F_{\mathrm{pbt}} \quad := \quad [\![\mathtt{bit}]\!] \multimap ([\![0\text{-}\mathtt{qbit}]\!] \mathbin{\dot\times} \_)$$

where the objects $[\![\mathtt{bit}]\!]$ and $[\![0\text{-}\mathtt{qbit}]\!]$ are as in Definition 5.18.

In the functor F_{pbt} we use $\dot\times$ 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_{pbt} represents the branching type of *probabilistic binary trees* like the one shown below. In the functor, the $[\![\mathtt{bit}]\!]$ part designates which of the left and right successors; and the $[\![0\text{-}\mathtt{qbit}]\!]$ part designates the value $p_i \in [0, 1]$

that is assigned to the edge (here $i \in \{0,1\}$).



(42)

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. [40, 41]), the collection of such trees is identified with (the carrier of) a final coalgebra.

Lemma 5.24 (The result type R)**.** *The functor F_{pbt} has a final coalgebra*

$$r \;:\; R \xrightarrow{\cong} F_{\mathrm{pbt}} R \enspace .$$

PROOF. We use the standard construction by a *final sequence* (see e.g. [75, 76]). Let

$$\mathsf{Bt} \;:=\; \{(\bot,\bot)\}$$

be a final object; here $\bot \in A_{\mathcal{Q}}$ is the least element with respect to the order $\sqsubseteq$ (Lemma 5.8). Although $\mathsf{Bt} \overset{\cong}{\Rightarrow} \mathrm{I}$, we use Bt due to its order-theoretic property (namely: Bt is admissible while I is not, see Example 5.10). Consider the final sequence

$$\mathsf{Bt} \xleftarrow{\;\mathsf{weak}\;} F_{\mathrm{pbt}}\mathsf{Bt} \xleftarrow{\;F_{\mathrm{pbt}}\mathsf{weak}\;} F_{\mathrm{pbt}}^2\mathsf{Bt} \xleftarrow{\;F_{\mathrm{pbt}}^2\mathsf{weak}\;} \cdots \tag{43}$$

in $\mathbf{PER}_{\mathcal{Q}}$. Here weak denotes the unique arrow to final Bt. For $j \le i$, let $c_{i,j}$ be (the canonical choice of) a realizer of the arrow $F_{\mathrm{pbt}}^i\mathsf{Bt} \to F_{\mathrm{pbt}}^j\mathsf{Bt}$ in the final sequence.

By Proposition 5.6 there is a limit R of the sequence (43). Moreover the PER R can be concretely described as: the symmetric closure of

$$\begin{aligned} \Big\{ \big(\dot{\mathsf{P}}(k_i)_i u, \dot{\mathsf{P}}(k_i')_i u'\big) \,\Big|\, & j \le i \text{ implies} \\ & \big(c_{i,j}(k_i u), k_j' u'\big) \in F_{\mathrm{pbt}}^j\mathsf{Bt} \text{ and } \big(k_j u, c_{i,j}(k_i' u')\big) \in F_{\mathrm{pbt}}^j\mathsf{Bt} \Big\} \enspace . \end{aligned} \tag{44}$$

Now the functor F_{pbt} preserves limits: $[\![\texttt{0-qbit}]\!] \mathbin{\dot{\times}} \_$ does since $\dot{\times}$ is for products; and $[\![\texttt{bit}]\!] \multimap \_$ does since it has a left adjoint $[\![\texttt{bit}]\!] \boxtimes \_$ (Theorem 4.21). Therefore the well-known argument (see e.g. [75, 76]) proves that R carries a final F_{pbt}-coalgebra, with the coalgebraic structure $r : R \overset{\cong}{\Rightarrow} F_{\mathrm{pbt}}R$ obtained as a suitable mediating arrow. $\square$

Lemma 5.25. *Let*

$$T \;:=\; (\_ \multimap R) \multimap R \enspace .$$

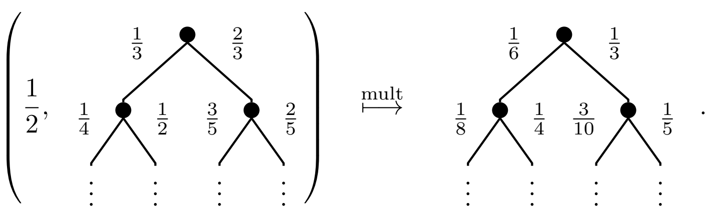
Then T is a strong monad on $\mathbf{PER}_{\mathcal{Q}}$.

Proof. This is a standard fact that is true—as one would readily prove—for any symmetric monoidal closed category $(\mathbb{C}, \mathrm{I}, \boxtimes, \multimap)$ and for any $R \in \mathbb{C}$. $\square$

We introduce a map

$$\mathrm{mult} \;:\; [\![\texttt{0-qbit}]\!] \boxtimes R \longrightarrow R$$

that will be needed later. Intuitively, what it does is to receive $p \in [0,1]$ and a binary tree t like (42) and returns the tree in which the probabilities assigned to all the edges are multiplied by p. For example,



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

Definition 5.26 (mult)**.** Let a coalgebra

$$c_{\mathrm{mult}} \;:\; [\![\texttt{0-qbit}]\!] \boxtimes R \longrightarrow F_{\mathrm{pbt}}\big([\![\texttt{0-qbit}]\!] \boxtimes R\big) \quad \text{in } \mathbf{PER}_{\mathcal{Q}}$$

be defined as the adjoint transpose of the following composite.

$$\begin{aligned} &[\![\texttt{0-qbit}]\!] \boxtimes R \boxtimes [\![\texttt{bit}]\!] \xrightarrow{\mathrm{id} \boxtimes r \boxtimes \mathrm{id}} [\![\texttt{0-qbit}]\!] \boxtimes ([\![\texttt{bit}]\!] \multimap ([\![\texttt{0-qbit}]\!] \dot{\times} R)) \boxtimes [\![\texttt{bit}]\!] \\ &\quad \xrightarrow{\mathrm{id} \boxtimes \mathsf{ev}} [\![\texttt{0-qbit}]\!] \boxtimes ([\![\texttt{0-qbit}]\!] \dot{\times} R) \xrightarrow{\langle \mathrm{id} \boxtimes \pi_\ell, \mathrm{id} \boxtimes \pi_r \rangle} ([\![\texttt{0-qbit}]\!] \boxtimes [\![\texttt{0-qbit}]\!]) \dot{\times} ([\![\texttt{0-qbit}]\!] \boxtimes R) \\ &\quad \xrightarrow{[\lambda w. w\mathsf{A}] \dot{\times} \mathrm{id}} [\![\texttt{0-qbit}]\!] \dot{\times} ([\![\texttt{0-qbit}]\!] \boxtimes R) \end{aligned}$$

Here the map $[\lambda w.w\mathsf{A}] : [\![\texttt{0-qbit}]\!] \boxtimes [\![\texttt{0-qbit}]\!] \to [\![\texttt{0-qbit}]\!]$—that carries $\mathsf{P}xy$ to $x \odot y$—plays the role of multiplication over $[0,1]$. By finality, this coalgebra c_{mult} induces a unique coalgebra homomorphism from c_{mult} to r. This is denoted by mult.

$$\begin{tikzcd} F_{\mathrm{pbt}}\big([\![\texttt{0-qbit}]\!] \boxtimes R\big) \arrow[r, dashed, "F_{\mathrm{pbt}}(\mathrm{mult})"] & F_{\mathrm{pbt}} R \\ {[\![\texttt{0-qbit}]\!] \boxtimes R} \arrow[u, "c_{\mathrm{mult}}"] \arrow[r, dashed, "\mathrm{mult}"'] & R \arrow[u, "\cong"', "r"] \end{tikzcd}$$

For the purpose of interpreting recursion we need the following property. The notion of admissibility is from Definition 5.9.

Lemma 5.27. *The PER R in (44) is admissible. Therefore by Lemma 5.13, $TX = (X \multimap R) \multimap R$ is admissible for each X; so is $X \multimap TY$.* $\square$

The last result (Lemma 5.27) implies that the following "fixed-point" construction is available, by Definition 5.14 (where U was required to be admissible).

$$\frac{f \;:\; !R \boxtimes !X \longrightarrow R}{\mathrm{fix}(f) \;:\; !X \longrightarrow R}\;, \quad \text{for each } X \in \mathbf{PER}_{\mathcal{Q}}.$$

Remark 5.28. The need for CPS-style semantics, as is the case here, does not seem to be a phenomenon unique to ("classical control and quantum data") quantum computation. The same kind of difficulties are already observed with nondeterministic branching (see the leading example in [77]), and they seem to occur with *any* computational effect—at least the *algebraic* ones in the sense of [78].

The work [77] presents an alternative approach to the one used here (namely the CPS-style semantics by a continuation monad): it identifies the underlying problem to be the "memoryless" nature of processes (i.e. links in proof nets, boxes in string diagrams, etc.), and solves the problem by systematically equipping the processes with internal states (called *memories*) exploiting *coalgebraic component calculus*.

The framework in [77] is categorical and general, parametrized by a monad and algebraic operations interpreted over it. We expect its application to the current question (of higher-order quantum computation) will simplify the current CPS model by a great deal. This is however left as future work.

5.5. Interpretation

Standing on all the constructs and properties exhibited in §5.1–5.4, we shall now interpret Hoq in the category $\mathbf{PER}_{\mathcal{Q}}$.

Definition 5.29 (Interpretation of types). For each Hoq-type A, we assign $[\![A]\!] \in \mathbf{PER}_{\mathcal{Q}}$ as follows, using the structures of $\mathbf{PER}_{\mathcal{Q}}$ we described in previous sections. For base types, $[\![N\text{-}\mathtt{qbit}]\!]$ is as in Definition 5.18.

$$\begin{array}{rcl@{\qquad\qquad}rcl} [\![!\,A]\!] & := & ![\![A]\!] & [\![A \multimap B]\!] & := & [\![A]\!] \multimap T[\![B]\!] \\ [\![\top]\!] & := & \mathrm{I} & [\![A \boxtimes B]\!] & := & [\![A]\!] \boxtimes [\![B]\!] \\ [\![A + B]\!] & := & [\![A]\!] + [\![B]\!] & & & \end{array}$$

Definition 5.30 (Interpretation of the subtype relation). We shall assign, to each derivable subtype relation $A <: B$, an arrow

$$[\![A <: B]\!] \;:\; [\![A]\!] \longrightarrow [\![B]\!] \qquad \text{in } \mathbf{PER}_{\mathcal{Q}}.$$

For that purpose we first introduce a natural transformation

$$\delta_{n,m} \;:\; !^n\, X \longrightarrow\, !^m\, X \;\;, \quad \text{natural in } X,$$

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

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

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

$$
\begin{aligned}
[\![!^n\, k\text{-}\mathtt{qbit} <: !^m\, k\text{-}\mathtt{qbit}]\!] &:= \quad \big(!^n [\![k\text{-}\mathtt{qbit}]\!] \xrightarrow{\delta_{n,m}} !^m [\![k\text{-}\mathtt{qbit}]\!] \big) \ , \\
[\![!^n \top <: !^m \top]\!] &:= \quad \big(!^n [\![\top]\!] \xrightarrow{\delta_{n,m}} !^m [\![\top]\!] \big) \ , \\
[\![!^n (A_1 \boxtimes A_2) <: !^m (B_1 \boxtimes B_2)]\!] &:=
\end{aligned}
$$
$$
\Big(!^n \big([\![A_1]\!] \boxtimes [\![A_2]\!] \big) \xrightarrow{!^n \left([\![A_1 <: B_1]\!] \boxtimes [\![A_2 <: B_2]\!] \right)} !^n \big([\![B_1]\!] \boxtimes [\![B_2]\!] \big) \xrightarrow{\delta_{n,m}} !^m \big([\![B_1]\!] \boxtimes [\![B_2]\!] \big) \Big) \ .
$$

$[\![!^n (A_1 + A_2) <: !^m (B_1 + B_2)]\!]$ and $[\![!^n (A_1 \multimap A_2) <: !^m (B_1 \multimap B_2)]\!]$ 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 5.31. *Let $A <: B$ and $B <: C$. Then $A <: C$ by Lemma 3.15.1; moreover*

$$
[\![A <: C]\!] = [\![B <: C]\!] \circ [\![A <: B]\!] \ .
$$

PROOF. Much like the proof of Lemma 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 Lemma 5.3. □

We now interpret constants. In the general definition (Definition 5.34) a typed term $\Delta \vdash M : A$ will be interpreted as an arrow $[\![M]\!] : [\![\Delta]\!] \to T[\![A]\!]$—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 $!\,\mathrm{DType}(c)$ (see (Ax.2) in Table 1). Therefore we first define $[\![c]\!]_{\mathrm{const}} : \mathrm{I} \to [\![\mathrm{DType}(c)]\!]$ whose descriptions are simpler, and then $[\![c]\!] : \mathrm{I} \to T[\![\mathrm{DType}(c)]\!]$ will be defined to be the embedding via the unit $\eta^T : \mathrm{id} \Rightarrow T$ of the monad T.

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

Definition 5.32 (Interpretation of constants). To each constant c in Hoq we assign an arrow

$$
[\![c]\!]_{\mathrm{const}} \ : \ \mathrm{I} \longrightarrow [\![\mathrm{DType}(c)]\!]
$$

as follows. For $c \equiv \mathtt{new}_\rho$, $[\![\mathtt{new}_\rho]\!]_{\mathrm{const}}$ is given by

$$
\mathrm{I} \xrightarrow{[\lambda x. \mathsf{Q}_\rho]} [\![n\text{-}\mathtt{qbit}]\!] \ .
$$

For $c \equiv \mathtt{meas}_i^{n+1}$ with $n \geq 1$, by transpose we need an arrow

$$
[\![(n+1)\text{-}\mathtt{qbit}]\!] \boxtimes \big(([\![\mathtt{bit}]\!] \boxtimes [\![n\text{-}\mathtt{qbit}]\!]) \multimap R \big) \xrightarrow{m} R \ , \tag{45}
$$

where we also used $![\![\mathtt{bit}]\!] \cong [\![\mathtt{bit}]\!]$ (Lemma 5.19). By R's fixed point property (namely $r : R \overset{\cong}{\Rightarrow} [\![\mathtt{bit}]\!] \multimap ([\![0\text{-}\mathtt{qbit}]\!] \times R)$), this is further reduced to an arrow

$$
[\![(n+1)\text{-}\mathtt{qbit}]\!] \boxtimes \big(([\![\mathtt{bit}]\!] \boxtimes [\![n\text{-}\mathtt{qbit}]\!]) \multimap R \big) \boxtimes [\![\mathtt{bit}]\!] \ \longrightarrow \ [\![0\text{-}\mathtt{qbit}]\!] \times R \ .
$$

This can be obtained as follows.

$$
\begin{array}{l}
[\![(n+1)\text{-}\mathtt{qbit}]\!] \boxtimes \big(([\![\mathtt{bit}]\!] \boxtimes [\![n\text{-}\mathtt{qbit}]\!]) \multimap R \big) \boxtimes [\![\mathtt{bit}]\!] \\
\cong\ [\![(n+1)\text{-}\mathtt{qbit}]\!] \boxtimes \Big(([\![n\text{-}\mathtt{qbit}]\!] + [\![n\text{-}\mathtt{qbit}]\!]) \multimap R \Big) \boxtimes [\![\mathtt{bit}]\!] \qquad \text{by (38) and } \mathrm{I} \boxtimes X \cong X \\
\cong\ [\![(n+1)\text{-}\mathtt{qbit}]\!] \boxtimes \Big(([\![n\text{-}\mathtt{qbit}]\!] \multimap R) \times ([\![n\text{-}\mathtt{qbit}]\!] \multimap R) \Big) \boxtimes [\![\mathtt{bit}]\!] \qquad \text{by (38)} \\
\cong\ [\![(n+1)\text{-}\mathtt{qbit}]\!] \boxtimes ([\![n\text{-}\mathtt{qbit}]\!] \multimap R)^{\times 2} + [\![(n+1)\text{-}\mathtt{qbit}]\!] \boxtimes ([\![n\text{-}\mathtt{qbit}]\!] \multimap R)^{\times 2} \qquad \text{by (38)} \\
\xrightarrow{\left[\mathsf{Pr}^{n+1}_{|0_i\rangle}\right] \boxtimes \pi_\ell + \left[\mathsf{Pr}^{n+1}_{|1_i\rangle}\right] \boxtimes \pi_r}\ [\![n\text{-}\mathtt{qbit}]\!] \boxtimes ([\![n\text{-}\mathtt{qbit}]\!] \multimap R) + [\![n\text{-}\mathtt{qbit}]\!] \boxtimes ([\![n\text{-}\mathtt{qbit}]\!] \multimap R) \\
\xrightarrow{\mathsf{ev}+\mathsf{ev}}\ R + R \xrightarrow{[\langle [\lambda x.\mathsf{Q}_0], \mathrm{id} \rangle, \langle [\lambda x.\mathsf{Q}_0], \mathrm{id} \rangle]}\ [\![0\text{-}\mathtt{qbit}]\!] \times R\ .
\end{array}
$$

Here $\mathsf{Pr}^{n+1}_{|0_i\rangle}$ and $\mathsf{Pr}^{n+1}_{|1_i\rangle}$ are from Definition 5.20, and Q_0 is from Definition 5.16 (see also Definition 5.18).

For $c \equiv \mathtt{meas}^1_1$, similarly, by transpose we need an arrow

$$
[\![\mathtt{qbit}]\!] \boxtimes \big([\![\mathtt{bit}]\!] \multimap R \big) \xrightarrow{m'} R\ , \tag{46}
$$

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

$$
[\![\mathtt{qbit}]\!] \boxtimes \big([\![\mathtt{bit}]\!] \multimap R \big) \boxtimes [\![\mathtt{bit}]\!] \longrightarrow [\![0\text{-}\mathtt{qbit}]\!] \times R\ .
$$

This is obtained as follows.

$$
\begin{array}{l}
[\![\mathtt{qbit}]\!] \boxtimes \big([\![\mathtt{bit}]\!] \multimap R \big) \boxtimes [\![\mathtt{bit}]\!] \\
\cong\ [\![\mathtt{qbit}]\!] \boxtimes \big([\![\mathtt{bit}]\!] \multimap R \big) + [\![\mathtt{qbit}]\!] \boxtimes \big([\![\mathtt{bit}]\!] \multimap R \big) \qquad \text{by (38)} \\
\cong\ [\![\mathtt{qbit}]\!] \boxtimes R^{\times 2} + [\![\mathtt{qbit}]\!] \boxtimes R^{\times 2} \qquad \text{by (38)} \\
\xrightarrow{\left[\mathsf{Pr}^1_{|0\rangle}\right] \boxtimes \pi_\ell + \left[\mathsf{Pr}^1_{|1\rangle}\right] \boxtimes \pi_r}\ [\![0\text{-}\mathtt{qbit}]\!] \boxtimes R + [\![0\text{-}\mathtt{qbit}]\!] \boxtimes R \\
\xrightarrow{[\kappa_\ell, \kappa_r]}\ [\![0\text{-}\mathtt{qbit}]\!] \boxtimes R \xrightarrow{\langle [\lambda x.\mathsf{Q}_0], \mathrm{mult} \rangle}\ [\![0\text{-}\mathtt{qbit}]\!] \times R\ ;
\end{array}
$$

here mult is from Definition 5.26.

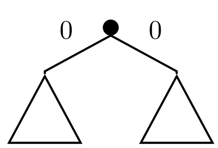
For the other constants we use Theorem 4.21, Lemma 5.1, Definition 5.16 and Definition 5.20. The arrow $[\![U]\!]_{\mathrm{const}}$ is the transpose of

$$
[\![n\text{-}\mathtt{qbit}]\!] \xrightarrow{[\mathsf{U}_U]} [\![n\text{-}\mathtt{qbit}]\!] \xrightarrow{\eta^T} T[\![n\text{-}\mathtt{qbit}]\!]\ ;
$$

$[\![\mathtt{cmp}_{m,n}]\!]_{\mathrm{const}}$ is the transpose of

$$
[\![m\text{-}\mathtt{qbit}]\!] \boxtimes [\![n\text{-}\mathtt{qbit}]\!] \xrightarrow{[\lambda w.w\mathsf{A}]} [\![(m+n)\text{-}\mathtt{qbit}]\!] \xrightarrow{\eta^T} T[\![(m+n)\text{-}\mathtt{qbit}]\!]\ ;
$$

The use of Q_0 (that stands for the value $0 \in [0,1]$) in the last line of the definition of $[\![\mathtt{meas}^{n+1}_i]\!]_{\mathrm{const}}$ indicates that a tree $m(t) \in |R|$ that can arise as an outcome of the map m in (45) looks as follows.



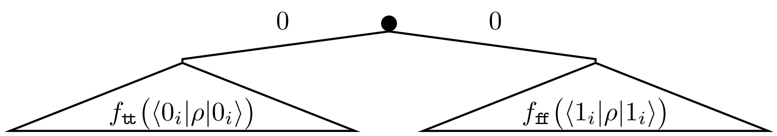
(47)

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 §5.4, the actual probabilities are carried implicitly by the remaining quantum states (consisting of n qubits) as their trace values. The labels 0 in (47) mean calculation of probabilities is postponed; they are done later and probabilities occur on some lower level in the tree (47).

More specifically, the idea for $[\![\mathtt{meas}_i^{n+1}]\!]_{\mathrm{const}}$ is as follows. Let $n \geq 1$ and consider the map m in (45), which can be identified (via Lemma 5.3) with an arrow

$$[\![(n+1)\text{-}\mathtt{qbit}]\!] \boxtimes ([\![n\text{-}\mathtt{qbit}]\!] \multimap R)^{\times 2} \xrightarrow{\overline{m}} R \ .$$

Roughly speaking its input is a triple $(\rho, f_{\mathtt{tt}}, f_{\mathtt{ff}})$ of $\rho \in \mathrm{DM}_{2^{n+1}}$ and $f_{\mathtt{tt}}, f_{\mathtt{ff}} : \mathrm{DM}_{2^n} \to R$. Then $\overline{m}$'s output is the following tree.

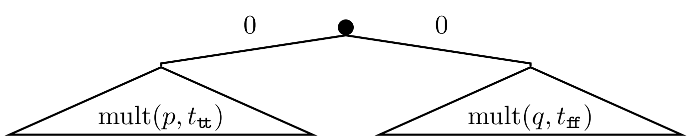


Here we put 0 as the labels on the edges of depth one; the probabilities for observing $|0_i\rangle$ or $|1_i\rangle$ are implicitly passed down in the form of the trace of the projected matrices.

When there is only one qubit left, we finally compute actual probabilities. This is what $[\![\mathtt{meas}_1^1]\!]_{\mathrm{const}}$ does. Consider m' in (46), which can be identified with

$$[\![\mathtt{qbit}]\!] \boxtimes (R \times R) \xrightarrow{\overline{m'}} R \ .$$

Its input is roughly a triple $(\rho, t_{\mathtt{tt}}, t_{\mathtt{ff}})$ of $\rho \in \mathrm{DM}_2$ and trees $t_{\mathtt{tt}}, t_{\mathtt{ff}} \in R$. Let $p = \langle 0|\rho|0\rangle$ and $q = \langle 1|\rho|1\rangle$; these are the probabilities for each outcome of the measurement. Then the output of $\overline{m'}$ is the following tree.



Recall that $\mathrm{mult}(p, t)$ 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 $t_{\mathtt{tt}}$ and $t_{\mathtt{ff}}$ 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 (Lemma 5.35) holds—and this adds all the details.

Definition 5.33 (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 $[\![\Delta]\!] \in \mathbf{PER}_{\mathcal{Q}}$ by $[\![A_{\sigma(1)}]\!] \boxtimes \cdots \boxtimes [\![A_{\sigma(n)}]\!]$, where σ is a bijection s.t. $x_{\sigma(1)} \prec \cdots \prec x_{\sigma(n)}$.

