Semantics of Higher-Order Quantum Computation via Geometry of Interaction

Ichiro Hasuo
Dept. Computer Science, University of Tokyo, Japan

Naohiko Hoshino
RIMS, Kyoto University, 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 based on linear lambda calculus and is 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 also 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*-quantum computation; lambda calculus; categorical semantics; geometry of interaction; realizability**

I. Introduction

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. The latter is being even tested for practical 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 trying to find a new "useful" algorithm. On quantum communication, the unintuitive 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 to solve these problems. Structured programming often leads to discovery of ingenious algorithms; well-formulated semantics would provide a ground for proving a communication protocol correct.

Towards this direction, there have been proposed several high-level languages tailored for quantum computation. Among them are those based on *linear λ-calculus*: while λ-calculus is a prototype of functional programming languages—inherently supporting higher-order computation—linearity provides a useful means of prohibiting duplication of quantum data ("no-cloning"). Examples of such languages are found in [1]–[4]. However, the study of their semantics is still at its infancy. There are only a few denotational models proposed; among them is the one in [2] which, however, fail to support the ! modality (hence erasure and duplication of classical data). In fact it seems to be an open problem to find a denotational model (aside from a term model) that supports full language features—the ! modality, recursion, etc.—of a quantum functional programming language. See [3] for a survey and an axiomatic study of such models.

In this paper we present a new language $\mathbf{q}\lambda_\ell$ and its denotational model that supports its full features. The language $\mathbf{q}\lambda_\ell$ is based on Selinger and Valiron's [3]—in particular we share their principle of "quantum data, classical control"—but is modified for a better fit to our denotational model. We also define its operational semantics and prove adequacy.

For the construction of the denotational model we employ a series of existing techniques in theoretical computer science (Fig. 1). Namely: 1) a monad with an order structure for modeling branching, used in the coalgebraic study of state-based systems (e.g. in [5]); 2) Girard's *Geometry of Interaction (GoI)* [6], categorically formulated by Abramsky, Haghverdi and Scott [7], 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 (used e.g. in [8]); 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.

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* [10], [11]; our denotational model, therefore, has a strong operational flavor. We are currently working on extracting abstract machines for quantum computation, much like in [12]. Our model is also one answer to the question "Quantum GoI?" raised in [13].

Organization of the paper: In §II we fix the notations for quantum computation and briefly describe the semantical techniques used later. In §III we introduce the *quantum branching monad $\mathcal{Q}$* on $\mathbf{Sets}$, from whose Kleisli category

$\mathcal{K}\ell(\mathcal{Q})$ we obtain the category $\mathbf{PER}_{\mathcal{Q}}$. We introduce our language $\mathbf{q}\lambda_\ell$ in §IV, which is interpreted in $\mathbf{PER}_{\mathcal{Q}}$ in §V with the help of a continuation monad. Finally in §VI we present operational semantics and prove adequacy.

Due to space limitation, most proofs as well as a few technical definitions are deferred to an extended version [14].

II. Preliminaries

A. Quantum Computation

We follow Kraus' formulation [15] of quantum mechanics, which is by now standard and is used in e.g. [1], [16]. Due to the limited space we only list definitions and results that are used later; for proofs and more detailed explanation, our principal reference is the standard textbook [16, Chap. 3 & Chap. 8]. Notations: $\mathcal{I}_m$ denotes the $m \times m$ identity matrix; $A^\dagger$ denotes a matrix A's adjoint (i.e. conjugate transpose).

A *quantum state*—a state of a quantum-dynamic system—is represented by a *density matrix*. For us such a system will consist of N qubits, in which case the system is 2^N-dimensional.

Definition II.1 (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 $\mathtt{tr}(\rho) \in [0,1]$. Here $[0,1]$ denotes the unit interval. The set of all m-dim. density matrices is denoted by D_m.

Note that we allow density matrices with trace less than 1.

The following order is standard and used e.g. in [1], [16].