Definition 5.34 (Interpretation of type judgments). For each derivation $\Pi \Vdash \Delta \vdash M : A$ of a type judgment in Hoq, we assign an arrow

$$[\![\Pi]\!] \ : \ [\![\Delta]\!] \longrightarrow T[\![A]\!]$$

in the following way. Let $\Delta|_{\mathrm{FV}(M)}$ denote the obvious restriction of a context Δ to the set $\mathrm{FV}(M)$ of free variables in M. First we define

$$[\![\Pi]\!]_{\mathrm{FV}} \ : \ [\![\Delta|_{\mathrm{FV}(M)}]\!] \longrightarrow T[\![A]\!]$$

by induction on the derivation, which is used in

$$\Bigl([\![\Delta]\!] \xrightarrow{[\![\Pi]\!]} T[\![A]\!] \Bigr) \quad := \quad \Bigl([\![\Delta]\!] \xrightarrow{\mathsf{weak}} [\![\Delta|_{\mathrm{FV}(M)}]\!] \xrightarrow{[\![\Pi]\!]_{\mathrm{FV}}} T[\![A]\!] \Bigr) \ .$$

The definition of $[\![\Pi]\!]_{\mathrm{FV}}$ is as shown below. It is by induction—therefore the definition relies on the interpretations $[\![\Pi']\!]_{\mathrm{FV}}, [\![\Pi'']\!]_{\mathrm{FV}}, \ldots$ of the sub-derivation(s) of Π that derive the second last typing judgment(s).

$$\dfrac{\cdots \qquad \begin{array}{c}\vdots\ \Pi' \\ \Delta' \vdash M' \colon A'\end{array} \qquad \cdots \qquad \begin{array}{c}\vdots\ \Pi'' \\ \Delta'' \vdash M'' \colon A''\end{array} \qquad \cdots}{\Delta \vdash M \colon A}$$

In such cases, for simplicity of presentation, we shall refer to $[\![\Pi']\!]_{\mathrm{FV}}$ as $[\![M']\!]_{\mathrm{FV}}$, letting a term stand for the derivation tree that assigns a type to it. This will not cause confusion.

$\boxed{\mathrm{Ax.1}}$ $[\![A]\!] \xrightarrow{[\![A<:A']\!]} [\![A']\!] \xrightarrow{\eta^T} T[\![A']\!]$

$\boxed{\mathrm{Ax.2}}$ $\mathrm{I} \xrightarrow{\varphi'} !\,\mathrm{I} \xrightarrow{![\![c]\!]_{\mathrm{const}} \text{ (cf. Definition 5.32)}} ![\![\mathrm{DType}(c)]\!] \xrightarrow{\eta^T} T\,![\![\mathrm{DType}(c)]\!] \xrightarrow{T[\![!\,\mathrm{DType}(c)<:A]\!]} T[\![A]\!]$

$\boxed{\multimap.\mathrm{I}_1}$ $[\![\Delta|_{\mathrm{FV}(\lambda x^A.M)}]\!] \xrightarrow{g} [\![A]\!] \multimap T[\![B]\!] \xrightarrow{[\![A'<:A]\!]} [\![A']\!] \multimap T[\![B]\!] = [\![A' \multimap B]\!] \xrightarrow{\eta^T} T[\![A' \multimap B]\!]$,

where $g = [\![M]\!]^{\wedge}_{\mathrm{FV}}$ if $x \in \mathrm{FV}(M)$; otherwise g is the adjoint transpose of

$$[\![A]\!] \boxtimes [\![\Delta|_{\mathrm{FV}(M)}]\!] \xrightarrow{\mathsf{weak}} [\![\Delta|_{\mathrm{FV}(M)}]\!] \xrightarrow{[\![M]\!]_{\mathrm{FV}}} [\![B]\!]$$

$\boxed{\multimap.\mathrm{I}_2}$ $[\![(!\,\Delta, \Gamma)|_{\mathrm{FV}(\lambda x^A.M)}]\!] = ![\![\Delta|_{\mathrm{FV}(\lambda x^A.M)}]\!] \xrightarrow{\delta} !^{n+1}[\![\Delta|_{\mathrm{FV}(\lambda x^A.M)}]\!] \xrightarrow{!^n\, g} !^n([\![A]\!] \multimap T[\![B]\!])$
$\xrightarrow{[\![A'<:A]\!]} !^n([\![A']\!] \multimap T[\![B]\!]) = [\![!^n(A' \multimap B)]\!] \xrightarrow{\eta^T} T[\![!^n(A' \multimap B)]\!]$,

where g is defined as in the case $\multimap.\mathrm{I}_1$

$\boxed{\multimap.\mathrm{E}}$ $[\![(!\,\Delta, \Gamma_1, \Gamma_2)|_{\mathrm{FV}(MN)}]\!] \xrightarrow{\mathsf{con}} [\![(!\,\Delta, \Gamma_1)|_{\mathrm{FV}(M)}]\!] \boxtimes [\![(!\,\Delta, \Gamma_2)|_{\mathrm{FV}(N)}]\!]$
$\xrightarrow{[\![M]\!]_{\mathrm{FV}} \boxtimes [\![N]\!]_{\mathrm{FV}}} T([\![A]\!] \multimap T[\![B]\!]) \boxtimes T[\![C]\!] \xrightarrow{[\![C<:A]\!]} T([\![A]\!] \multimap T[\![B]\!]) \boxtimes T[\![A]\!] \xrightarrow{\mathrm{str}'}$
$T\bigl(([\![A]\!] \multimap T[\![B]\!]) \boxtimes T[\![A]\!]\bigr) \xrightarrow{T\mathrm{str}} TT\bigl(([\![A]\!] \multimap T[\![B]\!]) \boxtimes [\![A]\!]\bigr) \xrightarrow{\mathsf{ev}, \mu, \mu} T[\![B]\!]$

$\boxed{\boxtimes.\mathrm{I}}$ $[\![(!\,\Delta, \Gamma_1, \Gamma_2)|_{\mathrm{FV}(\langle M_1, M_2\rangle)}]\!] \xrightarrow{\mathsf{con}} [\![(!\,\Delta, \Gamma_1)|_{\mathrm{FV}(M_1)}]\!] \boxtimes [\![(!\,\Delta, \Gamma_2)|_{\mathrm{FV}(M_2)}]\!] \xrightarrow{[\![M_1]\!]_{\mathrm{FV}} \boxtimes [\![M_2]\!]_{\mathrm{FV}}}$
$T\,!^n[\![A_1]\!] \boxtimes T\,!^n[\![A_2]\!] \xrightarrow{\mathrm{str}', \text{ and then } \mathrm{str}, \mu} T(!^n[\![A_1]\!] \boxtimes !^n[\![A_2]\!]) \xrightarrow{\text{Lemma 5.3}} T\,!^n([\![A_1]\!] \boxtimes [\![A_2]\!])$

$\boxed{\boxtimes.\mathrm{E}}$ $[\![(!\,\Delta,\Gamma_1,\Gamma_2)|_{\mathrm{FV}(\mathtt{let}\,\langle x_1^{!^n A_1}, x_2^{!^n A_2}\rangle = M\,\mathtt{in}\,N)}]\!] \xrightarrow{\mathsf{con}} [\![(!\,\Delta,\Gamma_1)|_{\mathrm{FV}(M)}]\!] \boxtimes [\![(!\,\Delta,\Gamma_2)|_{\mathrm{FV}(N)}]\!]$

$\xrightarrow{[\![M]\!]_{\mathrm{FV}}\boxtimes\mathrm{id}} T\,!^n([\![A_1]\!] \boxtimes [\![A_2]\!]) \boxtimes [\![(!\,\Delta,\Gamma_2)|_{\mathrm{FV}(N)}]\!]$

$\xrightarrow{\mathrm{str}',\ \text{Lemma 5.3},} T\big(!^n[\![A_1]\!] \boxtimes !^n[\![A_2]\!] \boxtimes [\![(!\,\Delta,\Gamma_2)|_{\mathrm{FV}(N)}]\!]\big) \xrightarrow{[\![N]\!]_{\mathrm{FV}},(*)} T^2[\![A]\!] \xrightarrow{\mu} T[\![A]\!]$,

where, in $(*)$, weak is suitably applied in case x_1 or x_2 is not in $\mathrm{FV}(N)$

$\boxed{\top.\mathrm{I}}$ $\mathrm{I} \xrightarrow{\varphi'} !\,\mathrm{I} \xrightarrow{\delta,\mathsf{der}} !^n\,\mathrm{I} \xrightarrow{\eta^T} T\,!^n\,\mathrm{I}$

$\boxed{\top.\mathrm{E}}$ Similar

$\boxed{+.\mathrm{I}_1}$ $[\![\Delta|_{\mathrm{FV}(\mathtt{inj}_\ell^{A_2} M)}]\!] = [\![\Delta|_{\mathrm{FV}(M)}]\!] \xrightarrow{[\![M]\!]_{\mathrm{FV}}} T(!^n[\![A_1]\!]) \xrightarrow{T\,!^n\,\kappa_\ell} T\,!^n([\![A_1]\!] + [\![A_2]\!])$

$\xrightarrow{[\![A_2 <: A_2']\!]} T\,!^n([\![A_1]\!] + [\![A_2']\!])$

$\boxed{+.\mathrm{I}_2}$ Similar

$\boxed{+.\mathrm{E}}$ $[\![(!\,\Delta,\Gamma,\Gamma')|_{\mathrm{FV}(\mathtt{match}\,P\,\mathtt{with}\,(x_1^{!^n A_1} \mapsto M_1 | x_2^{!^n A_2} \mapsto M_2))}]\!]$

$\xrightarrow{\mathsf{con}} [\![(!\,\Delta,\Gamma)|_{\mathrm{FV}(P)}]\!] \boxtimes [\![(!\,\Delta,\Gamma')|_{\mathrm{FV}(M_1)\cup\mathrm{FV}(M_2)}]\!]$

$\xrightarrow{[\![P]\!]_{\mathrm{FV}}} T\,!^n([\![A_1]\!] + [\![A_2]\!]) \boxtimes [\![(!\,\Delta,\Gamma')|_{\mathrm{FV}(M_1)\cup\mathrm{FV}(M_2)}]\!]$

$\xrightarrow{\mathrm{str}'} T\Big(!^n([\![A_1]\!] + [\![A_2]\!]) \boxtimes [\![(!\,\Delta,\Gamma')|_{\mathrm{FV}(M_1)\cup\mathrm{FV}(M_2)}]\!]\Big)$

$\xrightarrow{\text{Lemma 5.3}} T\Big((!^n[\![A_1]\!] + !^n[\![A_2]\!]) \boxtimes [\![(!\,\Delta,\Gamma')|_{\mathrm{FV}(M_1)\cup\mathrm{FV}(M_2)}]\!]\Big)$

$\xrightarrow{\text{Lemma 5.3}} T\Big(!^n[\![A_1]\!] \boxtimes [\![(!\,\Delta,\Gamma')|_{\mathrm{FV}(M_1)\cup\mathrm{FV}(M_2)}]\!] + !^n[\![A_2]\!] \boxtimes [\![(!\,\Delta,\Gamma')|_{\mathrm{FV}(M_1)\cup\mathrm{FV}(M_2)}]\!]\Big)$

$\xrightarrow{T\big[[\![M_1]\!]_{\mathrm{FV}},[\![M_2]\!]_{\mathrm{FV}}\big],(*)} T^2[\![B]\!] \xrightarrow{\mu} TB$, where, in $(*)$, weak is applied if needed

$\boxed{\mathrm{rec}}$ $[\![(!\,\Delta,\Gamma)|_{\mathrm{FV}(\mathtt{letrec}\,f^{A\multimap B}x=M\,\mathtt{in}\,N)}]\!] \xrightarrow{\mathsf{con},\delta} !\,![\![\Delta|_{\mathrm{FV}(M)}]\!] \boxtimes [\![(!\,\Delta,\Gamma)|_{\mathrm{FV}(N)}]\!] \xrightarrow{!\,g}$

$![\![A \multimap B]\!] \boxtimes [\![(!\,\Delta,\Gamma)|_{\mathrm{FV}(N)}]\!] \xrightarrow{[\![N]\!]_{\mathrm{FV}},\mathsf{weak}} T[\![C]\!]$,

where $g : ![\![\Delta|_{\mathrm{FV}(M)}]\!] \to [\![A \multimap B]\!]$ is obtained as follows.

$![\![\Delta|_{\mathrm{FV}(M)}]\!] \boxtimes !([\![A]\!] \multimap T[\![B]\!]) \boxtimes [\![A]\!] \longrightarrow T[\![B]\!]$ is obtained as $[\![M]\!]_{\mathrm{FV}}$ (possibly with weak applied too);

$![\![\Delta|_{\mathrm{FV}(M)}]\!] \boxtimes !([\![A]\!] \multimap T[\![B]\!]) \longrightarrow [\![A]\!] \multimap T[\![B]\!]$ as its adjoint transpose; and then

$![\![\Delta|_{\mathrm{FV}(M)}]\!] \longrightarrow [\![A]\!] \multimap T[\![B]\!] = [\![A \multimap B]\!]$ via the fixed point operator fix in Definition 5.14.

Recall that weak denotes a unique map $X \to \mathrm{I}$ to the tensor unit I that is terminal (Lemma 5.1). The arrows $\mathsf{der}, \delta, \varphi, \varphi'$ and con are from Theorem 4.21 (see also Lemma 5.3). In the above some obvious elements are omitted: we write weak in place of $\mathsf{weak} \boxtimes \mathrm{id}$, $[\![M]\!]$ in place of $[\![\Delta \vdash M : A]\!]$, etc. We denote f's transpose by $f^\wedge$. The strength $X \boxtimes TY \to T(X \boxtimes Y)$ is denoted by str; str' stands for $TX \boxtimes Y \to T(X \boxtimes Y)$. For the rule (rec) we use the fixed point operator from Definition 5.14; note that the PER $[\![A]\!] \multimap T[\![B]\!]$ is admissible (Lemma 5.27).

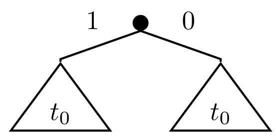
The proof of the following important lemma is rather complicated due to implicit linearity tracking. It is deferred to Appendix E.

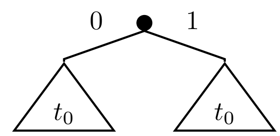
Lemma 5.35 (Interpretation of well-typed terms is well-defined)**.** *If Π, Π' are derivations of the same type judgment $\Delta \vdash M : A$, their interpretations are the same: $[\![\Pi]\!] = [\![\Pi']\!]$. Therefore the interpretation $[\![\Delta \vdash M : A]\!]$ of a derivable type judgment is well-defined.* □

To compare with operational semantics (introduced in §3.2), the interpretation $[\![\Delta \vdash M : A]\!] : [\![\Delta]\!] \to T[\![A]\!]$ thus obtained 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 `bit`. This is standard: for non-`bit` terms one will find distinguishing contexts of type `bit`.

In the following definition, the intuitions are:

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

$t_{\mathtt{tt}}$ = 

and $t_{\mathtt{ff}}$ = 

.

Definition 5.36 (Trees $t_0, t_{\mathtt{tt}}, t_{\mathtt{ff}}$, and test). Let c_{test} be the coalgebra

$$\mathrm{I} + \mathrm{I} + \mathrm{I} \xrightarrow{c_{\text{test}}} F_{\text{pbt}}(\mathrm{I} + \mathrm{I} + \mathrm{I})$$

whose transpose

$$[\![\mathtt{bit}]\!] \boxtimes (\mathrm{I} + \mathrm{I} + \mathrm{I}) \longrightarrow [\![0\text{-}\mathtt{qbit}]\!] \times (\mathrm{I} + \mathrm{I} + \mathrm{I})$$

is described as follows, using informal notations.

$$\begin{aligned}
\langle \mathtt{tt}, \kappa_1(*) \rangle &\longmapsto \langle 1, \kappa_2(*) \rangle \ , & \langle \mathtt{ff}, \kappa_1(*) \rangle &\longmapsto \langle 0, \kappa_2(*) \rangle \ , \\
\langle \mathtt{tt}, \kappa_2(*) \rangle &\longmapsto \langle 0, \kappa_2(*) \rangle \ , & \langle \mathtt{ff}, \kappa_2(*) \rangle &\longmapsto \langle 0, \kappa_2(*) \rangle \ , \\
\langle \mathtt{tt}, \kappa_3(*) \rangle &\longmapsto \langle 0, \kappa_2(*) \rangle \ , & \langle \mathtt{ff}, \kappa_3(*) \rangle &\longmapsto \langle 1, \kappa_2(*) \rangle \ .
\end{aligned}$$

By coinduction we obtain the following arrow $\overline{c_{\text{test}}}$.

$$\begin{tikzcd}
F_{\text{pbt}}(\mathrm{I} + \mathrm{I} + \mathrm{I}) \arrow[r, dashed, "F_{\text{pbt}}(\overline{c_{\text{test}}})"] & F_{\text{pbt}} R \\
(\mathrm{I} + \mathrm{I} + \mathrm{I}) \arrow[u, "c_{\text{test}}"] \arrow[r, dashed, "\overline{c_{\text{test}}}"'] & R \arrow[u, "\cong"', "r"]
\end{tikzcd}$$

Now the trees $t_0, t_{\mathtt{tt}}, t_{\mathtt{ff}} : \mathrm{I} \to R$ are defined by

$$t_0 := \overline{c_{\text{test}}} \circ \kappa_2 \ , \quad t_{\mathtt{tt}} := \overline{c_{\text{test}}} \circ \kappa_1 \ , \quad t_{\mathtt{ff}} := \overline{c_{\text{test}}} \circ \kappa_3 \ .$$

The arrow

$$\text{test} \ : \ \mathrm{I} \longrightarrow ([\![\mathtt{bit}]\!] \multimap R)$$

in $\mathbf{PER}_{\mathcal{Q}}$ is defined to be the adjoint transpose of $[t_{\mathtt{tt}}, t_{\mathtt{ff}}] : \mathrm{I} + \mathrm{I} \to R$.

Definition 5.37 (Operation prob **on trees).** For each arrow $t : \mathrm{I} \to R$ thought of as a tree, we define $\mathrm{prob}(t) \in (\mathbb{R} \cup \{\infty\})^2$ by:

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

For example, $\mathrm{prob}(t_{\mathtt{tt}}) = (1, 0)$ and $\mathrm{prob}(t_{\mathtt{ff}}) = (0, 1)$. See also Example 5.40 later.

The operation prob quotients the interpretation $[\![\Delta \vdash M : A]\!] : [\![\Delta]\!] \to T[\![A]\!]$ and yields a denotation relation $\Downarrow$, which is to be compared with the big-step operational semantics $\curlyvee$ (Definition 3.8). We note again that we swapped the notations $\curlyvee$ and $\Downarrow$ from the previous version [1].

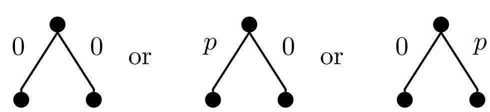
Definition 5.38 (Denotation relation $\Downarrow$). We define a relation $\Downarrow$ between closed $\mathtt{bit}$-terms M—i.e. those terms for which $\vdash M : \mathtt{bit}$ is derivable—and pairs (p, q) of real numbers, as follows. Such a term M gives rise to an arrow $\mathrm{tree}(M) : \mathrm{I} \to R$ in $\mathbf{PER}_{\mathcal{Q}}$ by:

$$\mathrm{tree}(M) \ := \ \left(\begin{array}{l} \mathrm{I} \xrightarrow{\cong} \mathrm{I} \boxtimes \mathrm{I} \xrightarrow{\mathrm{test} \boxtimes [\![\vdash M : \mathtt{bit}]\!]} ([\![\mathtt{bit}]\!] \multimap R) \boxtimes T[\![\mathtt{bit}]\!] \\ \qquad = \ ([\![\mathtt{bit}]\!] \multimap R) \boxtimes (([\![\mathtt{bit}]\!] \multimap R) \multimap R) \xrightarrow{\mathsf{ev}} R \end{array} \right) . \qquad (48)$$

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

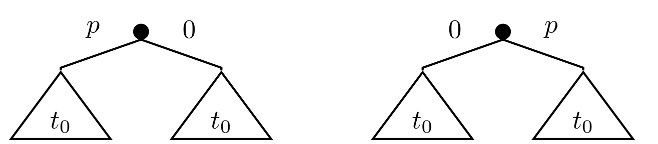
The infinite tree $\mathrm{tree}(M)$ always satisfies the following conditions:

- every branching is either



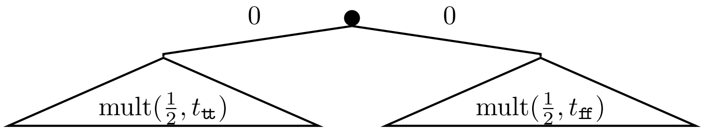
  for some $p \in (0, 1]$, and

- every non-zero branching is followed by the tree whose labels are all 0:

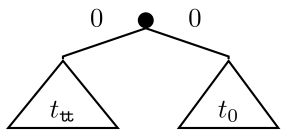


Intuitively, each zero-zero branch preceding a non-zero branch corresponds to measurement in the evaluation of the term M. We will see that the summation of labels on edges going down-left (going down-right) is the probability to get $\mathtt{tt}$ (to get $\mathtt{ff}$) by evaluating M. We note that when there are infinitely many measurements in the evaluation sequence of M, the tree associated to M may have infinitely many non-zero branching.

Example 5.39. Let H be the Hadamard matrix. The term $M := \mathtt{meas}_1^1(H(\mathtt{new}_{|0\rangle\langle 0|}))$ is a closed $\mathtt{bit}$-term; it measures the qubit $(|0\rangle + |1\rangle)/\sqrt{2}$. The associated tree $\mathrm{tree}(M)$ is



Indeed we have $M \Downarrow (1/2, 1/2)$. Similarly, $\mathrm{tree}(\mathtt{meas}_1^1(\mathtt{new}_{|0\rangle\langle 0|}))$ is

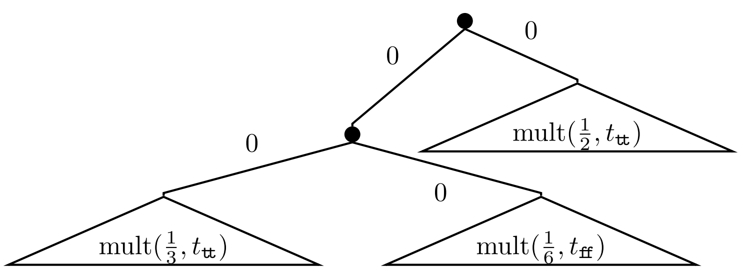


and we have $\mathtt{meas}_1^1(\mathtt{new}_{|0\rangle\langle 0|}) \Downarrow (1, 0)$. The zero-zero branches at the top node of $\mathrm{tree}(M)$ and $\mathrm{tree}(\mathtt{meas}_1^1(\mathtt{new}_{|0\rangle\langle 0|}))$ correspond to measurement in the evaluation sequences of M and $\mathtt{meas}_1^1(\mathtt{new}_{|0\rangle\langle 0|})$.

Example 5.40. Let M be the closed term of type $\mathtt{bit}$ given as follows.

1. The term M prepares two qubits $v = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $u = \frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$
2. The term M measures v.
3. If the result of the measurement of v is $\mathtt{ff}$, then M outputs $\mathtt{tt}$.
4. If the result of the measurement of v is $\mathtt{tt}$, then M measures the other qubit u and outputs the result.

The tree associated to M is



and we have $M \Downarrow (\frac{5}{6}, \frac{1}{6})$. The branch at the top node corresponds to measurement of v, and the branch of the node at the lower-left corresponds to the measurement of u.

As we observed in Example 5.39 and Example 5.40, $\mathrm{tree}(M)$ associated to closed $\mathtt{bit}$-term M has intentional information: we can see how measurement is done in the evaluation of M. For instance, $\mathrm{tree}(\mathtt{tt}) = t_{\mathtt{tt}}$ is different from $\mathrm{tree}(\mathtt{meas}_1^1(\mathtt{new}_{|0\rangle\langle 0|}))$.

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 $\vdash M : \mathtt{bit}$ is a tree in the result type R (Lemma 5.24) that reflects the evaluation tree of M. In this section, we show that the operation prob in Definition 5.37—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 $M \curlyvee (p, q)$.

We list several basic properties of our denotational semantics. Their proofs are found in Appendix F. 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 $[\![\Gamma \vdash E[M] : B]\!] : [\![\Gamma]\!] \to T[\![B]\!]$ is calculated by*

$$[\![\Gamma \vdash E[M] : B]\!] \quad = \quad \mu^T_{[\![B]\!]} \circ T[\![x : A \vdash E[x] : B]\!] \circ [\![\Gamma \vdash M : A]\!] \ . \qquad \square$$

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 (($\mathtt{meas}_1$–$\mathtt{meas}_4$) in Definition 3.7), then*

$$[\![\vdash M : A]\!] \quad = \quad [\![\vdash N : A]\!] \ .$$

Note that $\vdash N : A$ is derivable by Lemma 3.23. $\square$

If we allow reduction rules $\mathtt{meas}_1$–$\mathtt{meas}_4$, Lemma 6.2 is no longer correct. This is because our semantics contains some intentional information. In fact, tree($\mathtt{tt}$) is different from tree($\mathtt{meas}_1(\mathtt{new}_{|0\rangle\langle 0|})$) since the latter tree contains information about measurement in the evaluation of $\mathtt{meas}_1(\mathtt{new}_{|0\rangle\langle 0|})$ we observed in Example 5.39. Therefore, $[\![\vdash \mathtt{tt} : \mathtt{bit}]\!]$ is different from $[\![\vdash \mathtt{meas}_1(\mathtt{new}_{|0\rangle\langle 0|}) : \mathtt{bit}]\!]$. However, at the base type $\mathtt{bit}$, we can kill such intentionality by forgetting branching information in tree(M) by means of prob(_).

Lemma 6.3. *Let E be an evaluation context. If $\vdash E[\mathtt{meas}^{n+1}_i\, \mathtt{new}_\rho] : \mathtt{bit}$ is derivable and*

$$E[\langle \mathtt{tt}, \mathtt{new}_{\langle 0_i|\rho|0_i\rangle}\rangle] \Downarrow (p_0, q_0) \qquad\qquad E[\langle \mathtt{ff}, \mathtt{new}_{\langle 1_i|\rho|1_i\rangle}\rangle] \Downarrow (p_1, q_1),$$

then $E[\mathtt{meas}^{n+1}_i\, \mathtt{new}_\rho] \Downarrow (p_0 + p_1, q_0 + q_1)$.

Proof. By Lemma 6.1 we have

$$\begin{aligned}
&[\![\vdash E[\mathtt{meas}^{n+1}_i\, \mathtt{new}_\rho] : \mathtt{bit}\,]\!] \quad = \\
&\qquad \mu^T_{[\![\mathtt{bit}]\!]} \circ T[\![x : !\,\mathtt{bit} \boxtimes n\text{-}\mathtt{qbit} \vdash E[x] : \mathtt{bit}]\!] \circ [\![\vdash \mathtt{meas}^{n+1}_i\, \mathtt{new}_\rho : !\,\mathtt{bit} \boxtimes n\text{-}\mathtt{qbit}]\!] \\
&\qquad\qquad : \quad \mathrm{I} \longrightarrow T[\![\mathtt{bit}]\!] \ .
\end{aligned} \tag{49}$$

By the definition of the interpretation of $\mathtt{meas}_i^{n+1}$, the transpose of the interpretation

$$[\![\vdash \mathtt{meas}_i^{n+1}\,\mathtt{new}_\rho : !\,\mathtt{bit}\boxtimes n\text{-}\mathtt{qbit}\,]\!] \quad : \quad \mathrm{I}\longrightarrow T[\![!\,\mathtt{bit}\boxtimes n\text{-}\mathtt{qbit}]\!]$$

is equal to

$$\begin{aligned}
&(\,[\![!\,\mathtt{bit}]\!]\boxtimes[\![n\text{-}\mathtt{qbit}]\!]\,)\multimap R\\
&\xrightarrow{\cong}\quad \mathrm{I}\boxtimes\big(([\![!\,\mathtt{bit}]\!]\boxtimes[\![n\text{-}\mathtt{qbit}]\!])\multimap R\big)\\
&\xrightarrow{\langle[\![\mathtt{new}_\rho]\!]_{\mathrm{const}}\boxtimes\mathrm{id}\rangle}\quad [\![(n+1)\text{-}\mathtt{qbit}]\!]\boxtimes\big(([\![!\,\mathtt{bit}]\!]\boxtimes[\![n\text{-}\mathtt{qbit}]\!])\multimap R\big)\quad\xrightarrow{m}\quad R\ ,
\end{aligned}\tag{50}$$

where m is from (45). Recall that $[\![\mathtt{bit}]\!]\overset{\cong}{\Rightarrow}![\![\mathtt{bit}]\!]$; see Lemma 5.19. Under the following identifications

$$([\![!\,\mathtt{bit}]\!]\boxtimes[\![n\text{-}\mathtt{qbit}]\!])\multimap R\ \xrightarrow{\cong}\ ([\![n\text{-}\mathtt{qbit}]\!]\multimap R)^{\times 2}\quad\text{and}\quad R\ \xrightarrow{\cong}\ ([\![0\text{-}\mathtt{qbit}]\!]\times R)^{\times 2}$$

that are derived from Lemma 5.3, 5.19 and 5.24, the value of (50) at $\langle f,g\rangle:\mathrm{I}\to([\![n\text{-}\mathtt{qbit}]\!]\multimap R)^{\times 2}$ is

$$\begin{aligned}
&\Big\langle\ \mathrm{I}\xrightarrow{[\lambda x.\mathsf{Q}_0]}[\![0\text{-}\mathtt{qbit}]\!],\quad \mathrm{I}\xrightarrow{f\boxtimes[\lambda x.\mathsf{Pr}^{n+1}_{|0_i\rangle}\,\mathsf{Q}_\rho]}([\![n\text{-}\mathtt{qbit}]\!]\multimap R)\boxtimes[\![n\text{-}\mathtt{qbit}]\!]\xrightarrow{\mathsf{ev}}R,\\
&\qquad \mathrm{I}\xrightarrow{[\lambda x.\mathsf{Q}_0]}[\![0\text{-}\mathtt{qbit}]\!],\quad \mathrm{I}\xrightarrow{g\boxtimes[\lambda x.\mathsf{Pr}^{n+1}_{|1_i\rangle}\,\mathsf{Q}_\rho]}([\![n\text{-}\mathtt{qbit}]\!]\multimap R)\boxtimes[\![n\text{-}\mathtt{qbit}]\!]\xrightarrow{\mathsf{ev}}R\ \Big\rangle\\
&\qquad\qquad :\qquad \mathrm{I}\longrightarrow([\![0\text{-}\mathtt{qbit}]\!]\times R)^{\times 2}\ .
\end{aligned}\tag{51}$$

Here combinators like Q_0, Q_ρ and $\mathsf{Pr}_{|0_i\rangle}$ are from Definition 5.16 and 5.20; affine λ-terms like $\lambda x.\mathsf{Q}_0$ denote suitable elements of $A_{\mathcal{Q}}$ by combinatory completeness; and the arrow $[\lambda x.\mathsf{Q}_0]$ is the one in $\mathbf{PER}_{\mathcal{Q}}$ that is realized by $(\lambda x.\mathsf{Q}_0)\in A_{\mathcal{Q}}$. From Lemma 5.21, it is easy to see that the last arrow (51) is equal to

$$\begin{aligned}
&\big\langle\ [\lambda x.\mathsf{Q}_0],\ \mathsf{ev}\circ(f\boxtimes[\![\mathtt{new}_{\langle 0_i|\rho|0_i\rangle}]\!]_{\mathrm{const}}),\\
&\qquad\qquad [\lambda x.\mathsf{Q}_0],\ \mathsf{ev}\circ(g\boxtimes[\![\mathtt{new}_{\langle 1_i|\rho|1_i\rangle}]\!]_{\mathrm{const}})\ \big\rangle\ ,
\end{aligned}\tag{52}$$

where $[\![\_]\!]_{\mathrm{const}}$ is from Definition 5.32. Let us now define $f_0,f_1:[\![n\text{-}\mathtt{qbit}]\!]\to T[\![\mathtt{bit}]\!]$ to be the following arrows:

$$f_0\ :=\ \left(\begin{array}{l}[\![n\text{-}\mathtt{qbit}]\!]\xrightarrow{\cong}\mathrm{I}\boxtimes[\![n\text{-}\mathtt{qbit}]\!]\xrightarrow{\varphi'\boxtimes\mathrm{id}}!\,\mathrm{I}\boxtimes[\![n\text{-}\mathtt{qbit}]\!]\xrightarrow{!\,\kappa_\ell\boxtimes\mathrm{id}}![\![\mathtt{bit}]\!]\boxtimes[\![n\text{-}\mathtt{qbit}]\!]\\ \qquad\xrightarrow{[\![x:!\,\mathtt{bit}\boxtimes n\text{-}\mathtt{qbit}\vdash E[x]:\mathtt{bit}]\!]}T[\![\mathtt{bit}]\!]\end{array}\right)\ ,$$

$$f_1\ :=\ \left(\begin{array}{l}[\![n\text{-}\mathtt{qbit}]\!]\xrightarrow{\cong}\mathrm{I}\boxtimes[\![n\text{-}\mathtt{qbit}]\!]\xrightarrow{\varphi'\boxtimes\mathrm{id}}!\,\mathrm{I}\boxtimes[\![n\text{-}\mathtt{qbit}]\!]\xrightarrow{!\,\kappa_r\boxtimes\mathrm{id}}![\![\mathtt{bit}]\!]\boxtimes[\![n\text{-}\mathtt{qbit}]\!]\\ \qquad\xrightarrow{[\![x:!\,\mathtt{bit}\boxtimes n\text{-}\mathtt{qbit}\vdash E[x]:\mathtt{bit}]\!]}T[\![\mathtt{bit}]\!]\end{array}\right)\ ,$$

where $\varphi':\mathrm{I}\overset{\cong}{\Rightarrow}!\,\mathrm{I}$ is from Theorem 4.21. By (52), the transpose of (49) is equal to

$$\big\langle\ [\lambda x.\mathsf{Q}_0],\ g_0,\ [\lambda x.\mathsf{Q}_0],\ g_1\ \big\rangle\quad:\quad[\![\mathtt{bit}]\!]\multimap R\ \longrightarrow\ R$$

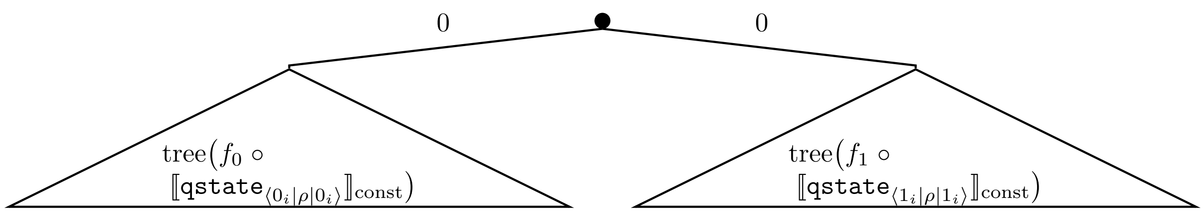
where we identified R with $(\llbracket 0\text{-}\mathtt{qbit} \rrbracket \times R)^{\times 2}$, and $g_k : \llbracket \mathtt{bit} \rrbracket \multimap R \longrightarrow R$ is the transpose of

$$\mathrm{I} \xrightarrow{[\lambda x.\mathsf{Q}_{\langle k_i | \rho | k_i \rangle}]} \llbracket n\text{-}\mathtt{qbit} \rrbracket \xrightarrow{f_k} T\llbracket \mathtt{bit} \rrbracket \ .$$

Hence,

$$\Big(\mathrm{I} \xrightarrow{\mathrm{tree}(E[\mathtt{meas}_i^{n+1}\, \mathtt{new}_\rho])} R \xrightarrow{\cong} (\llbracket 0\text{-}\mathtt{qbit} \rrbracket \times R)^{\times 2} \Big)$$
$$= \Big\langle\ [\lambda x.\mathsf{Q}_0],\ \mathrm{tree}\big(f_0 \circ \llbracket \mathtt{new}_{\langle 0_i | \rho | 0_i \rangle} \rrbracket_{\mathrm{const}}\big),\ [\lambda x.\mathsf{Q}_0],\ \mathrm{tree}\big(f_1 \circ \llbracket \mathtt{new}_{\langle 1_i | \rho | 1_i \rangle} \rrbracket_{\mathrm{const}}\big)\ \Big\rangle \ ,$$

where we abused the notation tree from (48). This means that the tree $\mathrm{tree}(E[\mathtt{meas}_i^{n+1}\, \mathtt{new}_\rho])$ can be illustrated as follows.



Therefore

$$\mathrm{prob}\Big(\mathrm{tree}\big(\llbracket \vdash E[\mathtt{meas}_i^{n+1}\, \mathtt{new}_\rho] : \mathtt{bit} \rrbracket\big)\Big)$$

is equal to

$$(0,0) + \mathrm{prob}(\mathrm{tree}(f_0 \circ \llbracket \mathtt{new}_{\langle 0_i | \rho | 0_i \rangle} \rrbracket_{\mathrm{const}})) + \mathrm{prob}(\mathrm{tree}(f_1 \circ \llbracket \mathtt{new}_{\langle 1_i | \rho | 1_i \rangle} \rrbracket_{\mathrm{const}}))$$

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

$$\begin{aligned} f_0 \circ \llbracket \mathtt{new}_{\langle 0_i | \rho | 0_i \rangle} \rrbracket_{\mathrm{const}} &= \llbracket \vdash E[\langle \mathtt{tt}, \mathtt{new}_{\langle 0_i | \rho | 0_i \rangle} \rangle] \rrbracket \ , \\ f_1 \circ \llbracket \mathtt{new}_{\langle 1_i | \rho | 1_i \rangle} \rrbracket_{\mathrm{const}} &= \llbracket \vdash E[\langle \mathtt{ff}, \mathtt{new}_{\langle 1_i | \rho | 1_i \rangle} \rangle] \rrbracket \ . \end{aligned}$$

Therefore, if

$$E[\langle \mathtt{tt}, \mathtt{new}_{\langle 0_i | \rho | 0_i \rangle} \rangle] \Downarrow (p_0, q_0) \qquad E[\langle \mathtt{ff}, \mathtt{new}_{\langle 1_i | \rho | 1_i \rangle} \rangle] \Downarrow (p_1, q_1),$$

then $E[\mathtt{meas}_i^{n+1}\, \mathtt{new}_\rho] \Downarrow (p_0 + p_1, q_0 + q_1)$. □

We can similarly prove the following lemma.

Lemma 6.4. *Let E be an evaluation context. If $\vdash E[\mathtt{meas}_1^1 \mathtt{new}_\rho] : \mathtt{bit}$ is derivable and*

$$E[\mathtt{tt}] \Downarrow (p_0, q_0) \qquad\qquad E[\mathtt{ff}] \Downarrow (p_1, q_1),$$

then $E[\mathtt{meas}_1^1\, \mathtt{new}_\rho] \Downarrow (\langle 0 \mid \rho \mid 0 \rangle p_0 + \langle 1 \mid \rho \mid 1 \rangle p_1, \langle 0 \mid \rho \mid 0 \rangle q_0 + \langle 1 \mid \rho \mid 1 \rangle q_1)$.

Our soundness result is restricted to closed $\texttt{bit}$-terms because we do not know how to relate evaluation results of closed terms $\vdash M : A$ with its interpretation $[\![\vdash M : A]\!] \colon I \to T[\![A]\!]$ when A is not $\texttt{bit}$.

Theorem 6.5 (Soundness)**.** *For any closed* $\texttt{bit}$*-term* M *(meaning that* $\vdash M : \texttt{bit}$ *is derivable), and for any* $k \in \mathbb{N}$*,*

$$M \curlyvee^k (p, q) \quad \textit{and} \quad M \Downarrow (p', q') \quad \textit{imply} \quad (p, q) \leq (p', q') \ .$$

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 $\texttt{tt}$ nor $\texttt{ff}$, then $p, q = 0$ and the statement is true. If $M \equiv \texttt{tt}$, then $p = p' = 1$ and $q = q' = 0$. If $M \equiv \texttt{ff}$, 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 $[\![\vdash M : \texttt{bit}]\!]$ is equal to $[\![\vdash N : \texttt{bit}]\!]$ by Lemma 6.2. Therefore, $M \Downarrow (p', q')$ if and only if $N \Downarrow (p', q')$. Since $M \curlyvee^k (p, q)$ if and only if $N \curlyvee^{k-1} (p, q)$, we obtain $p \leq p'$ and $q \leq q'$ from the induction hypothesis. If M is of the form $E[\texttt{meas}_i^{n+1}\, \texttt{new}_\rho]$, then we have $M \curlyvee^k (p_0 + p_1, q_0 + q_1)$ where $E[\langle \texttt{tt}, \texttt{new}_{\langle 0_i | \rho | 0_i \rangle} \rangle] \curlyvee^{k-1} (p_0, q_0)$ and $E[\langle \texttt{ff}, \texttt{new}_{\langle 1_i | \rho | 1_i \rangle} \rangle] \curlyvee^{k-1} (p_1, q_1)$. By Lemma 6.3, if

$$E[\langle \texttt{tt}, \texttt{new}_{\langle 0_i | \rho | 0_i \rangle} \rangle] \Downarrow (p'_0, q'_0) \qquad\qquad E[\langle \texttt{ff}, \texttt{new}_{\langle 1_i | \rho | 1_i \rangle} \rangle] \Downarrow (p'_1, q'_1),$$

then $E[\texttt{meas}_i^{n+1}\, \texttt{new}_\rho] \Downarrow (p'_0 + p'_1, q'_0 + q'_1) \geq (p_0 + p_1, q_0 + q_1)$.

We can similarly show the statement when the reductions are due to the $(\texttt{meas}_3)$ and $(\texttt{meas}_4)$ rules in Definition 3.7 by Lemma 6.4. □

We shall now show the other direction: if $M \curlyvee (p, q)$ and $M \Downarrow (p', q')$, then $(p', q') \leq (p, q)$. Our proof employs the techniques of logical relations (see e.g. [79, 80]) and $\top\top$-lifting (see e.g. [74, 81]). We write $\mathrm{Val}(A)$ for the set of closed values of type A and $\mathrm{ClTerm}(A)$ for the set of closed terms of type A. We write $\mathrm{EC}(A)$ for the set of evaluation contexts E such that $x : A \vdash E[x] : \texttt{bit}$ is derivable.

Firstly, we introduce a relation $\lessdot$ between $\mathbf{PER}_\mathcal{Q}(\mathrm{I}, T[\![\texttt{bit}]\!])$ and $\mathrm{ClTerm}(\texttt{bit})$. It is defined by by

$$t \lessdot M \quad \overset{\text{def.}}{\iff} \quad \text{if } M \curlyvee (p, q) \text{ then } \mathrm{prob}(\mathrm{tree}(t)) \leq (p, q) \ .$$

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

$$S \ \subseteq \ \mathbf{PER}_\mathcal{Q}(\mathrm{I}, [\![A]\!]) \times \mathrm{Val}(A) \ ,$$

we define a relation $S^\top \subseteq \mathbf{PER}_\mathcal{Q}([\![A]\!], T[\![\texttt{bit}]\!]) \times \mathrm{EC}(A)$ by

$$S^\top \ := \ \left\{ (k, E) \,\middle|\, \forall (t, V) \in S.\ k \circ t \lessdot E[V] \right\} \ ;$$

and we define a relation $S^{\top\top} \subseteq \mathbf{PER}_\mathcal{Q}(\mathrm{I}, T[\![A]\!]) \times \mathrm{ClTerm}(A)$ by

$$S^{\top\top} \ := \ \left\{ (t, M) \,\middle|\, \forall (k, E) \in S^\top.\ \mu^T_{[\![\texttt{bit}]\!]} \circ Tk \circ t \lessdot E[M] \right\} \ .$$

This operation of $\top\top$-lifting is applied to the following relation $R_A \subseteq \mathbf{PER}_{\mathcal{Q}}(\mathrm{I}, [\![A]\!]) \times \mathrm{Val}(A)$. It is inductively defined for each type A.

$$\begin{aligned}
R_{\mathrm{I}} &:= \big\{(\mathrm{id}_{\mathrm{I}}, *)\big\} \\
R_{n\text{-}\mathtt{qbit}} &:= \big\{ \big([\![\mathtt{new}_\rho]\!]_{\mathrm{const}}, \mathtt{new}_\rho\big) \,\big|\, \rho \in \mathrm{DM}_{2^n} \big\} \\
R_{A \boxtimes B} &:= \big\{ \big(t \boxtimes s, \langle V, W\rangle\big) \,\big|\, (t, V) \in R_A \text{ and } (s, W) \in R_B \big\} \\
R_{A \multimap B} &:= \big\{ (t, V) \,\big|\, \forall (s, W) \in R_A.\, \big(\mathsf{ev}_{[\![A]\!],[\![B]\!]} \circ (t \boxtimes s),\, V\,W\big) \in R_B^{\top\top} \big\} \\
R_{!\,A} &:= \big\{ (!\,t \circ \varphi', V) \,\big|\, (t, V) \in R_A \big\} \\
R_{A+B} &:= \big\{ \big(\kappa_\ell \circ t, \mathtt{inj}_\ell^{B'}(V)\big) \,\big|\, (t, V) \in R_A \text{ and } B' <: B \big\} \\
&\quad\ \cup \big\{ \big(\kappa_r \circ t, \mathtt{inj}_r^{A'}(V)\big) \,\big|\, (t, V) \in R_B \text{ and } A' <: A \big\}
\end{aligned} \tag{53}$$

Here $\varphi' : \mathrm{I} \overset{\cong}{\Rightarrow} !\,\mathrm{I}$ is from Theorem 4.21. In order to prove the *basic lemma* for the logical relation $\{R_A\}_{A:\mathrm{type}}$, we show some properties of R_A.

Lemma 6.6. 1. *If (t, V) is in R_A, then $(\eta^T_{[\![A]\!]} \circ t, V)$ is in $R_A^{\top\top}$.*

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. [74].

Lemma 6.7. *Let M be a closed term of type A.*

1. $([\perp], M) \in R_A^{\top\top}$.
2. *If there exists a sequence of realizers $a_1 \sqsubseteq a_2 \sqsubseteq \cdots$ of arrows $[a_1], [a_2], \ldots$ in $\mathbf{PER}_{\mathcal{Q}}(\mathrm{I}, T[\![A]\!])$, such that $([a_n], M) \in R_A^{\top\top}$ for each n, then we have $\big([\bigvee_{n\geq 1} a_n],\, M\big) \in R_A^{\top\top}$.* □

Theorem 6.8 (Basic Lemma)**.** *Let M be a term such that $x_1 : A_1, \cdots, x_n : A_n \vdash M : A$ is derivable. If (t_i, V_i) is in R_{A_i} for each $i \in [1, n]$, then the pair*

$$\big(\, [\![x_1 : A_1, \cdots, x_n : A_n \vdash M : A]\!] \circ (t_1 \boxtimes \cdots \boxtimes t_n), \quad M[V_1/x_1, \cdots, V_n/x_n] \,\big)$$

is in $R_A^{\top\top}$.

Proof. By induction on M. When $M \equiv x_i$, we have

$$[\![x_1 : A_1, \cdots, x_n : A_n \vdash M : A]\!] \circ (t_1 \boxtimes \cdots \boxtimes t_n) = \eta^T_{[\![A]\!]} \circ [\![A_i <: A]\!] \circ t_i. \tag{54}$$

By (2) in Lemma 6.6, the pair $([\![A_i <: A]\!] \circ t_i, V_i)$ is in R_A. Therefore, by (1) in Lemma 6.6 and (54), we see that

$$\big(\, [\![x_1 : A_1, \cdots, x_n : A_n \vdash M : A]\!] \circ (t_1 \boxtimes \cdots \boxtimes t_n),\ V_i \,\big)$$

is in $R_A^{\top\top}$. When M is a constant, see Lemma Appendix F.11–Appendix F.15. When M is an application $!\,\Delta, \Gamma_1, \Gamma_2 \vdash N_0 N_1 : B$ for $!\,\Delta, \Gamma_1 \vdash N_0 : A \multimap B$ and $!\,\Delta, \Gamma_2 \vdash N_1 : A \multimap B$, we suppose that Δ, Γ_1, Γ_2 are empty lists for simplicity. Generalization is straightforward. Since

$$\big(\mu^T_{[\![\mathtt{bit}]\!]} \circ Tk \circ \mathsf{ev}_{[\![A]\!],[\![B]\!]} \circ (t \boxtimes [\![A]\!]),\ E[V[\_]]\big) \in R_A^{\top}$$

for any $(t, V) \in R_{A \multimap B}$ and $(k, E) \in R_B^{\top}$, we have

$$\big(\mu^T_{[\![\mathtt{bit}]\!]} \circ T\mu^T_{[\![\mathtt{bit}]\!]} \circ TTk \circ T\mathsf{ev}_{[\![A]\!],[\![B]\!]} \circ \mathrm{str} \circ ([\![A \multimap B]\!] \boxtimes ([\![\vdash N_1 : A]\!])), \\ E[[\_]N_1] \big) \quad \in \; R^{\top}_{A \multimap B}$$

for any $(k, E) \in R_B^{\top}$. Therefore,

$$\mu^T_{[\![\mathtt{bit}]\!]} \circ T\mu^T_{[\![\mathtt{bit}]\!]} \circ TT\mu^T_{[\![\mathtt{bit}]\!]} \circ TTTk \circ TT\mathsf{ev}_{[\![A]\!],[\![B]\!]} \circ T\mathrm{str} \circ \mathrm{str}' \circ \\ ([\![\vdash N_0 : A \multimap B]\!] \boxtimes [\![\vdash N_1 : A]\!]) \lessdot E[N_0 N_1]$$

holds for any $(k, E) \in R_B^{\top}$, and we obtain

$$([\![\vdash N_0 N_1 : B]\!], N_0 N_1) \in R_B^{\top\top}.$$

When M is a lambda abstraction $x_1 : A_1, \cdots, x_n : A_n \vdash \lambda x^A.N : A' \multimap B$, we suppose that x is the only free variable of N and $n = 0$ for simplicity. Also in this case, generalization is straightforward. We have

$$\big([\![x : A \vdash M : B]\!] \circ [\![A' <: A]\!] \circ t,\, M[V/x] \big) \in R_B^{\top\top}$$

for any $(t, V) \in R_{A'}$. By the definition of $R_{A \multimap B}$,

$$\big(g,\, \lambda x^A.M\big) \in R_{A \multimap B}$$

where g is the adjoint transpose of $[\![x : A \vdash M : B]\!] \circ [\![A' <: A]\!]$. Hence, by Lemma 6.6, $\big([\![M]\!],\, \lambda x^A.M\big)$ is in $R_{A \multimap B}$. When M is $\mathtt{letrec}\ f^A\, x = N\ \mathtt{in}\ L$, the statement follows from Lemma 6.7. Note that the interpretation of $\mathtt{letrec}\ f^A\, x = N\ \mathtt{in}\ L$ is given by the least upper bound of a sequence of realizers. The other cases are easy. □

Corollary 6.9 (Adequacy)**.** *For a closed term* $\vdash M : \mathtt{bit}$*, we have*

$$M \curlyvee (p, q) \iff M \Downarrow (p, q).$$

Proof. We suppose that $M \curlyvee (p, q)$ and $M \Downarrow (p', q')$. By Theorem 6.5, we have $(p, q) \leq (p', q')$ on the one hand. On the other hand, by Theorem 6.8 (consider its special case where M is closed), we have $\big([\![\vdash M : \mathtt{bit}]\!], M\big) \in R_{\mathtt{bit}}^{\top\top}$. Since $(\eta^T_{[\![\mathtt{bit}]\!]}, [\_])$ is easily shown to be in $R_{\mathtt{bit}}^{\top}$, we obtain $[\![\vdash M : \mathtt{bit}]\!] \lessdot M$. Hence $(p', q') \leq (p, q)$. □

7. Conclusions and Future Work

We presented a concrete denotational model of a quantum λ-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 Figure 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 [82].

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].[13] In doing so, our current use of the continuation monad T (see §5) is a technical burden; it seems we need such continuation monads not only for quantum effects (in the current paper) but also for various computational effects (in general). Endowing realizers with an explicit notion of *memory* (or *state*) [77, 28]—in a systematic manner using *coalgebraic component calculus* [83, 84]—seems to be a potent alternative to use of continuation monads.

Acknowledgments

Thanks are due to Peter Selinger for a number of insightful comments; to Kazuyuki Asada, Ugo Dal Lago, Claudia Faggian, Bart Jacobs, Prakash Panangaden, Peter Selinger, Phil Scott, Benoît Valiron, Mingsheng Ying and the anonymous reviewers (for the earlier version [1] as well as for the current version) for useful discussions and comments; and to Andy Pitts for his ETAPS'07 talk that inspired the current work.

[13]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 Lemma 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 $\|\_\|_{\mathrm{tr}}$ and $\|\_\|_{\mathsf{Fr}}$). Given a matrix $A \in M_m$, its *trace norm* $\|\_\|_{\mathrm{tr}}$ is defined by

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

Here the matrix $A^{\dagger}A$ is positive hence its square root is well defined (see e.g. [38, §2.1.8]). In particular, we have

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

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

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

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 $\|\_\|_{\mathrm{tr}}$ is called the *trace distance* and heavily used in [38, §9.2].

Lemma Appendix A.2.

1. *For each matrix $A \in M_m$ we have $\|A\|_{\mathsf{Fr}} \le \|A\|_{\mathrm{tr}} \le m\|A\|_{\mathsf{Fr}}$; therefore the two norms induce the same topology on the set M_m.*
2. *Both norms $\|\_\|_{\mathsf{Fr}}$ and $\|\_\|_{\mathrm{tr}}$ are complete.*
3. *The subset $\mathrm{DM}_m \subseteq M_m$ is closed with respect to both norms $\|\_\|_{\mathsf{Fr}}$ and $\|\_\|_{\mathrm{tr}}$.*

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

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

which is obvious.

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

3. Let $(\rho_k)_{k\in\mathbb{N}}$ be a Cauchy sequence in DM_m. We show that $\lim_k \rho_k$ belongs to DM_m. It is positive because the mapping $\langle v|\_|v\rangle : \mathrm{DM}_m \to \mathbb{C}$ is continuous with respect to $\|\_\|_{\mathsf{Fr}}$ (hence also to $\|\_\|_{\mathrm{tr}}$). Similarly, continuity of $\mathsf{tr}(\_)$ yields that $\mathsf{tr}(\lim_k \rho_k) \le 1$. □

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 $\mathsf{tr}(\_) : \mathrm{DM}_m \to \mathbb{C}$. On the one hand, the Frobenius norm $\|\_\|_{\mathsf{Fr}}$ is useful since many functions—such as $\mathsf{tr}(\_)$—are obviously continuous with respect to it. On the other hand, the trace norm $\|\_\|_{\mathrm{tr}}$ 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 Definition 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 $n \le n'$, we have

$$\|\rho_{n'} - \rho_n\|_{\mathrm{tr}} \overset{(*)}{=} \mathsf{tr}(\rho_{n'} - \rho_n) = \mathsf{tr}(\rho_{n'}) - \mathsf{tr}(\rho_n) \ ; \tag{A.2}$$

where $(*)$ holds since $\rho_{n'} - \rho_n$ is positive (see (A.1)). Now observe that the sequence $(\mathsf{tr}(\rho_n))_{n\in\mathbb{N}}$ 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 $\|\_\|_{\mathrm{tr}}$. By Lemma Appendix A.2, it has a limit $\lim_n \rho_n$ in DM_m. □

LEMMA (LEMMA 2.5, REPEATED). The relation $\sqsubseteq$ in Definition 2.4 is indeed a partial order. Moreover it is an ω-CPO: any increasing ω-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 ω-CPO is proved in [7, Proposition 3.6] via the translation into quadratic forms. Here we present a proof using norms. By Lemma Appendix A.3, an increasing ω-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

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

the equality is due to the continuity of $\langle v|\_|v\rangle : \mathrm{DM}_m \to \mathbb{C}$. 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 (PROPOSITION 2.13, REPEATED). The order $\sqsubseteq$ on $\mathrm{QO}_{m,n}$ (Definition 2.12) is an ω-CPO.

PROOF. Let $(\mathcal{E}_k)_{k\in\mathbb{N}}$ be an increasing chain in $\mathrm{QO}_{m,n}$. We define $\mathcal{E}$ to be its "pointwise supremum": for each $\rho \in \mathrm{DM}_m$,

$$\mathcal{E}(\rho) \ := \ \sup_{k\to\infty} \mathcal{E}_k(\rho) \ \overset{(*)}{=} \ \lim_{k\to\infty} \mathcal{E}_k(\rho) \tag{A.4}$$

where the supremum is taken in the ω-CPO DM_n (Lemma 2.5). In the proof of Lemma 2.5 we exhibited that the supremum is indeed the limit ((∗) above). We claim that this $\mathcal{E}$ is the supremum of the chain $(\mathcal{E}_k)_{k\in\mathbb{N}}$.

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

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

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 $\|\_\|_{\mathrm{tr}}$. To prove complete positivity of $\mathcal{E}$, one can use Choi's characterization of complete positive maps (see [7, Theorem 6.5]). The operations involved in the characterization are all continuous, hence one can conclude complete positivity of $\mathcal{E}$ from that of $\mathcal{E}_k$.

It remains to show that $\mathcal{E}$ is indeed the least upper bound. This is obvious since $\sqsubseteq$ on $\mathrm{QO}_{m,n}$ is a pointwise extension of $\sqsubseteq$ on density matrices. □

Appendix B. Proofs for §3.4

Appendix B.1. Proof of Lemma 3.6

Proof. We let

- the set of evaluation contexts that is defined in Definition 3.5 denoted by EV, and
- that which is defined in Lemma 3.6 denoted by $\overline{\mathrm{EV}}$.

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

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

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

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

The other direction $\overline{\mathrm{EV}} \subseteq \mathrm{EV}$ is similar; we present only one case. For $D \equiv D'M \in \overline{\mathrm{EV}}$, by the induction hypothesis we have $D' \in \mathrm{EV}$; moreover $[\_]M \in \mathrm{EV}$. Therefore by the fact 1 above, $D \equiv D'M = [D']M$ belongs to EV. □

Appendix B.2. Proof of Lemma 3.15

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

$$A <: A' \quad \text{and} \quad A' <: A'' \qquad \Longrightarrow \qquad A <: A''$$

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

$$\frac{B <: B' \quad C <: C' \quad n = 0 \Rightarrow n' = 0}{A \equiv\, !^n(B \boxtimes C) <: !^{n'}(B' \boxtimes C') \equiv A'} \ (\boxtimes) \ . \tag{B.1}$$

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

$$\frac{B' <: B'' \quad C' <: C'' \quad n' = 0 \Rightarrow n'' = 0}{A' \equiv\, !^{n'}(B' \boxtimes C') <: !^{n''}(B'' \boxtimes C'') \equiv A''} \ (\boxtimes) \ . \tag{B.2}$$

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 ($\boxtimes$) rule we derive $A <: A''$.

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

$$\frac{B_1 <: A_1 \quad A_2 <: B_2 \quad n = 0 \Rightarrow m = 0}{A \equiv\, !^n(A_1 \multimap A_2) <: !^m(B_1 \multimap B_2) \equiv B} \ (\multimap) \ .$$

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

3. By cases on the outermost type constructor in A (ignoring !). Assume it is $\multimap$, with $A \equiv\, !^k(B \multimap C)$. 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

$$\frac{B <: B \quad C <: C \quad n + k = 0 \Rightarrow m + k = 0}{!^{n+k}(B \multimap C) <: !^{m+k}(B \multimap C)} \ (\multimap)$$

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

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

This A_0 is clearly a supremum of A_1 and A_2.

Assume n-$\texttt{qbit} <: A_1$ and n-$\texttt{qbit} <: A_2$. Then A_1 and A_2 must both be n-$\texttt{qbit}$, and $A_0 :\equiv n$-$\texttt{qbit}$ is the infimum.

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

Assume $A_1 <: B \multimap C$ and $A_2 <: B \multimap C$. Then the rules in (11) force that $A_1 \equiv !^{n_1}(B_1 \multimap C_1)$ and $A_2 \equiv !^{n_2}(B_2 \multimap C_2)$, with $B <: B_1$, $B <: B_2$, $C_1 <: C$ and $C_2 <: C$. Since the complexity of B or C is smaller than that of $B \multimap C$ 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

$$A_0 \;:\equiv\; \begin{cases} !(B_0 \multimap C_0) & \text{if } n_1 \neq 0 \text{ and } n_2 \neq 0, \\ B_0 \multimap C_0 & \text{otherwise.} \end{cases}$$

is easily shown to be a supremum of A_1, A_2. The other cases are similar. □

Appendix B.3. Proof of Lemma 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 $(\multimap.\mathrm{I}_2)$ rule the one applied last, and $\Delta = (!\,\Delta_0, \Gamma)$, $M \equiv \lambda x^B.N$, $A \equiv !^n(B' \multimap C)$.

$$\frac{x : B, !\,\Delta_0, \Gamma \vdash N : C \qquad \mathrm{FV}(N) \subseteq |\Delta_0| \cup \{x\} \qquad B' <: B}{!\,\Delta_0, \Gamma \vdash \lambda x^B.N : !^n(B' \multimap C)}\ (\multimap.\mathrm{I}_2)$$

Since $\Delta' <: \Delta = (!\,\Delta_0, \Gamma)$, we have $\Delta' = (\Delta_0', \Gamma')$ with $\Delta_0' <: !\,\Delta_0$ and $\Gamma' <: \Gamma$. Furthermore, by Lemma 3.15.4, Δ_0' must be of the form $\Delta_0' = !\,\Delta_0''$ with some Δ_0''. Thus $\Delta' = (!\,\Delta_0'', \Gamma')$. Similarly, from the assumption that $A \equiv !^n(B' \multimap C) <: A'$ we have $A' \equiv !^m(B'' \multimap C'')$ with $B'' <: B'$, $C <: C''$ and $n = 0 \Rightarrow m = 0$.

Now we have $(x : B, !\,\Delta_0'', \Gamma') <: (x : B, !\,\Delta_0, \Gamma)$. Using the induction hypothesis we obtain $\Vdash x : B, !\,\Delta_0'', \Gamma' \vdash N : C''$. Since $\mathrm{FV}(N) \subseteq |\Delta_0| \cup \{x\} = |\Delta_0''| \cup \{x\}$, the $(\multimap.\mathrm{I}_2)$ rule can be applied.

$$\frac{x : B, !\,\Delta_0'', \Gamma' \vdash N : C'' \qquad \mathrm{FV}(N) \subseteq |\Delta_0''| \cup \{x\} \qquad B'' <: B}{!\,\Delta_0'', \Gamma' \vdash \lambda x^B.N : !^m(B'' \multimap C'')}\ (\multimap.\mathrm{I}_2)$$

To obtain $B'' <: B$ we used transitivity (Lemma 3.15.1). This derives $\Delta' \vdash \lambda x^B.N : A'$. □

Appendix B.4. Proof of Lemma 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 Lemma 3.15.4.

If V is some constant (i.e. $\texttt{new}, \texttt{meas}_i^{n+1}, U, \texttt{cmp}_{m,n}, or \texttt{new}_\rho$), or if $V \equiv *$, $\mathrm{FV}(V)$ is empty.

If $V \equiv \lambda x^B.M$, the type judgment $\Delta \vdash V : !A$ must be derived by the $(\multimap.\mathrm{I}_2)$ rule:

$$\frac{x : B, !\Delta_0, \Gamma \vdash M : C \qquad \mathrm{FV}(M) \subseteq |\Delta_0| \cup \{x\} \qquad B' <: B}{!\Delta_0, \Gamma \vdash \lambda x^B.M : !^n(B' \multimap C)}\ (\multimap.\mathrm{I}_2)$$

with $\Delta = (!\Delta_0, \Gamma)$ and $A \equiv !^{n-1}(B \multimap C)$. Now we have $\mathrm{FV}(V) = \mathrm{FV}(M) \setminus \{x\} \subseteq \big(|\Delta_0| \cup \{x\}\big) \setminus \{x\} = |\Delta_0|$, from which the claim follows.

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

$$\frac{!\Delta_0, \Gamma_1 \vdash V_1 : !^n A_1 \qquad !\Delta_0, \Gamma_2 \vdash V_2 : !^n A_2}{!\Delta_0, \Gamma_1, \Gamma_2 \vdash \langle V_1, V_2 \rangle : !^n(A_1 \boxtimes A_2)}\ (\boxtimes.\mathrm{I}), (\dagger)$$

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

$$(!\Delta_0, \Gamma_1)|_{\mathrm{FV}(V_1)} \ = \ !\Delta_1 \ , \qquad (!\Delta_0, \Gamma_2)|_{\mathrm{FV}(V_2)} \ = \ !\Delta_2$$

for some Δ_1 and Δ_2 (here $(!\Delta_0, \Gamma_1)|_{\mathrm{FV}(V_1)}$ denotes the suitable restriction of a context). The claim follows immediately. The cases where $V \equiv \mathtt{inj}_\ell^B\, V'$ or $V \equiv \mathtt{inj}_r^A\, V'$ are similar.

2. By induction on the construction of a value V.

If $V \equiv x$, a variable, then the claim follows easily from Lemma 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 $A \equiv !^n(B' \multimap C)$ with $n \geq 1$, the type judgment $!\Delta, \Gamma \vdash V : A$ is derived by the $(\multimap.\mathrm{I}_2)$ rule and it also derives $!\Delta, \Gamma \vdash V : !A$. If A is of the form $A \equiv B' \multimap C$, the derivation of $!\Delta, \Gamma \vdash V : A$ looks as follows.

$$\frac{x : B, !\Delta, \Gamma \vdash M : C \qquad B' <: B}{!\Delta, \Gamma \vdash \lambda x^B.M : B' \multimap C}\ (\multimap.\mathrm{I}_1) \tag{B.3}$$

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

$$\frac{x : B, !\Delta, \Gamma \vdash M : C \qquad \mathrm{FV}(M) \subseteq |\Delta| \cup \{x\} \qquad B' <: B}{!\Delta, \Gamma \vdash \lambda x^B.M : !(B' \multimap C)}\ (\multimap.\mathrm{I}_2)\ .$$

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

Finally, the cases where $V \equiv \mathtt{inj}_\ell^B\, V'$ or $V \equiv \mathtt{inj}_r^A\, V'$ are easy using the induction hypothesis. This concludes the proof. □

Appendix B.5. Proof of Lemma 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 (Lemma 3.17) and the weakening rule (Lemma 3.19.3). The third rule (Subst_3) follows from (Subst_2) via Lemma 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 (Lemma 3.6) in place of Definition 3.5. □

Appendix B.6. Proof of Lemma 3.23

PROOF. By induction on the construction of an evaluation context E. The step cases where $E \not\equiv [\_]$ are easy, since the local character of the typing rules of Hoq (for a rule to be applied, the terms in the assumptions can be anything). For the base case (i.e. $E \equiv [\_]$) we prove by cases according to Definition 3.7.

In the case where $M \longrightarrow_p N$ is by the ($\multimap$) rule of Definition 3.7, 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.

$$\dfrac{\dfrac{\begin{matrix}\vdots\\ x : A', !\,\Delta', \Gamma_1 \vdash M' : A \quad B <: A'\end{matrix}}{!\,\Delta', \Gamma_1 \vdash \lambda x^{A'}.M' : B \multimap A}\,(\multimap.\mathrm{I}_1) \qquad \begin{matrix}\vdots\\ !\,\Delta', \Gamma_2 \vdash V : C\end{matrix} \qquad C <: B}{!\,\Delta', \Gamma_1, \Gamma_2 \vdash (\lambda x^{A'}.M')V : A}\,(\multimap.\mathrm{E}) \tag{B.4}$$

Here $\Delta = (\Delta', \Gamma_1, \Gamma_2)$. We have $C <: B <: A'$, thus $C <: A'$ (Lemma 3.15.1). Using the monotonicity rule (Lemma 3.17) we have $\Vdash !\,\Delta', \Gamma_2 \vdash V : A'$. Combining this with the top-left judgment in (B.4) via the (Subst$_3$) rule in Lemma 3.22, we obtain $\Vdash !\,\Delta', \Gamma_1, \Gamma_2 \vdash M'[V/x] : A$, which is our goal.

The cases of the $(\boxtimes), (\top)$, and $(+_i)$ rules are similar, where we rely on the substitution rules in Lemma 3.22.

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

$$\begin{aligned} N &\equiv N'\bigl[(\lambda x^B.\,\texttt{letrec}\, f^{B\multimap C} x = M' \,\texttt{in}\, M')/f\bigr] \\ &\equiv N'\bigl[(\lambda z^B.\,\texttt{letrec}\, f^{B\multimap C} x = M' \,\texttt{in}\, M'[z/x])\,/\,f\bigr] \ . \end{aligned}$$

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

$$\dfrac{\begin{matrix}\vdots\\ !\,\Delta', f : !(B \multimap C), x : B \vdash M' : C\end{matrix} \qquad \begin{matrix}\vdots\\ !\,\Delta', \Gamma, f : !(B \multimap C) \vdash N' : A\end{matrix}}{!\,\Delta', \Gamma \vdash \texttt{letrec}\, f^{B\multimap C} x = M' \,\texttt{in}\, N' : A}\,(\mathrm{rec}) \tag{B.5}$$

Here $\Delta = (\Delta', \Gamma)$. Now by α-converging $!\,\Delta', f : !(B \multimap C), x : B \vdash M' : C$—the top-left judgment in (B.5)—we have $\Vdash !\,\Delta', f : !(B \multimap C), z : B \vdash M'[z/x] : C$. Applying the (rec) rule to the last two judgments, we obtain $\Vdash !\,\Delta', z : B \vdash \texttt{letrec}\, f^{B\multimap C} x = M' \,\texttt{in}\, M'[z/x] : C$. By the ($\multimap.\mathrm{I}_2$) rule this leads to $\Vdash !\,\Delta' \vdash \lambda z^B.\,\texttt{letrec}\, f^{B\multimap C} x = M' \,\texttt{in}\, M'[z/x] : !(B \multimap C)$. The last judgment is combined with the second assumption in (B.5) via the substitution rule (Subst$_2$) in Lemma 3.22, and yields $\Vdash !\,\Delta', \Gamma \vdash N'\bigl[\,\lambda z^B.\,\texttt{letrec}\, f^{B\multimap C} x = M' \,\texttt{in}\, M'[z/x]\,/\,f\,\bigr] : A$. This is our goal.

In the other cases, the reduction $M \longrightarrow_p N$ is derived by one of the rules in Definition 3.7 that deal with quantum constants (such as `new` and U). We do only one case, of the rule ($\texttt{meas}_1$). The other cases are similar.

By inspecting the typing rules it is easy to see that the type A of the term $\mathtt{meas}_i^{n+1}(\mathtt{new}_\rho)$ must be $A \equiv {!}\mathtt{bit} \boxtimes n\text{-}\mathtt{qbit}$. Therefore it suffices to show that the term $\mathtt{tt} \equiv \mathtt{inj}_\ell^\top(*)$ can indeed have the type ${!}\mathtt{bit} \equiv {!}(\top + \top)$. This is shown as follows.

$$\dfrac{\dfrac{}{\Delta \vdash * : {!}\top}\ (\top.\mathrm{I})}{\Delta \vdash \mathtt{inj}_\ell^\top(*) : {!}(\top + \top)}\ (+.\mathrm{I}_1)$$

This concludes the proof. □

Appendix B.7. Proof of Lemma 3.24

Proof. By induction on the construction of the (closed) term M. We only present the case where $M \equiv NL$. If N is not a value, by the induction hypothesis N has a reduction $N \longrightarrow_p N'$; this yields $M \equiv NL \longrightarrow_p N'L$. 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 $M \equiv NL$ is typable, we must have $\Vdash\vdash N : B \multimap A$ for some B. A value N of the type $B \multimap A$ must be either of the following forms: $\lambda x^{B'}.\, N'$, $\mathtt{new}$, $\mathtt{meas}_i^n$, U or $\mathtt{cmp}$.

If $N \equiv \lambda x^{B'}.\, N'$, since L is a value we have $M \equiv NL \longrightarrow_1 N'[L/x]$. If $N \equiv \mathtt{new}$, it is easy to see that a closed value L of type $\mathtt{bit}$ must be either $\mathtt{tt}$ or $\mathtt{ff}$. Therefore the reduction $(\mathtt{new}_1)$ or $(\mathtt{new}_2)$ in Definition 3.7 is enabled from $M \equiv NL$. The other cases are similar.

□

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

The following characterization is standard. See e.g. [38].

Lemma Appendix C.1. *The trace condition (17) holds if and only if: for each $m \in \mathbb{N}$,*

$$\sum_{x\in X}\sum_{n\in\mathbb{N}} M\Big(\big(c(x)\big)_{m,n}\Big) \sqsubseteq \mathcal{I}_m \ . \tag{C.1}$$

Here $\sqsubseteq$ is the Löwner partial order (Definition 2.4); $M\big((c(x))_{m,n}\big)$ is the matrix from Definition 2.8. Note that $M\big((c(x))_{m,n}\big)$ 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

$$A := \mathcal{I}_m - \sum_{x\in X}\sum_{n\in\mathbb{N}} M\Big(\big(c(x)\big)_{m,n}\Big) \ . \tag{C.2}$$

To prove the 'if' part, assume that A is positive. We have, for each $\rho \in \mathrm{DM}_m$,

$$\begin{aligned}
&\mathsf{tr}(A\rho) + \sum_{x\in X}\sum_{n\in\mathbb{N}} \mathsf{tr}\Big((c(x))_{m,n}(\rho)\Big) \\
&= \mathsf{tr}(A\rho) + \sum_{x\in X}\sum_{n\in\mathbb{N}} \mathsf{tr}\Big(\sum_{i\in I_{x,m,n}} E^{(i)}_{x,m,n}\cdot\rho\cdot(E^{(i)}_{x,m,n})^{\dagger}\Big) \\
&\qquad\text{where } \{E^{(i)}_{x,m,n}\}_{i\in I_{x,m,n}} \text{ is an operator-sum representation of } (c(x))_{m,n} \\
&= \mathsf{tr}(A\rho) + \sum_{x\in X}\sum_{n\in\mathbb{N}} \mathsf{tr}\Big(\sum_{i\in I_{x,m,n}} (E^{(i)}_{x,m,n})^{\dagger}\cdot E^{(i)}_{x,m,n}\cdot\rho\Big) \\
&= \mathsf{tr}\Big(\big(A + \sum_{x\in X}\sum_{n\in\mathbb{N}} M\big((c(x))_{m,n}\big)\big)(\rho)\Big) \\
&= \mathsf{tr}(\rho) \le 1 \qquad\qquad \text{by (C.2).}
\end{aligned}$$

Hence it suffices to show that $\mathsf{tr}(A\rho) \ge 0$. It is a standard fact that any density matrix $\rho \in \mathrm{DM}_m$ can be written as

$$\sum_{i\in I} \lambda_i |v_i\rangle\langle v_i| \ ,$$

with $|v_i\rangle \in \mathbb{C}^m$, $\||v_i\rangle\| = 1$, $\lambda_i \ge 0$ and $\sum_i \lambda_i \le 1$. Therefore it suffices to show that $\mathsf{tr}(A|v\rangle\langle v|) \ge 0$ if $\||v\rangle\| = 1$. Now:

$$\mathsf{tr}(A\rho) = \mathsf{tr}(A|v\rangle\langle v|) \overset{(*)}{=} \langle v|A|v\rangle \ge 0 \ ,$$

where $(*)$ is because $\mathsf{tr}(BC) = \mathsf{tr}(CB)$ 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, $\langle v|A|v\rangle \ge 0$.

$$\begin{aligned}
\langle v|A|v\rangle &= \big\langle v\big|\, \mathcal{I}_m - \sum_{x,n}\sum_i (E^{(i)}_{x,m,n})^{\dagger} E^{(i)}_{x,m,n}\,\big|v\big\rangle \\
&\qquad\text{where } \{E^{(i)}_{x,m,n}\}_{i\in I_{x,m,n}} \text{ is an operator-sum representation of } (c(x))_{m,n} \\
&= \langle v|v\rangle - \sum_{x,n}\sum_i \langle v|\,(E^{(i)}_{x,m,n})^{\dagger} E^{(i)}_{x,m,n}\,|v\rangle \\
&= 1 - \sum_{x,n}\sum_i \mathsf{tr}\Big(E^{(i)}_{x,m,n}|v\rangle\langle v|(E^{(i)}_{x,m,n})^{\dagger}\Big) \\
&\qquad\qquad \text{using } \mathsf{tr}(BC) = \mathsf{tr}(CB) \text{ and } \langle v|v\rangle = \|v\|^2 = 1 \\
&= 1 - \sum_{x,n} \mathsf{tr}\Big((c(x))_{m,n}\big(|v\rangle\langle v|\big)\Big) \quad \ge 0 \qquad \text{by (17).}
\end{aligned}$$

This concludes the proof. □

Proposition Appendix C.2. *The construction $\mathcal{Q}$ in Definition 4.1 is indeed a functor.*

Proof. First we check that, given a function $f : X \to Y$ and $c \in \mathcal{Q}X$, the data $(\mathcal{Q}f)(c)$ defined in (18) indeed satisfies the trace condition. This is easy by direct calculations. It remains to be shown that: $\mathcal{Q}(\mathrm{id}) = \mathrm{id}$ and $\mathcal{Q}(g \circ f) = \mathcal{Q}g \circ \mathcal{Q}f$. These are easy consequences of the facts that $\mathrm{id}^{-1} = \mathrm{id}$ and $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$, respectively. □

Lemma Appendix C.3. *The sum in the definition (20) of μ is well-defined.*

Proof. First we show that, for fixed $\gamma \in \mathcal{Q}\mathcal{Q}X$, $m \in \mathbb{N}$ and $\rho \in \mathrm{DM}_m$, there are only countably many pairs $(c, k) \in \mathcal{Q}X \times \mathbb{N}$ such that

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

To see this, observe that the trace condition (17) for $\gamma \in \mathcal{Q}\mathcal{Q}X$ means $\sum_{c,k} \mathsf{tr}\big((\gamma(c))_{m,k}(\rho)\big) \leq 1$. It is a standard fact that a discrete distribution with sum ≤ 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 (20) amounts to

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

The right-hand side is the limit of a sequence (over $l \in \mathbb{N}$) in DM_n that satisfies the assumption of Lemma Appendix A.3. Thus it is well-defined. □

Proposition Appendix C.4. *The construction $\mathcal{Q}$ in Definition 4.1 is indeed a monad.*

Proof. First we verify that the data $\eta_X(x)$ in (19) and $\mu_X(\gamma)$ in (20) satisfy the trace condition (17) and hence belong indeed to $\mathcal{Q}X$. For the unit $\eta_X(x)$ this is obvious. For the multiplication $\mu_X(\gamma)$ we shall verify (17). For any $\rho \in \mathrm{DM}_m$, we have

$$
\begin{aligned}
&\sum_{x\in X}\sum_{n\in\mathbb{N}} \mathsf{tr}\Big(\,\big(\,(\mu_X(\gamma))(x)\,\big)_{m,n}(\rho)\,\Big) \\
&= \sum_{x\in X}\sum_{n\in\mathbb{N}}\sum_{c\in\mathcal{Q}X}\sum_{k\in\mathbb{N}} \mathsf{tr}\Big(\,(c(x))_{k,n}\Big(\,(\gamma(c))_{m,k}(\rho)\,\Big)\Big) \\
&= \sum_{x\in X}\sum_{n\in\mathbb{N}}\sum_{c\in\mathcal{Q}X}\sum_{k\in\mathbb{N}} \mathsf{tr}\Big(\,(\gamma(c))_{m,k}(\rho)\,\Big)\cdot \mathsf{tr}\Big(\,(c(x))_{k,n}\Big(\,\frac{(\gamma(c))_{m,k}(\rho)}{\mathsf{tr}\Big(\,(\gamma(c))_{m,k}(\rho)\,\Big)}\,\Big)\Big) \\
&\hspace{20em} \text{since } (c(x))_{k,n} \text{ and } \mathsf{tr} \text{ are linear} \\
&= \sum_{c\in\mathcal{Q}X}\sum_{k\in\mathbb{N}} \mathsf{tr}\Big(\,(\gamma(c))_{m,k}(\rho)\,\Big)\cdot\Big(\sum_{x\in X}\sum_{n\in\mathbb{N}} \mathsf{tr}\Big(\,(c(x))_{k,n}\Big(\,\frac{(\gamma(c))_{m,k}(\rho)}{\mathsf{tr}\Big(\,(\gamma(c))_{m,k}(\rho)\,\Big)}\,\Big)\Big)\Big) \\
&\leq \sum_{c\in\mathcal{Q}X}\sum_{k\in\mathbb{N}} \mathsf{tr}\Big(\,(\gamma(c))_{m,k}(\rho)\,\Big)\cdot 1 \qquad \text{by the trace condition for } c \in \mathcal{Q}X,\ (*) \\
&\leq 1 \qquad \text{by the trace condition for } \gamma \in \mathcal{Q}\mathcal{Q}X.
\end{aligned}
$$

Note that in the above $(*)$, the matrix

$$\frac{(\gamma(c))_{m,k}(\rho)}{\mathsf{tr}\Big(\,(\gamma(c))_{m,k}(\rho)\,\Big)}$$

has its trace 1 hence is a density matrix.

Next we verify that the maps η_X and μ_X in (19–20) are natural in X. For η_X it is obvious. For μ_X, given $\gamma \in \mathcal{Q}\mathcal{Q}X$ and $f : X \to Y$:

$$\begin{aligned}
&\Big(\,(\mu_Y \circ \mathcal{Q}\mathcal{Q}f)(\gamma)(y)\,\Big)_{m,n} \\
&= \sum_{c' \in \mathcal{Q}Y} \sum_{k \in \mathbb{N}} (c'(y))_{k,n} \circ \big(\,(\mathcal{Q}\mathcal{Q}f)(\gamma)(c')\,\big)_{m,k} \\
&= \sum_{c' \in \mathcal{Q}Y} \sum_{k \in \mathbb{N}} (c'(y))_{k,n} \circ \big(\sum_{c \in (\mathcal{Q}f)^{-1}(\{c'\})} (\gamma(c))_{m,k}\,\big) \\
&= \sum_{c' \in \mathcal{Q}Y} \sum_{c \in (\mathcal{Q}f)^{-1}(\{c'\})} \sum_{k \in \mathbb{N}} (c'(y))_{k,n} \circ (\gamma(c))_{m,k} \qquad \text{since } (c'(y))_{k,n} \text{ is linear} \\
&= \sum_{c \in \mathcal{Q}X} \sum_{k \in \mathbb{N}} \big(\,(\mathcal{Q}f)(c)(y)\,\big)_{k,n} \circ (\gamma(c))_{m,k} \\
&= \sum_{c \in \mathcal{Q}X} \sum_{k \in \mathbb{N}} \big(\sum_{x \in f^{-1}(\{y\})} (c(x))_{k,n}\,\big) \circ (\gamma(c))_{m,k} \\
&= \sum_{x \in f^{-1}(\{y\})} \sum_{c \in \mathcal{Q}X} \sum_{k \in \mathbb{N}} (c(x))_{k,n} \circ (\gamma(c))_{m,k} \\
&= \sum_{x \in f^{-1}(\{y\})} (\mu_X(\gamma)(x))_{m,n} = \Big(\,(\mathcal{Q}f \circ \mu_X)(\gamma)(y)\,\Big)_{m,n} \;.
\end{aligned}$$

This proves the naturality of μ.

Finally we verify that η and μ indeed satisfy the monad laws, that is, that the following diagrams commute.

$$\begin{tikzcd}
\mathcal{Q}X \arrow[r, "\eta_{\mathcal{Q}X}"] \arrow[dr, equal] & \mathcal{Q}\mathcal{Q}X \arrow[d, "\mu_X"] & \mathcal{Q}X \arrow[l, "\mathcal{Q}\eta_X"'] \arrow[dl, equal] \\
& \mathcal{Q}X &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{Q}\mathcal{Q}\mathcal{Q}X \arrow[r, "\mathcal{Q}\mu_X"] \arrow[d, "\mu_{\mathcal{Q}X}"'] & \mathcal{Q}\mathcal{Q}X \arrow[d, "\mu_X"] \\
\mathcal{Q}\mathcal{Q}X \arrow[r, "\mu_X"'] & \mathcal{Q}X
\end{tikzcd}
\tag{C.3}$$

The leftmost triangle is obvious; for the other triangle, we first observe

$$\Big(\,(\mathcal{Q}\eta_X)(c)(c')\,\Big)_{m,n} \;=\; \begin{cases} (c(x'))_{m,n} & \text{if } c' = \eta_X(x') \text{ for some } x' \in X, \\ 0 & \text{otherwise.} \end{cases} \tag{C.4}$$

This is used in the following calculation.

$$
\begin{aligned}
\Big(\big(\mu_X \circ (\mathcal{Q}\eta_X) \big)(c)(x) \Big)_{m,n} &= \sum_{c' \in \mathcal{Q}X} \sum_{k \in \mathbb{N}} (c'(x))_{k,n} \circ \Big((\mathcal{Q}\eta_X)(c)(c') \Big)_{m,k} \\
&= \sum_{x' \in X} \sum_{k \in \mathbb{N}} \big((\eta_X(x'))(x) \big)_{k,n} \circ (c(x'))_{m,k} \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{by (C.4)} \\
&= \mathcal{I}_n \circ (c(x))_{m,n} = (c(x))_{m,n} \enspace .
\end{aligned}
$$

This proves the commutativity of the triangle in the middle of (C.3). For the square on the right, given $\Gamma \in \mathcal{Q}\mathcal{Q}\mathcal{Q}X$:

$$
\begin{aligned}
\Big((\mu_X \circ \mathcal{Q}\mu_X)(\Gamma)(x) \Big)_{m,n} &= \sum_{c \in \mathcal{Q}X} \sum_{k \in \mathbb{N}} (c(x))_{k,n} \circ \Big((\mathcal{Q}\mu_X)(\Gamma)(c) \Big)_{m,k} \\
&= \sum_{c \in \mathcal{Q}X} \sum_{k \in \mathbb{N}} (c(x))_{k,n} \circ \Big(\sum_{\gamma \in \mu_X^{-1}(\{c\})} (\Gamma(\gamma))_{m,k} \Big) \\
&= \sum_{c \in \mathcal{Q}X} \sum_{k \in \mathbb{N}} \sum_{\gamma \in \mu_X^{-1}(\{c\})} (c(x))_{k,n} \circ (\Gamma(\gamma))_{m,k} \\
&= \sum_{\gamma \in \mathcal{Q}\mathcal{Q}X} \sum_{k \in \mathbb{N}} (\mu_X(\gamma)(x))_{k,n} \circ (\Gamma(\gamma))_{m,k} \\
&= \sum_{\gamma \in \mathcal{Q}\mathcal{Q}X} \sum_{k \in \mathbb{N}} \sum_{c \in \mathcal{Q}X} \sum_{l \in \mathbb{N}} (c(x))_{l,n} \circ (\gamma(c))_{k,l} \circ (\Gamma(\gamma))_{m,k} \enspace ; \\
\Big((\mu_X \circ \mu_{\mathcal{Q}X})(\Gamma)(x) \Big)_{m,n} &= \sum_{c \in \mathcal{Q}X} \sum_{l \in \mathbb{N}} (c(x))_{l,n} \circ (\mu_{\mathcal{Q}X}(\Gamma)(c))_{m,l} \\
&= \sum_{c \in \mathcal{Q}X} \sum_{l \in \mathbb{N}} (c(x))_{l,n} \circ \big(\sum_{\gamma \in \mathcal{Q}\mathcal{Q}X} \sum_{k \in \mathbb{N}} (\gamma(c))_{k,l} \circ (\Gamma(\gamma))_{m,k} \big) \\
&= \sum_{\gamma \in \mathcal{Q}\mathcal{Q}X} \sum_{k \in \mathbb{N}} \sum_{c \in \mathcal{Q}X} \sum_{l \in \mathbb{N}} (c(x))_{l,n} \circ (\gamma(c))_{k,l} \circ (\Gamma(\gamma))_{m,k} \enspace .
\end{aligned}
$$

This concludes the proof. □

Appendix C.1. The Kleisli Category $\mathcal{K}\ell(\mathcal{Q})$

Lemma (Lemma 4.2, repeated). Given two successive arrows $f : X \nrightarrow Y$ and $g : Y \nrightarrow U$ in $\mathcal{K}\ell(\mathcal{Q})$, their composition $g \odot f : X \nrightarrow U$ is concretely given as follows.

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

PROOF. Given $x \in X$, $u \in U$ and $\rho \in \mathrm{DM}_m$:

$$\begin{aligned}
\Big((g \odot f)(x)(u) \Big)_{m,n} &= \Big((\mu_U \circ \mathcal{Q}g \circ f)(x)(u) \Big)_{m,n} \\
&= \Big(\mu_U \big((\mathcal{Q}g)(f(x)) \big)(u) \Big)_{m,n} \\
&= \sum_{c \in \mathcal{Q}U} \sum_{k \in \mathbb{N}} (c(u))_{k,n} \circ \Big(\big((\mathcal{Q}g)(f(x)) \big)(c) \Big)_{m,k} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{by def. of } \mu \\
&= \sum_{c \in \mathcal{Q}U} \sum_{k \in \mathbb{N}} (c(u))_{k,n} \circ \Big(\sum_{y \in g^{-1}(\{c\})} (f(x)(y))_{m,k} \Big) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{by def. of } \mathcal{Q}g \\
&= \sum_{c \in \mathcal{Q}U} \sum_{k \in \mathbb{N}} \sum_{y \in g^{-1}(\{c\})} (c(u))_{k,n} \circ (f(x)(y))_{m,k} \\
&= \sum_{y \in Y} \sum_{k \in \mathbb{N}} (g(y)(u))_{k,n} \circ (f(x)(y))_{m,k} \ . \qquad (*)
\end{aligned}$$

Here the equality $(*)$ holds because, due to $g : Y \to \mathcal{Q}U$ being a function, we have $Y = \coprod_{c \in \mathcal{Q}U} \{y \mid g(y) = c\}$. □

Note that $\mathcal{K}\ell(\mathcal{Q})$ has finite coproducts, carried over from **Sets** by the Kleisli inclusion functor.

Theorem Appendix C.5. *The monad $\mathcal{Q}$ on* **Sets** *satisfies the following conditions (from [33, Requirements 4.7]); and therefore by [33, Proposition 4.8], the category $\mathcal{K}\ell(\mathcal{Q})$ is partially additive.*

1. *$\mathcal{K}\ell(\mathcal{Q})$ is ω-* **CPO** *enriched.*
2. *$\mathcal{K}\ell(\mathcal{Q})$ has monotone cotupling.*
3. *For each $X, Y \in \mathcal{K}\ell(\mathcal{Q})$, the least element $\perp_{X,Y} \in \mathcal{K}\ell(Q)(X, Y)$ in the homset is preserved by both pre- and post-composition: that is, $f \odot \perp = \perp$ and $\perp \odot g = \perp$.*
   We note that, under this condition, there exist "projection" maps $p_j : \coprod_{i \in I} X_i \nrightarrow X_j$ such that
   $$p_j \odot \kappa_i \ = \begin{cases} \mathrm{id} & \textit{if } i = j, \\ \perp & \textit{otherwise,} \end{cases}$$
   where $\kappa_j : X_j \nrightarrow \coprod_{i \in I} X_i$ denotes a coprojection.
4. *The "bicartesian" maps*
   $$\mathrm{bc}_{(X_i)_{i \in I}} \ := \ \left(\mathcal{Q}(\coprod_{i \in I} X_i) \xrightarrow{\langle p_i^\flat \rangle_{i \in I}} \prod_{i \in I} \mathcal{Q}X_i \right) \quad \textit{where} \quad p_i^\flat := \mu \circ T p_i$$

form a cartesian natural transformation with monic components. This means that all the naturality squares

$$\begin{tikzcd}
T(\coprod_i X_i) \arrow[r, tail, "\mathrm{bc}"] \arrow[d, "T(\coprod_i f_i)"'] & \prod_i TX_i \arrow[d, "\prod_i Tf_i"] \\
T(\coprod_i Y_i) \arrow[r, tail, "\mathrm{bc}"'] & \prod_i TY_i
\end{tikzcd}$$

are pullback diagrams in **Sets***, for each* $f_i : X_i \to Y_i$ *in* **Sets**.

The original condition [33, Requirements 4.7] is stated in terms of DCPOs instead of ω-CPOs. This difference is not important.

Proof. We use the pointwise extension of the order $\sqsubseteq$ in Definition 4.4 in homsets $\mathcal{K}\ell(\mathcal{Q})(X, Y)$. It is an ω-CPO due to Proposition 2.13. It is easy to see that the bottoms are preserved by pre- and post-composition. To see that supremums 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 Lemma Appendix C.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 $\mathrm{bc}(c) = \mathrm{bc}(d)$. Then $p_i^\flat(c) = p_i^\flat(d)$ for each $i \in I$. It is easy to see that $p_i^\flat(c) = c \circ \kappa_i$, therefore

$$c = [c \circ \kappa_i]_i = [d \circ \kappa_i]_i = d \ .$$

It is straightforward to see that the naturality squares are pullbacks. □

Appendix D. Proofs for §5

Appendix D.1. Proof of Lemma 5.8

Proof. The set $A_\mathcal{Q} = \mathcal{K}\ell(\mathcal{Q})(\mathbb{N}, \mathbb{N})$ is an ω-CPO due to the ω-**CPO** enriched structure of the category $\mathcal{K}\ell(\mathcal{Q})$ (see Theorem 4.5). Therefore the order $\sqsubseteq$ on $A_\mathcal{Q}$ is essentially the Löwner partial order (Definition 2.4).

To show the item 1, we use the fact that composition $\odot$ of arrows and the trace operator $\mathtt{tr}$ are both continuous in the Kleisli category $\mathcal{K}\ell(\mathcal{Q})$. Indeed, the former is part of the fact that $\mathcal{K}\ell(\mathcal{Q})$ is ω-**CPO** enriched (Theorem Appendix C.5). The proof for the latter is not hard either, exploiting the explicit presentation of $\mathtt{tr}$ by Girard's execution formula (see [59, 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 (29) and (30).

The item 2 is proved using the presentation of $\mathtt{tr}$ by Girard's execution formula and the fact that composition $\odot$ in $\mathcal{K}\ell(\mathcal{Q})$ is (left and right) strict. □

Appendix D.2. Proof of Lemma 5.13

PROOF. For inductiveness of $U \dot{\times} V$, assume $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$, $x'_0 \sqsubseteq x'_1 \sqsubseteq \cdots$ and $(x_i, x'_i) \in U \dot{\times} V$ for each i. By the definition of $U \dot{\times} V$, we find $k_i, l_i, u_i, k'_i, l'_i, u'_i$ such that $x_i = \dot{\mathsf{P}}k_i(\dot{\mathsf{P}}l_i u_i)$, $x'_i = \dot{\mathsf{P}}k'_i(\dot{\mathsf{P}}l'_i u'_i)$, $(k_i u_i, k'_i u'_i) \in U$ and $(l_i u_i, l'_i u'_i) \in V$. Since $k_i = \dot{\mathsf{P}}_{\mathsf{l}} x_i$, by continuity of $\cdot$ we have that $(k_i)_i$ is an increasing chain. So are $(l_i)_i, (u_i)_i, (k'_i)_i, (l'_i)_i, (u'_i)_i$; therefore $(k_i u_i)_i, (l_i u_i)_i, (k'_i u'_i)_i, (l'_i u'_i)_i$ are increasing, too. By the admissibility of U and V we have

$$\bigl(\sup_i k_i u_i\,,\, \sup_i k'_i u'_i\bigr) \in U \quad \text{and} \quad \bigl(\sup_i l_i u_i\,,\, \sup_i l'_i u'_i\bigr) \in V \enspace .$$

Again by continuity of $\cdot$ we have

$$\sup_i x_i = \dot{\mathsf{P}}(\sup_i k_i)\bigl(\dot{\mathsf{P}}(\sup_i l_i)(\sup_i u_i)\bigr) \quad \text{and} \quad \sup_i x'_i = \dot{\mathsf{P}}(\sup_i k'_i)\bigl(\dot{\mathsf{P}}(\sup_i l'_i)(\sup_i u'_i)\bigr) \enspace ;$$

since $\sup_i k_i u_i = (\sup_i k_i)(\sup_i u_i)$ (and so on) we conclude that $(\sup_i x_i, \sup_i x'_i) \in U \dot{\times} V$.

Strictness is the reason we use $\dot{\times}$ instead of $\times$. We have $\dot{\mathsf{P}}\bot(\dot{\mathsf{P}}\bot\bot) = \bot$: this is because $\dot{\mathsf{P}}xy = j \odot (x + y) \odot k$ (see (40)) and that $\odot$ is (left and right) strict. This shows $(\bot, \bot) \in U \dot{\times} V$.

Inductiveness of $X \multimap U$ is easily shown by similar arguments. Finally, strictness of $X \multimap U$ is because for each $(x, x') \in X$, $(\bot x, \bot x') = (\bot, \bot) \in U$. Here the left strictness of $\cdot$ is crucial. $\square$

Appendix D.3. Proof of Lemma 5.27

PROOF. The PER $[\![0\text{-}\mathtt{qbit}]\!]$ (Definition 5.18) is admissible. Indeed, $(\bot, \bot) = (\mathsf{Q}_0, \mathsf{Q}_0) \in [\![0\text{-}\mathtt{qbit}]\!]$ and an increasing chain in $[\![0\text{-}\mathtt{qbit}]\!]$ is precisely an increasing chain in $[0, 1]$. Therefore Lemma 5.13 shows that the functor $F_{\mathrm{pbt}} = [\![\mathtt{bit}]\!] \multimap ([\![0\text{-}\mathtt{qbit}]\!] \dot{\times} \_)$ preserves admissibility. Since $\mathsf{Bt} = \{(\bot, \bot)\}$ is admissible, each object $F^i_{\mathrm{pbt}} B$ in the final sequence is admissible.

We prove strictness of R. By Definition 5.4 we have $\bot = (\bot)_{i \in \mathbb{N}}$; hence $\dot{\mathsf{P}}(\bot)_i \bot = \bot$ by (40). Thus it suffices to show that for any j and any i such that $j \le i$, $(c_{i,j}\bot, \bot) \in F^j_{\mathrm{pbt}}\mathsf{Bt}$. This is by cases: we distinguish $j = 0$ and $j > 0$.

If $j = 0$, $[c_{i,j}] : F^i_{\mathrm{pbt}}\mathsf{Bt} \to \mathsf{Bt}$ is the unique map to $\mathsf{Bt} = \{(\bot, \bot)\}$ and hence $c_{i,j}\bot = \bot$. Therefore $(c_{i,j}\bot, \bot) \in F^j_{\mathrm{pbt}}\mathsf{Bt}$ for any i.

Assume $j > 0$. Let us first note the functor $[\![\mathtt{bit}]\!] \multimap \_$'s action on arrows: it carries

$$[c] : X \longrightarrow Y \quad \text{to} \quad [\lambda tb.c(tb)] : [\![\mathtt{bit}]\!] \multimap X \longrightarrow [\![\mathtt{bit}]\!] \multimap Y \enspace . \tag{D.1}$$

The functor $[\![0\text{-}\mathtt{qbit}]\!] \times \_$ carries $[c] : X \to Y$ to

$$\bigl[\lambda v.v(\lambda k_1 w.w(\lambda k_2 u.\mathsf{P}k_1(\mathsf{P}(\lambda z.c(k_2 z))u)))\bigr] : [\![0\text{-}\mathtt{qbit}]\!] \times X \longrightarrow [\![0\text{-}\mathtt{qbit}]\!] \times Y \enspace ; \tag{D.2}$$

after suitable insertion of the conversion combinators $\mathsf{C}_{\mathsf{P} \mapsto \dot{\mathsf{P}}}$ and $\mathsf{C}_{\dot{\mathsf{P}} \mapsto \mathsf{P}}$, it describes the functor $[\![0\text{-}\mathtt{qbit}]\!] \dot{\times} \_$'s action on arrows.

Our aim now is to show

$$(c_{i,j}\bot, \bot) \in F^j_{\mathrm{pbt}}\mathsf{Bt} = [\![\mathtt{bit}]\!] \multimap ([\![0\text{-}\mathtt{qbit}]\!] \dot{\times} F^{j-1}_{\mathrm{pbt}}\mathsf{Bt}) ;$$

by (32) it suffices to show

$$\big(c_{i,j}\bot b,\ \bot b'\big) \in [\![0\text{-}\mathtt{qbit}]\!] \dot{\times} F^{j-1}_{\mathrm{pbt}}\mathsf{Bt} \quad \text{for each } (b,b') \in [\![\mathtt{bit}]\!]. \tag{D.3}$$

By (D.1) we have

$$\big(c_{i,j}\ ,\ \lambda tb.d_{i-1,j-1}(tb)\big) \in F^i_{\mathrm{pbt}}\mathsf{Bt} \multimap F^j_{\mathrm{pbt}}\mathsf{Bt}\ ,$$

where $d_{i-1,j-1}$ is the realizer of the arrow

$$[\![0\text{-}\mathtt{qbit}]\!] \dot{\times} [c_{i-1,j-1}] : [\![0\text{-}\mathtt{qbit}]\!] \dot{\times} F^{i-1}_{\mathrm{pbt}}\mathsf{Bt} \longrightarrow [\![0\text{-}\mathtt{qbit}]\!] \dot{\times} F^{j-1}_{\mathrm{pbt}}\mathsf{Bt}$$

described as in (D.2). Therefore

$$\big(c_{i,j}\bot b\ ,\ d_{i-1,j-1}(\bot b)\big) \in [\![0\text{-}\mathtt{qbit}]\!] \dot{\times} F^{j-1}_{\mathrm{pbt}}\mathsf{Bt}\ . \tag{D.4}$$

Now using that $\cdot$ is left strict,

$$d_{i-1,j-1}(\bot b) = d_{i-1,j-1}\bot = \bot = \bot b'\ ,$$

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).

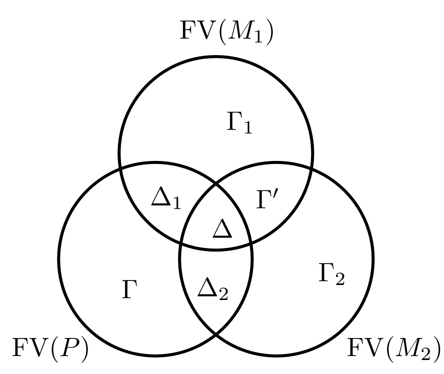
Inductiveness of R is proved much like the proof of Lemma 5.13. □

Appendix E. Well-Definedness of Interpretation of Well-Typed Terms

Towards our goal of proving Lemma 5.35, 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 $(+.\mathrm{E})_{\mathrm{P}}$ rule are complicated: they form the following partition of the free variables of the term in the conclusion. Notice that Γ' need not be of the form $!\,\Gamma''$.



$$\frac{}{x : A \vdash x : A}\ (\mathrm{Ax}.1)_{\mathrm{P}} \qquad \frac{}{\vdash c : !\,\mathrm{DType}(c)}\ (\mathrm{Ax}.2)_{\mathrm{P}}$$

$$\frac{[x : A], \Delta \vdash M : B \qquad \Delta \not\equiv !\,\Delta' \text{ for any } \Delta'}{\Delta \vdash \lambda x^{A}.M : A \multimap B}\ (\multimap.\mathrm{I}_1)_{\mathrm{P}}$$

$$\frac{[x : A], \Delta \vdash M : B \qquad \Delta \equiv !\,\Delta' \text{ for some } \Delta'}{\Delta \vdash \lambda x^{A}.M : !(A \multimap B)}\ (\multimap.\mathrm{I}_2)_{\mathrm{P}}$$

$$\frac{!\,\Delta, \Gamma_1 \vdash M : !^{n}(A \multimap B) \qquad !\,\Delta, \Gamma_2 \vdash N : C \qquad C <: A}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash MN : B}\ (\multimap.\mathrm{E})_{\mathrm{P}}$$

$$\frac{!\,\Delta, \Gamma_1 \vdash M_1 : !^{n} A_1 \quad !\,\Delta, \Gamma_2 \vdash M_2 : !^{n} A_2 \quad \begin{array}{l}(m = 0 \Leftrightarrow n = 0)\\ \wedge(m = 1 \Leftrightarrow n \geq 1)\end{array} \quad \begin{array}{l}\text{At least one of } A_1 \text{ and } A_2 \text{ is}\\ \text{not of the form } !\,B\end{array}}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash \langle M_1, M_2\rangle : !^{m}(!^{m} A_1 \boxtimes !^{m} A_2)}\ (\boxtimes.\mathrm{I})_{\mathrm{P}}$$

$$\frac{\begin{array}{ll} !\,\Delta, \Gamma_1 \vdash M : !^{m}(C_1 \boxtimes C_2) & m = 0 \Rightarrow n = 0 \\ !\,\Delta, \Gamma_2, [x_1 : !^{n} A_1], [x_2 : !^{n} A_2] \vdash N : A & C_1 <: A_1 \quad C_2 <: A_2 \end{array}}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash \mathtt{let}\, \langle x_1^{!^{n} A_1}, x_2^{!^{n} A_2}\rangle = M \,\mathtt{in}\, N : A}\ (\boxtimes.\mathrm{E})_{\mathrm{P}}$$

$$\frac{}{\Delta \vdash * : !\top}\ (\top.\mathrm{I})_{\mathrm{P}} \qquad \frac{!\,\Delta, \Gamma_1 \vdash M : !^{n} \top \qquad !\,\Delta, \Gamma_2 \vdash N : A}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash \mathtt{let}\, * = M \,\mathtt{in}\, N : A}\ (\top.\mathrm{E})_{\mathrm{P}}$$

$$\frac{\Delta \vdash M : !^{n} A_1 \quad \begin{array}{l}(m = 0 \Leftrightarrow n = 0)\\ \wedge(m = 1 \Leftrightarrow n \geq 1)\end{array} \quad \begin{array}{l}A_1 \text{ is}\\ \text{not of the form } !\,B\end{array}}{\Delta \vdash \mathtt{inj}_{\ell}^{A_2} M : !^{m}(!^{m} A_1 + A_2)}\ (+.\mathrm{I}_1)_{\mathrm{P}}$$

$$\frac{\Delta \vdash N : !^{n} A_2 \quad \begin{array}{l}(m = 0 \Leftrightarrow n = 0)\\ \wedge(m = 1 \Leftrightarrow n \geq 1)\end{array} \quad \begin{array}{l}A_2 \text{ is}\\ \text{not of the form } !\,B\end{array}}{\Delta \vdash \mathtt{inj}_{r}^{A_1} N : !^{m}(A_1 + !^{m} A_2)}\ (+.\mathrm{I}_2)_{\mathrm{P}}$$

$$\frac{\begin{array}{ll} !\,\Delta, !\,\Delta_1, !\,\Delta_2, \Gamma \vdash P : !^{m}(C_1 + C_2) & m = 0 \Rightarrow n = 0 \\ !\,\Delta, !\,\Delta_1, \Gamma', \Gamma_1', [x_1 : !^{n} A_1] \vdash M_1 : B & C_1 <: A_1 \quad C_2 <: A_2 \\ !\,\Delta, !\,\Delta_2, \Gamma', \Gamma_2', [x_2 : !^{n} A_2] \vdash M_2 : B & x_1 \notin |\Gamma, \Delta_2, \Gamma_2|, x_2 \notin |\Gamma, \Delta_1, \Gamma_1| \end{array}}{!\,\Delta, !\,\Delta_1, !\,\Delta_2, \Gamma, \Gamma', \Gamma_1', \Gamma_2' \vdash \mathtt{match}\, P \,\mathtt{with}\, (x_1^{!^{n} A_1} \mapsto M_1 \mid x_2^{!^{n} A_2} \mapsto M_2) : B}\ (+.\mathrm{E})_{\mathrm{P}}$$

$$\frac{\begin{array}{ll} !\,\Delta, [f : !(A \multimap B)], [x : A] \vdash M : B'' & \\ !\,\Delta, \Gamma, [f : !(A \multimap B)] \vdash N : C & B'' <: B \end{array}}{!\,\Delta, \Gamma \vdash \mathtt{letrec}\, f^{A \multimap B} x = M \,\mathtt{in}\, N : C}\ (\mathrm{rec})_{\mathrm{P}}$$

Table E.2: Principal typing rules for Hoq

We shall write $\Pi \Vdash_{\mathrm{P}} \Delta \vdash M : A$ if a derivation tree Π, according to these rules, derives the type judgment. We write $\Vdash_{\mathrm{P}} \Delta \vdash M : A$ if there exists such Π, that is, the type judgment is derivable.

Lemma Appendix E.2. $\Vdash_{\mathrm{P}} \Delta \vdash M : A$ *implies* $\Vdash \Delta \vdash M : A$.

Proof. By induction on the principal type derivation of $\Vdash_{\mathrm{P}} \Delta \vdash M : A$. We only present some cases.

When the last rule applied is $(\multimap.\mathrm{E})_{\mathrm{P}}$, that is

$$\frac{!\,\Delta', \Gamma_1 \vdash M' : !^n(A' \multimap A) \qquad !\,\Delta', \Gamma_2 \vdash N' : C \qquad C <: A'}{!\,\Delta', \Gamma_1, \Gamma_2 \vdash M'N' : A}\ (\multimap.\mathrm{E})_{\mathrm{P}}$$

with $\Delta = (!\,\Delta', \Gamma_1, \Gamma_2)$ and $M \equiv M'N'$, by the induction hypothesis we have

$$\Vdash\ !\,\Delta', \Gamma_1 \vdash M' : !^n(A' \multimap A) \quad \text{and} \quad \Vdash\ !\,\Delta', \Gamma_2 \vdash N' : C\ .$$

Applying Corollary 3.18 to the former yields $\Vdash\ !\,\Delta', \Gamma_1 \vdash M' : A' \multimap A$. Then, together with $C <: A'$, we can use the $(\multimap.\mathrm{E})$ rule (of the original type system) to derive $!\,\Delta', \Gamma_1, \Gamma_2 \vdash M'N' : A$. The case $(\boxtimes.\mathrm{I})_{\mathrm{P}}$ is similar using Corollary 3.18. For the case $(\boxtimes.\mathrm{E})_{\mathrm{P}}$ we additionally use Lemma 3.17 to show that

$$\Vdash\ !\,\Delta, \Gamma_2, x_1 : !^n A_1, x_2 : !^n A_2 \vdash N : A \quad \text{implies} \quad \Vdash\ !\,\Delta, \Gamma_2, x_1 : !^n C_1, x_2 : !^n C_2 \vdash N : A\ .$$

The case $(+.\mathrm{E})_{\mathrm{P}}$ and $(\mathrm{rec})_{\mathrm{P}}$ are similar. $\square$

Lemma Appendix E.3. 1. $\Vdash_{\mathrm{P}} \Delta \vdash M : A$ *implies* $|\Delta| = \mathrm{FV}(M)$.

2. *Principal typing is unique in the following sense:*

   $\Vdash_{\mathrm{P}} \Delta \vdash M : A$ *and* $\Vdash_{\mathrm{P}} \Delta \vdash M : A'$ *imply* $A \equiv A'$.

3. *Derivation in principal typing is unique: if* $\Pi \Vdash_{\mathrm{P}} \Delta \vdash M : A$ *and* $\Pi' \Vdash_{\mathrm{P}} \Delta \vdash M : A$*, 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 $\mathtt{let}\,\langle x_1^{!^n A_1}, x_2^{!^n A_2}\rangle = M'\,\mathtt{in}\,N$. Then its principal type derivation must end with the $(\boxtimes.\mathrm{E})_{\mathrm{P}}$ rule, as below.

$$\frac{\begin{array}{ll} !\,\Delta, \Gamma_1 \vdash M' : !^m(C_1 \boxtimes C_2) & m = 0 \Rightarrow n = 0 \\ !\,\Delta, \Gamma_2, [x_1 : !^n A_1], [x_2 : !^n A_2] \vdash N : A & C_1 <: A_1 \quad C_2 <: A_2 \end{array}}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash \mathtt{let}\,\langle x_1^{!^n A_1}, x_2^{!^n A_2}\rangle = M'\,\mathtt{in}\,N : A}\ (\boxtimes.\mathrm{E})_{\mathrm{P}} \tag{E.1}$$

The context $!\,\Delta, \Gamma_2, [x_1 : !^{n_1} A_1], [x_2 : !^{n_2} A_2]$ 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 $(\multimap.\mathrm{E})_{\mathrm{P}}$ is involved) the items 1–2. play an essential role. $\square$

Definition Appendix E.4 (Interpretation of principal type judgments). For each derivation $\Pi \Vdash_{\mathrm{P}} \Delta \vdash M : A$ by the rules in Table E.2, we assign an arrow

$$[\![\Pi]\!]^{\mathrm{P}} \;:\; [\![\Delta]\!] \longrightarrow T[\![A]\!]$$

in the way that is a straightforward adaptation of Definition 5.34.

Lemma Appendix E.5. *Assume $\Pi \Vdash \Delta \vdash M : A$ (in the original rules in Table 1). Then there exist a type A° and a derivation Π° (in the principal typing rules in Table E.2) such that:*

1. $\Pi^\circ \Vdash_{\mathrm{P}} \Delta|_{\mathrm{FV}(M)} \vdash M : A^\circ$,
2. $A^\circ <: A$, *and*
3. *the following diagram commutes.*

$$\begin{tikzcd}
[\![\Delta]\!] \arrow[r, "\mathsf{weak}"] \arrow[drr, bend right=10, "{[\![\Pi]\!]}"'] & {[\![\Delta|_{\mathrm{FV}(M)}]\!]} \arrow[r, "{[\![\Pi^\circ]\!]^{\mathrm{P}}}"] & T[\![A^\circ]\!] \arrow[d, "{T[\![A^\circ <: A]\!]}"] \\
& & T[\![A]\!]
\end{tikzcd} \tag{E.2}$$

Proof. The diagram in (E.2) can be refined into the following one; we shall prove that the triangle therein commutes.

$$\begin{tikzcd}
[\![\Delta]\!] \arrow[r, "\mathsf{weak}"] & {[\![\Delta|_{\mathrm{FV}(M)}]\!]} \arrow[r, "{[\![\Pi^\circ]\!]^{\mathrm{P}}}"] \arrow[dr, "{[\![\Pi]\!]_{\mathrm{FV}}}"'] & T[\![A^\circ]\!] \arrow[d, "{T[\![A^\circ <: A]\!]}"] \\
& & T[\![A]\!]
\end{tikzcd} \tag{E.3}$$

Here $[\![\Pi]\!]_{\mathrm{FV}}$ is as in Definition 5.34. The proof is by induction on Π. We present one case; the others are similar.

Assume Π is in the following form, with the last rule applied being ($\multimap$.E).

$$\Pi \;\equiv\; \left[\dfrac{\begin{matrix}\vdots\, \Pi_1 \\ !\,\Delta, \Gamma_1 \vdash M : A \multimap B\end{matrix} \qquad \begin{matrix}\vdots\, \Pi_2 \\ !\,\Delta, \Gamma_2 \vdash N : C\end{matrix} \qquad C <: A}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash MN : B}\;(\multimap.\mathrm{E})\right]$$

By the induction hypothesis, there exist types D, E and derivations Π_1°, Π_2° such that

$\Pi_1^\circ \Vdash_{\mathrm{P}} !\,\Delta, \Gamma_1 \vdash M : D$,
$D <: A \multimap B$,
$[\![\Pi_1]\!]_{\mathrm{FV}} = T[\![D <: A \multimap B]\!] \circ [\![\Pi_1^\circ]\!]^{\mathrm{P}}$, and

$\Pi_2^\circ \Vdash_{\mathrm{P}} !\,\Delta, \Gamma_2 \vdash N : E$;
$E <: C$;
$[\![\Pi_2]\!]_{\mathrm{FV}} = T[\![E <: C]\!] \circ [\![\Pi_2^\circ]\!]^{\mathrm{P}}$.

Since $D <: A \multimap B$, the type D must be of the form $D \equiv !^m(A' \multimap B')$ with $A <: A'$ and $B' <: B$. Now consider the following derivation Π°.

$$\Pi^\circ \;:\equiv\; \left[\dfrac{\begin{matrix}\vdots\, \Pi_1^\circ \\ !\,\Delta, \Gamma_1 \vdash M : !^m(A' \multimap B')\end{matrix} \qquad \begin{matrix}\vdots\, \Pi_2^\circ \\ !\,\Delta, \Gamma_2 \vdash N : E\end{matrix} \qquad E <: A'}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash MN : B'}\;(\multimap.\mathrm{E})_{\mathrm{P}}\right]$$

Here the side condition $E <: A'$ holds since $E <: C <: A <: A'$. Thus we obtain $\Pi^\circ \Vdash_{\mathrm{P}} !\,\Delta, \Gamma_1, \Gamma_2 \vdash MN : B'$ with $B' <: B$.

It remains to show that $[\![\Pi]\!]_{\mathrm{FV}} = (T[\![B' <: B]\!]) \circ [\![\Pi^\circ]\!]^{\mathrm{P}}$. This is, however, an immediate consequence of

- the induction hypothesis,
- $[\![A <: C]\!] = [\![B <: C]\!] \circ [\![A <: B]\!]$ (Lemma 5.31),
- bifunctoriality of $\boxtimes$ and $\multimap$, and
- naturality of $\mathrm{str}, \mathrm{str}', \mathsf{ev}$ and μ.

This can be checked by straightforward diagram chasing. □

We are ready to prove Lemma 5.35.

Proof. (Of Lemma 5.35) Assume $\Pi \Vdash \Delta \vdash M : A$ and $\Pi' \Vdash \Delta \vdash M : A$. We apply Lemma Appendix E.5 to obtain $A^\circ, \Pi^\circ, (A')^\circ, (\Pi')^\circ$ such that

$$\begin{array}{lcl} \Pi^\circ \Vdash_{\mathrm{P}} \Delta|_{\mathrm{FV}(M)} \vdash M : A^\circ \ , & & (\Pi')^\circ \Vdash_{\mathrm{P}} \Delta|_{\mathrm{FV}(M)} \vdash M : (A')^\circ \ , \\ [\![\Pi]\!] = T[\![A^\circ <: A]\!] \circ [\![\Pi^\circ]\!]^{\mathrm{P}} \circ \mathsf{weak} & \text{and} & [\![\Pi']\!] = T[\![(A')^\circ <: A]\!] \circ [\![(\Pi')^\circ]\!]^{\mathrm{P}} \circ \mathsf{weak} \ . \end{array}$$

Applying Lemma Appendix E.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 $[\![\Pi]\!] = [\![\Pi']\!]$. □

Appendix F. Proofs for §6

Lemma Appendix F.1. *Let M be a term such that $\Gamma \vdash M : A$ is derivable. Assume a subtype relation $A <: B$. Then we have*

$$T[\![A <: B]\!] \circ [\![\Gamma \vdash M : A]\!] \quad = \quad [\![\Gamma \vdash M : B]\!] \ .$$

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

$$[\![\Gamma]\!] \xrightarrow{\mathsf{weak}} [\![A']\!] \xrightarrow{[\![A' <: A]\!]} [\![A]\!] \xrightarrow{[\![A <: B]\!]} [\![B]\!] \xrightarrow{\eta^T_{[\![B]\!]}} T[\![B]\!]. \tag{F.1}$$

Since $[\![A <: B]\!] \circ [\![A' <: A]\!]$ is equal to $[\![A' <: B]\!]$ by Lemma 5.31, the arrow (F.1) is equal to $[\![\Gamma \vdash x : B]\!]$.

When $M \equiv M_1\, M_2$, the term environment Γ is of the form $!\,\Delta, \Gamma_1, \Gamma_2$, and there is a type C such that $!\,\Delta, \Gamma_1 \vdash M_1 : C \multimap A$ and $!\,\Delta, \Gamma_2 \vdash M_2 : C$ are derivable. The interpretation $[\![\Gamma \vdash M_1\, M_2 : A]\!]$ is

$$\begin{aligned} &\mu_{[\![A]\!]} \circ \mu_{T[\![A]\!]} \circ TT\mathsf{ev}_{[\![C]\!], T[\![A]\!]} \circ T\mathrm{str}_{[\![C]\!] \multimap T[\![A]\!], [\![C]\!]} \circ \mathrm{str}'_{[\![C]\!] \multimap T[\![A]\!], T[\![C]\!]} \\ &\qquad\qquad \circ \big([\![!\,\Delta, \Gamma_1 \vdash M_1 : C \multimap A]\!] \boxtimes [\![!\,\Delta, \Gamma_2 \vdash M_2 : C]\!] \big) \circ c \ , \end{aligned}$$

where $c : [\![!\,\Delta, \Gamma_1, \Gamma_2]\!] \to [\![!\,\Delta, \Gamma_1]\!] \boxtimes [\![!\,\Delta, \Gamma_2]\!]$ is a suitable permutation followed by contractions. By naturality of μ, str, str$'$ and ev, the composition $T[\![A <: B]\!] \circ [\![\Gamma \vdash M_1\, M_2 : A]\!]$ is equal to

$$\mu_{[\![B]\!]} \circ \mu_{T[\![B]\!]} \circ TT\mathsf{ev}_{[\![C]\!],T[\![B]\!]} \circ T\mathrm{str}_{[\![C]\!] \multimap T[\![B]\!],[\![C]\!]} \circ \mathrm{str}'_{[\![C]\!] \multimap T[\![B]\!],T[\![C]\!]}$$
$$\circ \Big(\big(T([\![C]\!] \multimap T[\![A <: B]\!]) \circ [\![!\,\Delta, \Gamma_1 \vdash M_1 : C \multimap A]\!] \big) \boxtimes [\![!\,\Delta, \Gamma_2 \vdash M_2 : C]\!] \Big) \circ c\ ,$$

where the arrow

$$T\big(\,[\![C]\!] \multimap T[\![A <: B]\!]\,\big) \quad : \quad T([\![C]\!] \multimap T[\![A]\!]) \;\longrightarrow\; T([\![C]\!] \multimap T[\![B]\!])$$

is obtained by applying suitable functors to the arrow $[\![A <: B]\!] : [\![A]\!] \to [\![B]\!]$.

Since $T\big(\,[\![C]\!] \multimap T[\![A <: B]\!]\,\big) \circ [\![!\,\Delta, \Gamma_1 \vdash M_1 : C \multimap A]\!]$ is equal to $[\![!\,\Delta, \Gamma_1 \vdash M_1 : C \multimap B]\!]$, 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 η^T and is the form $\eta^T_{[\![A]\!]} \circ f$ for some arrow $f : [\![\Gamma]\!] \to [\![A]\!]$ (see Definition 5.34). We write $[\![\Gamma \vdash V : A]\!]_{\mathsf{v}}$ for the arrow such that $[\![\Gamma \vdash V : A]\!] = \eta^T_{[\![A]\!]} \circ [\![\Gamma \vdash V : A]\!]_{\mathsf{v}}$ given in Definition 5.34. It is easy to see that $[\![\vdash V : !\,A]\!]_{\mathsf{v}}$ is of the form $!\,f \circ \varphi'$ for some $f : \mathrm{I} \to [\![A]\!]$. Here $\varphi' : \mathrm{I} \overset{\cong}{\Rightarrow} !\,\mathrm{I}$ is from Theorem 4.21.

Lemma Appendix F.2. *Assume that $\Gamma, x : A \vdash M : B$ and $\vdash V : A$ are derivable for a term M and a closed value V. Then the composition $[\![\Gamma, x : A \vdash M : B]\!] \circ (\mathrm{id}_{[\![\Gamma]\!]} \boxtimes [\![\vdash V : A]\!]_{\mathsf{v}})$ is equal to $[\![\Gamma \vdash M[V/x] : B]\!]$.*

In Lemma Appendix F.2, we assume that x is the largest in $|\Gamma| \cup \{x\}$ with respect to the linear order $\prec$ in Definition 5.33. 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 present two cases. When $M \equiv x$, the composition $[\![\Gamma, x : A \vdash x : B]\!] \circ (\mathrm{id}_{[\![\Gamma]\!]} \boxtimes [\![\vdash V : A]\!]_{\mathsf{v}})$ is equal to $\eta^T_{[\![B]\!]} \circ [\![A <: B]\!] \circ [\![\Gamma \vdash V : A]\!]_{\mathsf{v}}$, which is equal to $[\![\Gamma \vdash V : B]\!]$ by Lemma Appendix F.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 $!\,\Delta, \Gamma_1 \vdash M_1 : C \multimap B$ and $!\,\Delta, \Gamma_2 \vdash M_2 : C$ are derivable. The interpretation $[\![\Gamma, x : A \vdash M_1\, M_2 : B]\!]$ is

$$\mu_{[\![B]\!]} \circ \mu_{T[\![B]\!]} \circ TT\mathsf{ev}_{[\![C]\!],T[\![B]\!]} \circ T\mathrm{str}_{[\![C]\!] \multimap T[\![B]\!],[\![C]\!]} \circ \mathrm{str}'_{[\![C]\!] \multimap T[\![B]\!],T[\![C]\!]}$$
$$\circ ([\![!\,\Delta, \Gamma_1 \vdash M_1 : C \multimap B]\!] \boxtimes [\![!\,\Delta, \Gamma_2 \vdash M_2 : C]\!]) \circ c$$

where $c : [\![!\,\Delta, \Gamma_1, \Gamma_2]\!] \to [\![!\,\Delta, \Gamma_1]\!] \boxtimes [\![!\,\Delta, \Gamma_2]\!]$ is a suitable permutation followed by contractions. By the induction hypothesis and naturality of the permutation and contractions, the composition $[\![\Gamma, x : A \vdash M_1\, M_2 : B]\!] \circ (\mathrm{id}_{[\![\Gamma]\!]} \boxtimes [\![\vdash V : A]\!]_{\mathsf{v}})$ is equal to

$$\mu_{[\![B]\!]} \circ \mu_{T[\![B]\!]} \circ TT\mathsf{ev}_{[\![C]\!],T[\![B]\!]} \circ T\mathrm{str}_{[\![C]\!]\multimap T[\![B]\!],[\![C]\!]} \circ \mathrm{str}'_{[\![C]\!]\multimap T[\![B]\!],T[\![C]\!]} \\ \circ ([\![!\,\Delta', \Gamma'_1 \vdash M_1[V/x] : C \multimap B]\!] \boxtimes [\![!\,\Delta', \Gamma'_2 \vdash M_2[V/x] : C]\!]) \circ c'$$

where $!\,\Delta', \Gamma'_1, \Gamma'_2$ is the term environment obtained by removing $x : A$ from $!\,\Delta, \Gamma_1, \Gamma_2$, and $c' : [\![!\,\Delta', \Gamma'_1, \Gamma'_2]\!] \to [\![!\,\Delta', \Gamma'_1]\!] \boxtimes [\![!\,\Delta, \Gamma'_2]\!]$ is a suitable permutation followed by contractions. Hence, $[\![\Gamma, x : A \vdash M_1\, M_2 : B]\!] \circ (\mathrm{id}_{[\![\Gamma]\!]} \boxtimes [\![\vdash V : A]\!]_{\mathsf{v}})$ is equal to $[\![\Gamma \vdash (M_1\, M_2)[V/x] : B]\!]$.

We can prove the other cases in the same way. □

Lemma Appendix F.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. □

Lemma Appendix F.4 (Lemma 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 $[\![\Gamma \vdash E[M] : B]\!] : [\![\Gamma]\!] \to T[\![B]\!]$ is calculated by*

$$[\![\Gamma \vdash E[M] : B]\!] \quad = \quad \mu^T_{[\![B]\!]} \circ T[\![x : A \vdash E[x] : B]\!] \circ [\![\Gamma \vdash M : A]\!] \ .$$

Proof. By induction on the evaluation context E, where we use the characterization in Lemma 3.6. We only present two cases. When $E \equiv [\_]$, since $x : A \vdash x : B$ is derivable, we must have $A <: B$. By Lemma Appendix F.1, $[\![\Gamma \vdash E[M] : B]\!]$ is equal to $T[\![A <: B]\!] \circ [\![\Gamma \vdash M : A]\!]$, 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 $x : A \vdash E'[x] : C \multimap B$ and $\vdash N : C$ are derivable. The interpretation $[\![\Gamma \vdash \big(E'[M]\big)\, N : B]\!]$ is given, by Definition 5.34, by

$$\mu_{[\![B]\!]} \circ \mu_{T[\![B]\!]} \circ TT\mathsf{ev}_{[\![C]\!],T[\![B]\!]} \circ T\mathrm{str}_{[\![C]\!]\multimap T[\![B]\!],[\![C]\!]} \circ \mathrm{str}'_{[\![C]\!]\multimap T[\![B]\!],T[\![C]\!]} \\ \circ ([\![\Gamma \vdash E'[M] : C \multimap B]\!] \boxtimes [\![\vdash N : C]\!]) \ . \quad \text{(F.2)}$$

By the induction hypothesis we have

$$[\![\Gamma \vdash E'[M] : C \multimap B]\!] \quad = \quad \mu_{[\![C \multimap B]\!]} \circ T[\![x : A \vdash E'[x] : C \multimap B]\!] \circ [\![\Gamma \vdash M : A]\!] \ .$$

Using this we see that (F.2) is equal to

$$\mu_{[\![B]\!]} \circ Tf \circ [\![\Gamma \vdash M : A]\!] \ , \quad \text{where}$$

$$f \ := \ \left[\begin{array}{l} \mu_{[\![B]\!]} \circ \mu_{T[\![B]\!]} \circ TT\mathsf{ev}_{[\![C]\!],T[\![B]\!]} \circ T\mathrm{str}_{[\![C]\!]\multimap T[\![B]\!],[\![C]\!]} \circ \\ \mathrm{str}'_{[\![C]\!]\multimap T[\![B]\!],T[\![C]\!]} \circ \big([\![x : A \vdash E'[x] : C \multimap B]\!] \boxtimes [\![\vdash N : C]\!] \big) \end{array} \right] \quad : \quad [\![A]\!] \longrightarrow T[\![B]\!] \ .$$

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. □

Lemma Appendix F.5 (Lemma 6.2, repeated)**.** *For a closed term M such that $\Vdash\vdash M : A$, if there is a reduction $M \to_1 N$ that is not due to a measurement rule ((*$\mathtt{meas}_1$*–*$\mathtt{meas}_4$*) in Definition 3.7), then*

$$[\![\vdash M : A]\!] \quad = \quad [\![\vdash N : A]\!] \ .$$

Note that $\vdash N : A$ is derivable by Lemma 3.23.

Proof. Assume $M \equiv E\big[(\lambda x^C.\, L)\, V\big]$ and $N \equiv E\big[L[V/x]\big]$. By Lemma Appendix F.3, there exists a type B such that $y : B \vdash E[y] : A$ and $\vdash (\lambda x^C.\, L)\, V : B$ are derivable. By Lemma 3.23, we also have $\vdash L[V/x] : B$. Since

$$\begin{aligned}
[\![\vdash E[(\lambda x^C.\, L)\, V] : A]\!] &= \mu_{[\![A]\!]} \circ T[\![x : B \vdash E[x] : A]\!] \circ [\![\vdash (\lambda x^C.\, L)\, V : B]\!] \qquad \text{and} \\
[\![\vdash E[L[V/x]] : A]\!] &= \mu_{[\![A]\!]} \circ T[\![x : B \vdash E[x] : A]\!] \circ [\![\vdash L[V/x] : B]\!]
\end{aligned}$$

by Lemma 6.1, it is enough to show that $[\![\vdash (\lambda x^C.\, L)\, V : B]\!]$ is equal to $[\![\vdash L[V/x] : B]\!]$. By unfolding the definition of the interpretation (Definition 5.34), we obtain

$$[\![\vdash (\lambda x^C.\, L)\, V : B]\!] \quad = \quad [\![x : A \vdash L : B]\!] \circ [\![\vdash V : A]\!]_{\mathsf{v}} \ ;$$

this coincides with $[\![\vdash L[V/x] : B]\!]$ by Lemma Appendix F.2.

For the other reduction rules, we can prove the statement in the same way. □

Lemma Appendix F.6 (Lemma 6.6 (1), repeated)**.** *If (t, V) is in R_A, then $(\eta^T_{[\![A]\!]} \circ t, V)$ is in $R_A^{\top\top}$.*

Proof. For $(k, E) \in R_A^{\top}$, since $\mu_{[\![\mathtt{bit}]\!]} \circ Tk \circ \eta^T_A \circ t = k \circ t$, we have

$$\mu_{[\![\mathtt{bit}]\!]} \circ Tk \circ \eta^T_A \circ t \ \lessdot\ E[V] \ .$$

Therefore $(\eta^T_{[\![A]\!]} \circ t, V)$ is in $R_A^{\top\top}$.

Lemma Appendix F.7 (Lemma 6.6 (2), repeated)**.** *If $(t, V) \in R_A$ and $A <: A'$, then $([\![A <: A']\!] \circ t, V) \in R_{A'}$.*

Proof. By induction on A. When A is $\top$ or n-$\mathtt{qbit}$, the type A' is equal to A, and the statement is straightforward.

When A is $B \boxtimes C$, by the definition of subtyping relation, A' must be of the form $B' \boxtimes C'$ for some $B' :> B$ and $C' :> C$. For $(t \boxtimes s, \langle V, W\rangle) \in R_{B\boxtimes C}$, the composition $[\![B \boxtimes C <: B' \boxtimes C']\!] \circ (t \boxtimes s)$ is equal to $([\![B <: B']\!] \circ t) \boxtimes ([\![C <: C']\!] \circ s)$. By the induction hypothesis, $([\![B <: B']\!] \circ t, V)$ is in $R_{B'}$, and $([\![C <: C']\!] \circ s, W)$ is in $R_{C'}$, and therefore $([\![B\boxtimes C <: B'\boxtimes C']\!] \circ (t \boxtimes s), \langle V, W\rangle)$ is in $R_{B'\boxtimes C'}$. We can similarly show the statement for $A \equiv B + C$.

When A is $B \multimap C$, the type A' is of the form $B' \multimap C'$ for some $B' <: B$ and $C <: C'$. For $(t, V) \in R_{B\multimap C}$ and $(s, W) \in R_{B'}$, by naturality of ev we have

$$\begin{aligned}
&\mathsf{ev}_{[\![B']\!],[\![C']\!]} \circ (([\![B \multimap C <: B' \multimap C']\!] \circ t) \boxtimes s) \\
&\qquad = T[\![C <: C']\!] \circ \mathsf{ev}_{[\![B]\!],[\![C]\!]} \circ (t \boxtimes ([\![B' <: B]\!] \circ s)). \qquad \text{(F.3)}
\end{aligned}$$

By the induction hypothesis, $(\llbracket B' <: B \rrbracket \circ s, W)$ is in R_B. Since (t, V) is in $R_{B \multimap C}$, we see that $\bigl(\mathsf{ev}_{\llbracket B \rrbracket, \llbracket C \rrbracket} \circ (t \boxtimes (\llbracket B' <: B \rrbracket \circ s)), V\,W\bigr)$ is in $R_C^{\top\top}$. By the induction hypothesis, $(k \circ \llbracket C <: C' \rrbracket, E)$ is easily seen to be in $R_C^{\top}$ for any $(k, E) \in R_{C'}^{\top}$. Therefore,

$$\mu_{\llbracket \mathtt{bit} \rrbracket} \circ Tk \circ T\llbracket C <: C' \rrbracket \circ \mathsf{ev}_{\llbracket B \rrbracket, \llbracket C \rrbracket} \circ (t \boxtimes (\llbracket B' <: B \rrbracket \circ s)) \ \lessdot\ E[V\,W]$$

for any $(k, E) \in R_{C'}^{\top}$. By the definition of $R_C^{\top\top}$ and (F.3), we obtain

$$\bigl(\,\mathsf{ev}_{\llbracket B' \rrbracket, \llbracket C' \rrbracket} \circ ((\llbracket B \multimap C <: B' \multimap C' \rrbracket \circ t) \boxtimes s),\ V\,W\,\bigr) \quad \in R_{C'}^{\top\top}$$

for any $(s, W) \in R_{B'}$, which implies that $(\llbracket B \multimap C <: B' \multimap C' \rrbracket \circ t, V)$ is in $R_{B' \multimap C'}$.

When A is $!\,B$, the type A' is of the form $!^n\,B'$ for some $n \geq 0$ and $B <: B'$. If $(!\,t \circ \varphi', V)$ is in R_A, then $\llbracket A <: A' \rrbracket \circ !\,t \circ \varphi' = !^n(\llbracket B <: B' \rrbracket \circ t) \circ !^{n-1}\,\varphi' \circ \cdots \circ !\,\varphi' \circ \varphi'$. By the induction hypothesis, $(\llbracket B <: B' \rrbracket \circ t, V)$ is in $R_{B'}$, and therefore by (53), $(\llbracket A <: A' \rrbracket \circ !\,t \circ \varphi', V)$ is in $R_{A'}$. □

Lemma Appendix F.8 (Lemma 6.7 (1), repeated)**.** *For any type A and $M \in \mathrm{ClTerm}(A)$, we have $([\bot], M) \in R_A^{\top\top}$.*

Proof. Let $(k, E) \in R_A^{\top}$. We claim

$$\mu_{\llbracket \mathtt{bit} \rrbracket} \circ Tk \circ [\bot] \ =\ [\bot]\ , \tag{F.4}$$

where $[\bot]$ denotes the arrow $\mathrm{I} \to T\llbracket A \rrbracket$ in $\mathbf{PER}_{\mathcal{Q}}$ that is realized by $\bot \in A_{\mathcal{Q}}$ (cf. Lemma 5.27, 5.8.2). We have

$$\begin{aligned} \mu &= [\lambda k^{(((\llbracket A \rrbracket \multimap R) \multimap R) \multimap R) \multimap R} y^{\llbracket A \rrbracket \multimap R}.k(\lambda h^{(\llbracket A \rrbracket \multimap R) \multimap R}.hy)] \\ Tk &= [\lambda v^{(\llbracket A \rrbracket \multimap R) \multimap R} x^{((\llbracket \mathtt{bit} \rrbracket \multimap R) \multimap R) \multimap R}.v(\lambda a^{\llbracket A \rrbracket}.x(c_k a))] \end{aligned}$$

where c_k is a choice of a realizer of k. We put type annotations to explain intentions of these realizers. The arrow $\mu_{\llbracket \mathtt{bit} \rrbracket} \circ Tk : T\llbracket A \rrbracket \to T\llbracket \mathtt{bit} \rrbracket$ is realized by

$$\begin{aligned} \lambda v.(\lambda ky.k(\lambda h.hy))((\lambda vx.v(\lambda a.x(c_k a)))v) &= \lambda vy.(\lambda x.v(\lambda a.x(c_k a)))(\lambda h.hy) \\ &\overset{(1)}{=} \lambda vy.v(\lambda a.(\lambda h.hy)(c_k a)) \\ &\overset{(2)}{=} \lambda vy.v(\lambda a.c_k ay). \end{aligned}$$

We note that (1) and (2) follow from Remark 4.12: the LCA $A_{\mathcal{Q}}$ is a model of the untyped linear lambda calculus modulo beta-reductions. Therefore the left-hand side of (F.4) is

$$\bigl[\,\lambda x.\,\lambda y.\,\bot(\lambda a.\,c_k ay)\,\bigr]$$

that is nothing but $[\bot]$ due to the left strictness of application of $A_{\mathcal{Q}}$.

From (F.4) it easily follows that $\mathrm{prob}(\mathrm{tree}(\mu_{\llbracket \mathtt{bit} \rrbracket} \circ Tk \circ [\bot])) = (0, 0)$. Therefore we always have $\mu_{\llbracket \mathtt{bit} \rrbracket} \circ Tk \circ [\bot] \lessdot E[M]$. □

Lemma Appendix F.9 (Lemma 6.7 (2), repeated)**.** *For any type A and $M \in \mathrm{ClTerm}(A)$, if there exists a sequence of realizers $a_1 \sqsubseteq a_2 \sqsubseteq \cdots$ of arrows in $\mathbf{PER}_{\mathcal{Q}}(\mathrm{I}, T[\![A]\!])$ such that $([a_n], M) \in R_A^{\top\top}$, then we have $([\bigvee_{n\geq 1} a_n], M) \in R_A^{\top\top}$.*

Proof. Let (k, E) be an element in $R_A^{\top}$. Since the application of $A_{\mathcal{Q}}$ is continuous, the value assigned to any edge of the tree

$$\mathrm{tree}\Big(\mu_{[\![\mathtt{bit}]\!]} \circ Tk \circ [\bigvee_{n\geq 1} a_n]\Big)$$

is the least upper bound of the value on the corresponding edge of the trees

$$\mathrm{tree}(\mu_{[\![\mathtt{bit}]\!]} \circ Tk \circ [a_n]).$$

Therefore, if we have $\mu_{[\![\mathtt{bit}]\!]} \circ Tk \circ [a_n] \lessdot E[M]$ for every $n \geq 1$, then it follows that $\mu_{[\![\mathtt{bit}]\!]} \circ Tk \circ [\bigvee_{n\geq 1} a_n] \lessdot E[M]$. □

Lemma Appendix F.10. *Let $M \to_1 N$ be a reduction that is not due to a measurement rule (($\mathtt{meas}_1$–$\mathtt{meas}_4$) in Definition 3.7). Then*

$$(t, M) \in R_A^{\top\top} \iff (t, N) \in R_A^{\top\top} \ .$$

Proof. Let (k, E) be an element in $R_A^{\top}$. Assume (t, M) is in $R_A^{\top\top}$; then we have $\mu_{[\![\mathtt{bit}]\!]} \circ Tk \circ t \lessdot E[M]$. By the definition of big-step semantics, $E[M] \curlyvee (p, q)$ if and only if $E[N] \curlyvee (p, q)$. Therefore, $\mu_{[\![\mathtt{bit}]\!]} \circ Tk \circ t \lessdot E[N]$. The other direction is similar. □

Lemma Appendix F.11. *For a type A such that $!(\mathtt{bit} \multimap \mathtt{qbit}) <: A$, the pair $([\![\vdash \mathtt{new} : A]\!], \mathtt{new})$ is in $R_A^{\top\top}$.*

Proof. When $A = \mathtt{bit} \multimap \mathtt{qbit}$, since $([\![\mathtt{new}_\rho]\!]_{\mathrm{const}}, \mathtt{new}_\rho)$ is in $R_{\mathtt{qbit}}$ for any $\rho \in \mathrm{DM}_2$, both $([\![\vdash \mathtt{new}_{|0\rangle\langle 0|} : \mathtt{qbit}]\!], \mathtt{new}\,\mathtt{tt})$ and $([\![\vdash \mathtt{new}_{|1\rangle\langle 1|} : \mathtt{qbit}]\!], \mathtt{new}\,\mathtt{ff})$ are in $R_{\mathtt{qbit}}^{\top\top}$ by Lemma Appendix F.6 and Lemma Appendix F.10. Therefore, $([\![\mathtt{new}]\!]_{\mathrm{const}}, \mathtt{new})$ is in $R_{\mathtt{bit}\multimap\mathtt{qbit}}$, and by Lemma Appendix F.6, $([\![\vdash \mathtt{new} : A]\!], \mathtt{new})$ is in $R_{\mathtt{bit}\multimap\mathtt{qbit}}^{\top\top}$. When $A = !(\mathtt{bit} \multimap \mathtt{qbit})$, we have $\eta^T_{[\![A]\!]} \circ ![\![\mathtt{new}]\!]_{\mathrm{const}} \circ \varphi' = [\![\vdash \mathtt{new} : A]\!]$. Since $([\![\mathtt{new}]\!]_{\mathrm{const}}, \mathtt{new})$ is in $R_{\mathtt{bit}\multimap\mathtt{qbit}}$, the pair $(![\![\mathtt{new}]\!]_{\mathrm{const}} \circ \varphi', \mathtt{new})$ is in $R_{!(\mathtt{bit}\multimap\mathtt{qbit})}$. Therefore, by Lemma Appendix F.6, $([\![\vdash \mathtt{new} : A]\!], \mathtt{new})$ is in $R_A^{\top\top}$. When A satisfies $!(\mathtt{bit} \multimap \mathtt{qbit}) <: A$, the statement follows from Lemma Appendix F.7, Lemma Appendix F.6 and that $(![\![\mathtt{new}]\!]_{\mathrm{const}} \circ \varphi', \mathtt{new})$ is in $R_{!(\mathtt{bit}\multimap\mathtt{qbit})}$. □

Lemma Appendix F.12. *For a type A such that $!(n\text{-}\mathtt{qbit} \multimap n\text{-}\mathtt{qbit}) <: A$, the pair $([\![\vdash U : A]\!], U)$ is in $R_A^{\top\top}$.*

Proof. Similar to the proof of Lemma Appendix F.11. □

Lemma Appendix F.13. *For a type A such that $!(n\text{-}\mathtt{qbit} \boxtimes m\text{-}\mathtt{qbit} \multimap (n+m)\text{-}\mathtt{qbit}) <: A$, the pair $([\![\vdash \mathtt{cmp}_{n,m} : A]\!], \mathtt{cmp}_{n,m})$ is in $R_A^{\top\top}$.*

PROOF. Similar to the proof of Lemma Appendix F.11. $\square$

Lemma Appendix F.14. *For a type A such that $!\big((n+1)\text{-}\mathtt{qbit} \multimap \,!\,\mathtt{bit} \boxtimes n\text{-}\mathtt{qbit}\big) <: A$, the pair $([\![\vdash \mathtt{meas}_i^{n+1} : A]\!], \mathtt{meas}_i^{n+1})$ is in $R_A^{\top\top}$.*

PROOF. First we shall prove that, when $A \equiv (n+1)\text{-}\mathtt{qbit} \multimap \,!\,\mathtt{bit} \boxtimes n\text{-}\mathtt{qbit}$, we have $([\![\mathtt{meas}_i^{n+1}]\!]_{\mathrm{const}}, \mathtt{meas}_i^{n+1})$ is in R_A. It is enough to show that for any $\rho \in \mathrm{DM}_{2^{n+1}}$, the pair $([\![\vdash \mathtt{meas}_i^{n+1}\,\mathtt{new}_\rho : \,!\,\mathtt{bit} \boxtimes n\text{-}\mathtt{qbit}]\!], \mathtt{meas}_i^{n+1}\,\mathtt{new}_\rho)$ is in $R^{\top\top}_{!\,\mathtt{bit}\boxtimes n\text{-}\mathtt{qbit}}$. Let (k, E) be an element in $R^{\top}_{!\,\mathtt{bit}\boxtimes n\text{-}\mathtt{qbit}}$. We define $k_0, k_1 : [\![n\text{-}\mathtt{qbit}]\!] \to T[\![\mathtt{bit}]\!]$ to be the following arrows.

$$k_0 \;=\; k \circ ((!\,\kappa_\ell \circ \varphi') \boxtimes \mathrm{id}_{[\![n\text{-}\mathtt{qbit}]\!]}) \qquad k_1 \;=\; k \circ ((!\,\kappa_r \circ \varphi') \boxtimes \mathrm{id}_{[\![n\text{-}\mathtt{qbit}]\!]})$$

Then $\mathrm{prob}(\mathrm{tree}(\mu_{[\![\mathtt{bit}]\!]} \circ Tk \circ [\![\vdash \mathtt{meas}_i^{n+1}\,\mathtt{new}_\rho : \,!\,\mathtt{bit} \boxtimes n\text{-}\mathtt{qbit}]\!]))$ is equal to

$$(0,0) + \mathrm{prob}(\mathrm{tree}(k_0 \circ [\![\mathtt{new}_{\langle 0_i|\rho|0_i\rangle}]\!]_{\mathrm{const}})) + \mathrm{prob}(\mathrm{tree}(k_1 \circ [\![\mathtt{new}_{\langle 1_i|\rho|1_i\rangle}]\!]_{\mathrm{const}})) \;\;;$$

this is seen much like in the proof of Theorem 6.5. Since

$$\begin{aligned} k_0 \circ [\![\mathtt{new}_{\langle 0_i|\rho|0_i\rangle}]\!]_{\mathrm{const}} &\lessdot E[\langle \mathtt{tt}, \mathtt{new}_{\langle 0_i|\rho|0_i\rangle}\rangle] \quad \text{and} \\ k_1 \circ [\![\mathtt{new}_{\langle 1_i|\rho|1_i\rangle}]\!]_{\mathrm{const}} &\lessdot E[\langle \mathtt{ff}, \mathtt{new}_{\langle 1_i|\rho|1_i\rangle}\rangle] \;\;, \end{aligned}$$

it follows that $\mu_{[\![\mathtt{bit}]\!]} \circ Tk \circ [\![\vdash \mathtt{meas}_i^{n+1}\,\mathtt{new}_\rho : \,!\,\mathtt{bit}\boxtimes n\text{-}\mathtt{qbit}]\!] \lessdot E[\mathtt{meas}_i^{n+1}\,\mathtt{new}_\rho]$, and therefore, $\big([\![\vdash \mathtt{meas}_i^{n+1}\,\mathtt{new}_\rho : \,!\,\mathtt{bit}\boxtimes n\text{-}\mathtt{qbit}]\!], \mathtt{meas}_i^{n+1}\,\mathtt{new}_\rho\big)$ is in $R^{\top\top}_{!\,\mathtt{bit}\boxtimes n\text{-}\mathtt{qbit}}$.

When $A \equiv \,!\big((n+1)\text{-}\mathtt{qbit} \multimap \,!\,\mathtt{bit}\boxtimes n\text{-}\mathtt{qbit}\big)$, since $([\![\vdash \mathtt{meas}_i^{n+1}]\!]_{\mathrm{const}}, \mathtt{meas}_i^{n+1})$ is in $R_{(n+1)\text{-}\mathtt{qbit}\multimap !\,\mathtt{bit}\boxtimes n\text{-}\mathtt{qbit}}$, the pair $(![\![\mathtt{meas}_i^{n+1}]\!]_{\mathrm{const}} \circ \varphi', \mathtt{meas}_i^{n+1})$ is in R_A. Therefore, by Lemma Appendix F.6, $([\![\vdash \mathtt{meas}_i^{n+1} : A]\!], \mathtt{meas}_i^{n+1})$ is in $R_A^{\top\top}$.

Finally, when A satisfies $!((n+1)\text{-}\mathtt{qbit} \multimap \,!\,\mathtt{bit}\boxtimes \mathtt{qbit}) <: A$, the statement follows from Lemma Appendix F.7, Lemma Appendix F.6 and that $(![\![\mathtt{meas}_i^{n+1}]\!]_{\mathrm{const}} \circ \varphi', \mathtt{meas}_i^{n+1})$ is in $R_{!((n+1)\text{-}\mathtt{qbit}\multimap !\,\mathtt{bit}\boxtimes\mathtt{qbit})}$. $\square$

Lemma Appendix F.15. *For a type A such that $!(\mathtt{qbit} \multimap \,!\,\mathtt{bit}) <: A$, the pair $([\![\vdash \mathtt{meas}_1^1 : A]\!], \mathtt{meas}_1^1)$ is in $R_A^{\top\top}$.*

PROOF. Similar to the proof of Lemma Appendix F.14. $\square$

[1] I. Hasuo, N. Hoshino, Semantics of higher-order quantum computation via geometry of interaction, in: LICS, IEEE Computer Society, 2011, pp. 237–246.

[2] B. Valiron, Quantum computation: From a programmer's perspective, New Generation Comput. 31 (1) (2013) 1–26.

[3] B. Ömer, Quantum programming in QCL, Master's thesis, Institute of Information Systems, Technical University of Vienna (2000).

[4] A. Green, T. Altenkirch, The quantum IO monad, in: S. J. Gay, I. Mackie (Eds.), Semantic Techniques in Quantum Computation, Cambridge Univ. Press, 2009, Ch. 5, pp. 173–205.

[5] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, B. Valiron, Quipper: a scalable quantum programming language, in: H.-J. Boehm, C. Flanagan (Eds.), PLDI, ACM, 2013, pp. 333–342.

[6] Y. Delbecque, P. Panangaden, Game semantics for quantum stores, Elect. Notes in Theor. Comp. Sci. 218 (2008) 153–170.

[7] P. Selinger, Towards a quantum programming language, Math. Struct. in Comp. Sci. 14 (4) (2004) 527–586.

[8] P. Selinger, B. Valiron, On a fully abstract model for a quantum linear functional language: (extended abstract), Elect. Notes in Theor. Comp. Sci. 210 (2008) 123–137.

[9] P. Selinger, B. Valiron, Quantum lambda calculus, in: S. Gay, I. Mackie (Eds.), Semantic Techniques in Quantum Computation, Cambridge Univ. Press, 2009, pp. 135–172.

[10] U. Dal Lago, A. Masini, M. Zorzi, Confluence results for a quantum lambda calculus with measurements, Elect. Notes in Theor. Comp. Sci. 270 (2) (2011) 251–261.

[11] U. Dal Lago, C. Faggian, On multiplicative linear logic, modality and quantum circuits, in: B. Jacobs, P. Selinger, B. Spitters (Eds.), QPL, Vol. 95 of EPTCS, 2011, pp. 55–66.

[12] J. Grattage, An overview of qml with a concrete implementation in Haskell, Elect. Notes in Theor. Comp. Sci. 270 (1) (2011) 165–174.

[13] A. van Tonder, A lambda calculus for quantum computation, SIAM J. Comput. 33 (5) (2004) 1109–1135.

[14] O. Malherbe, Categorical models of computation: Partially traced categories and presheaf models of quantum computation, Ph.D. thesis, Univ. of Ottawa (2010).

[15] M. Pagani, P. Selinger, B. Valiron, Applying quantitative semantics to higher-order quantum computing, in: S. Jagannathan, P. Sewell (Eds.), POPL, ACM, 2014, pp. 647–658.

[16] T. Ehrhard, Finiteness spaces, Math. Struct. in Comp. Sci. 15 (4) (2005) 615–646.

[17] V. Danos, T. Ehrhard, Probabilistic coherence spaces as a model of higher-order probabilistic computation, Inf. Comput. 209 (6) (2011) 966–991.

[18] J.-Y. Girard, Normal functors, power series and lambda-calculus, Ann. Pure & Appl. Logic 37 (2) (1988) 129–177.

[19] R. Hasegawa, Two applications of analytic functors, Theor. Comp. Sci. 272 (1–2) (2002) 113–175.

[20] J.-Y. Girard, Geometry of interaction I: Interpretation of system F, in: R. Ferro, et al. (Eds.), Logic Colloquium '88, North-Holland, 1989, pp. 221–260.

[21] S. Abramsky, E. Haghverdi, P. Scott, Geometry of interaction and linear combinatory algebras, Math. Struct. in Comp. Sci. 12 (5) (2002) 625–665. doi:http://dx.doi.org/10.1017/S0960129502003730.

[22] S. Abramsky, R. Jagadeesan, P. Malacaria, Full abstraction for PCF, Inf. & Comp. 163 (2) (2000) 409–470.

[23] J. M. E. Hyland, C.-H. L. Ong, On full abstraction for PCF: I, II, and III, Inf. & Comp. 163 (2) (2000) 285–408.

[24] I. Mackie, The geometry of interaction machine, in: POPL '95: Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages, ACM, New York, NY, USA, 1995, pp. 198–208. doi:http://doi.acm.org/10.1145/199448.199483.

[25] D. R. Ghica, A. I. Smith, Geometry of synthesis ii: From games to delay-insensitive circuits, Electr. Notes Theor. Comput. Sci. 265 (2010) 301–324.

[26] D. R. Ghica, A. I. Smith, Geometry of synthesis iii: resource management through type inference, in: T. Ball, M. Sagiv (Eds.), POPL, ACM, 2011, pp. 345–356.

[27] D. R. Ghica, A. I. Smith, S. Singh, Geometry of synthesis IV: compiling affine recursion into static hardware, in: M. M. T. Chakravarty, Z. Hu, O. Danvy (Eds.), ICFP, ACM, 2011, pp. 221–233.

[28] K. Muroya, T. Kataoka, I. Hasuo, N. Hoshino, Compiling effectful terms to transducers: Prototype implementation of memoryful geometry of interaction, in: International Workshop on Syntax and Semantics of Low-Level Language (LOLA 2014), 2014.

[29] I. Hasuo, B. Jacobs, A. Sokolova, Generic trace semantics via coinduction, Logical Methods in Comp. Sci. 3 (4:11).

[30] G. M. Bierman, What is a categorical model of intuitionistic linear logic, in: M. Dezani-Ciancaglini, G. Plotkin (Eds.), Typed Lambda Calculi and Applications, no. 902 in Lect. Notes Comp. Sci., Springer, Berlin, 1995, pp. 78–93.

[31] P. N. Benton, P. Wadler, Linear logic, monads and the lambda calculus, in: LICS, 1996, pp. 420–431.

[32] S. Abramsky, M. Lenisa, Linear realizability and full completeness for typed lambda-calculi, Ann. Pure & Appl. Logic 134 (2–3) (2005) 122–168.

[33] B. Jacobs, From coalgebraic to monoidal traces, in: Coalgebraic Methods in Computer Science (CMCS 2010), Vol. 264 of Elect. Notes in Theor. Comp. Sci., Elsevier, Amsterdam, 2010, pp. 125–140.

[34] J. R. Longley, Realizability toposes and language semantics, Ph.D. thesis, Edinburgh Univ. (1994).

[35] N. Hoshino, Linear realizability, in: J. Duparc, T. A. Henzinger (Eds.), CSL, Vol. 4646 of Lect. Notes Comp. Sci., Springer, 2007, pp. 420–434.

[36] I. Hasuo, N. Hoshino, Semantics of higher-order quantum computation via geometry of interaction, preprint version of the current paper, supplemented with appendices. Available at `http://arxiv.org/abs/15??.?????` (2015).

[37] K. Kraus, States, effects and operations. Fundamental notions of quantum theory, Vol. 190 of Lect. Notes Phys., Springer-Verlag, 1983.

[38] M. A. Nielsen, I. L. Chuang, Quantum Computation and Quantum Information, Cambridge Univ. Press, 2000.

[39] E. Moggi, Notions of computation and monads, Inf. & Comp. 93(1) (1991) 55–92.

[40] J. J. M. M. Rutten, Universal coalgebra: a theory of systems, Theor. Comp. Sci. 249 (2000) 3–80.

[41] B. Jacobs, Introduction to coalgebra. Towards mathematics of states and observations, Draft of a book (ver. 2.0), available online (2012).

[42] M. B. Smyth, G. D. Plotkin, The category theoretic solution of recursive domain equations, SIAM Journ. Comput. 11 (1982) 761–783.

[43] J. S. Pinto, Implantation parallèle avec la logique linéaire (applications des réseaux d'interaction et de la géométrie de l'interaction), Ph.D. thesis, École Polytechnique, Main text in English (2001).

[44] A. Joyal, R. Street, D. Verity, Traced monoidal categories, Math. Proc. Cambridge Phil. Soc. 119(3) (1996) 425–446.

[45] J. R. B. Cockett, R. A. G. Seely, Linearly distributive functors, Journ. of Pure & Appl. Algebra 143 (1–3) (1999) 155–203. doi:DOI:10.1016/S0022-4049(98)00110-8.
URL http://www.sciencedirect.com/science/article/B6V0K-3XV2JDC-6/2/2826c60de2a7d05eb7e8

[46] R. A. G. Seely, Linear logic, ∗-autonomous categories and cofree coalgebras, in: J. Gray, A. Scedrov (Eds.), Categories in Computer Science and Logic, no. 92 in AMS Contemp. Math., Providence, 1989, pp. 371–382.

[47] P.-A. Melliès, Categorical semantics of linear logic, Vol. 27 of Panoramas et Synthèses, Société Mathématique de France, 2009, Ch. 1, pp. 15–215.

[48] B. Day, On closed categories of functors, in: G. M. Kelly (Ed.), Proc. Sydney Category Theory Seminar 1972/1973, no. 420 in Lect. Notes Math., Springer, Berlin, 1974, pp. 20–54.

[49] U. Dal Lago, M. Zorzi, Wave-style token machines and quantum lambda calculi, in: S. Alves, I. Cervesato (Eds.), Proceedings Third International Workshop on Linearity, LINEARITY 2014, Vienna, Austria, 13th July, 2014., Vol. 176 of EPTCS, 2015, pp. 64–78. doi:10.4204/EPTCS.176.6.
URL http://dx.doi.org/10.4204/EPTCS.176.6

[50] A. Yoshimizu, I. Hasuo, C. Faggian, U. Dal Lago, Measurements in proof nets as higher-order quantum circuits, in: Z. Shao (Ed.), ESOP, Vol. 8410 of Lecture Notes in Computer Science, Springer, 2014, pp. 371–391. doi:10.1007/978-3-642-54833-8.
URL http://dx.doi.org/10.1007/978-3-642-54833-8

[51] U. Dal Lago, C. Faggian, I. Hasuo, A. Yoshimizu, The geometry of synchronization, in: Henzinger and Miller [85], p. 35. doi:10.1145/2603088.2603154.
URL http://doi.acm.org/10.1145/2603088.2603154

[52] U. Dal Lago, C. Faggian, B. Valiron, A. Yoshimizu, Parallelism and synchronization in an infinitary context, in: 30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015, IEEE, 2015, pp. 559–572. doi:10.1109/LICS.2015.58.
URL http://dx.doi.org/10.1109/LICS.2015.58

[53] G. D. Plotkin, Call-by-name, call-by-value and the lambda-calculus, Theor. Comput. Sci. 1 (2) (1975) 125–159. doi:10.1016/0304-3975(75)90017-1.
URL http://dx.doi.org/10.1016/0304-3975(75)90017-1

[54] J. Lambek, P. J. Scott, Introduction to higher order Categorical Logic, no. 7 in Cambridge Studies in Advanced Mathematics, Cambridge Univ. Press, 1986.

[55] R. L. Crole, Categories for Types, Cambridge Mathematical Textbooks, Cambridge Univ. Press, 1993.

[56] B. Valiron, Semantics for a higher order functional programming language for quantum computation, Ph.D. thesis, Univ. Ottawa (2008).

[57] N. Gisin, A. Methot, V. Scarani, Pseudo-telepathy: input cardinality and bell-type inequalities, International Journal of Quantum Information 5 (4) (2007) 525–534.

[58] M. A. Arbib, E. G. Manes, Algebraic Approaches to Program Semantics, Texts and Monogr. in Comp. Sci.,, Springer, Berlin, 1986.

[59] E. Haghverdi, A categorical approach to linear logic, geometry of proofs and full completeness, Ph.D. thesis, Univ. of Ottawa (2000).

[60] M. Ying, N. Yu, Y. Feng, Alternation in quantum programming: From superposition of data to superposit
CoRR abs/1402.5172.
URL `http://arxiv.org/abs/1402.5172`

[61] C. Badescu, P. Panangaden, Quantum alternation: Prospects and problems, in: International Workshop on Quantum Physics and Logic (QPL 2015), 2015, to appear.

[62] V. Danos, L. Regnier, Reversible, irreversible and optimal lambda-machines, Theor. Comput. Sci. 227 (1-2) (1999) 79–97.
`doi:10.1016/S0304-3975(99)00049-3`.
URL `http://dx.doi.org/10.1016/S0304-3975(99)00049-3`

[63] M. Hasegawa, On traced monoidal closed categories, Math. Struct. in Comp. Sci. 19 (2) (2009) 217–244.

[64] A. Joyal, R. Street, The geometry of tensor calculus, I, Adv. Math 88 (1991) 55–112.

[65] P.-A. Melliès, Functorial boxes in string diagrams, in: Z. Ésik (Ed.), CSL, Vol. 4207 of Lect. Notes Comp. Sci., Springer, 2006, pp. 1–30.

[66] E. W. Dijkstra, A Discipline of Programming, Prentice Hall, 1976.

[67] C. A. R. Hoare, An axiomatic basis for computer programming, Commun. ACM 12 (1969) 576–580, 583.

[68] B. Jacobs, B. Westerbaan, B. Westerbaan, States of convex sets, in: A. M. Pitts (Ed.), Foundations of Software Science and Computation Structures - 18th International Conference, FoSSaCS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software,

ETAPS 2015, London, UK, April 11-18, 2015. Proceedings, Vol. 9034 of Lecture Notes in Computer Science, Springer, 2015, pp. 87–101. doi:10.1007/978-3-662-46678-0.
URL http://dx.doi.org/10.1007/978-3-662-46678-0

[69] J. van Oosten, Realizability: An Introduction to its Categorical Side, Studies in Logic and the Foundations of Mathematics, Elsevier, 2008.
URL https://books.google.co.jp/books?id=0Fvvurmr7FsC

[70] A. K. Simpson, Reduction in a linear lambda-calculus with applications to operational semantics, in: J. Giesl (Ed.), RTA, Vol. 3467 of Lecture Notes in Computer Science, Springer, 2005, pp. 219–234.

[71] A. Yoshimizu, On geometry of interaction construction and its applications, BSc thesis, the University of Tokyo, available at http://www-mmm.is.s.u-tokyo.ac.jp/~ayoshimizu/ (2012).

[72] P. N. Benton, A mixed linear and non-linear logic: Proofs, terms and models (extended abstract), in: L. Pacholski, J. Tiuryn (Eds.), CSL, Vol. 933 of Lect. Notes Comp. Sci., Springer, 1994, pp. 121–135.

[73] M. Abadi, G. D. Plotkin, A per model of polymorphism and recursive types, in: LICS, IEEE Computer Society, 1990, pp. 355–365.

[74] M. Abadi, $\top\top$-closed relations and admissibility, Mathematical Structures in Computer Science 10 (3) (2000) 313–320.

[75] D. Pattinson, An introduction to the theory of coalgebras, Course notes for NASSLLI, available online (2003).

[76] J. Worrell, On the final sequence of a finitary set functor, Theor. Comp. Sci. 338 (1–3) (2005) 184–199.

[77] N. Hoshino, K. Muroya, I. Hasuo, Memoryful geometry of interaction: from coalgebraic components to alg
in: Henzinger and Miller [85], p. 52. doi:10.1145/2603088.2603124.
URL http://doi.acm.org/10.1145/2603088.2603124

[78] G. D. Plotkin, J. Power, Adequacy for algebraic effects, in: F. Honsell, M. Miculan (Eds.), FoSSaCS, Vol. 2030 of Lecture Notes in Computer Science, Springer, 2001, pp. 1–24.

[79] J. C. Mitchell, Foundations of Programming Languages, MIT Press, Cambridge, MA, 1996.

[80] G. D. Plotkin, J. Power, D. Sannella, R. D. Tennent, Lax logical relations, in: U. Montanari, J. D. P. Rolim, E. Welzl (Eds.), ICALP, Vol. 1853 of Lecture Notes in Computer Science, Springer, 2000, pp. 85–102.

[81] S. Katsumata, Relating computational effects by $\top\top$-lifting, Inf. Comput. 222 (2013) 228–246.

[82] P. Scott, Tutorial on geometry of interaction, Tutorial talk at FMCS 2004, slides available online (2004).

[83] L. Barbosa, Components as coalgebras, Ph.D. thesis, Univ. Minho (2001).

[84] I. Hasuo, B. Jacobs, Traces for coalgebraic components, Math. Struct. in Comp. Sci. 21 (2) (2011) 267–320.

[85] T. A. Henzinger, D. Miller (Eds.), Joint Meeting of the Twenty-Third EACSL Annual Conference on Com
ACM, 2014.
URL `http://dl.acm.org/citation.cfm?id=2603088`