Definition II.2 (Löwner partial order)**.** The order $\sqsubseteq$ on the set D_m of density matrices is defined by: $\rho \sqsubseteq \sigma$ if and only if $\sigma - \rho$ is a positive matrix.

The following fact is crucial in this work. It is proved in [1, Prop. 3.6] using a translation into quadratic forms; in [14, Appendix A] we present another proof using matrix norms.

Lemma II.3. *The relation $\sqsubseteq$ in Def. II.2 is a partial order. Moreover it is an ω-CPO: an increasing chain has the least upper bound.* □

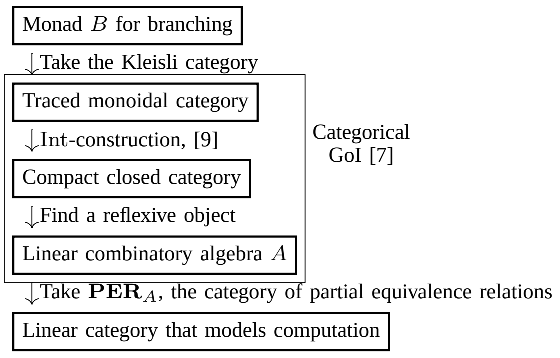

Monad B for branching
↓Take the Kleisli category
Traced monoidal category
↓Int-construction, [9]
Compact closed category
↓Find a reflexive object
Linear combinatory algebra A
Categorical GoI [7]
↓Take $\mathbf{PER}_A$, the category of partial equivalence relations
Linear category that models computation

Figure 1. The construction of the model

Definition II.4 (Quantum operation, QO)**.** A *quantum operation (QO)* from an m-dim. system to an n-dim. system is a mapping $\mathcal{E} : D_m \to D_n$ subject to the following axioms.

1) (Trace condition) $\mathtt{tr}[\mathcal{E}(\rho)] \in [0,1]$ for any $\rho \in D_m$.
2) (Linearity) Let $(\rho_i)_{i\in I}$ be a family of m-dim. density matrices; and $(p_i)_{i\in I}$ be a probability subdistribution (meaning $\sum_i p_i \le 1$). Then: $\mathcal{E}\bigl(\sum_{i\in I} p_i\rho_i\bigr) = \sum_{i\in I} p_i\mathcal{E}(\rho_i)$.
3) (Complete positivity) An arbitrary "extension" of $\mathcal{E}$ of the form $\mathcal{I}_k \otimes \mathcal{E} : M_k \otimes M_m \to M_k \otimes M_n$ carries a positive matrix to a positive one.

The set of QOs from an m-dim. system to an n-dim. one shall be denoted by $\mathrm{QO}_{m,n}$.

We extend the order $\sqsubseteq$ in Def. II.2 in a pointwise manner to obtain an order between QOs. This is done also in [1].

Definition II.5 (Order $\sqsubseteq$ on $\mathrm{QO}_{m,n}$)**.** Given $\mathcal{E}, \mathcal{F} \in \mathrm{QO}_{m,n}$, we have $\mathcal{E} \sqsubseteq \mathcal{F}$ if and only if $\mathcal{E}(\rho) \sqsubseteq \mathcal{F}(\rho)$ for each $\rho \in D_m$. The latter $\sqsubseteq$ is the Löwner partial order (Def. II.2).

Proposition II.6. *The order $\sqsubseteq$ on $\mathrm{QO}_{m,n}$ is an ω-CPO.*

Proof. See [14, Appendix A]; also [1, Lem. 6.4]. □

B. Monads for Branching

The notion of *monad* is standard in category theory. In computer science, after Moggi [17], the notion has been used for encapsulating *computational effect* in functional programming. One such monad T appears in this paper—at the last stage, as a part of a categorical model.

There is another monad $\mathcal{Q}$—called the *quantum branching monad*—that marks the beginning of our development. The idea is drawn from the coalgebraic study of state-based systems; in particular from the use of a monad B on $\mathbf{Sets}$ for modeling *branching*, e.g. in [5]. Some examples of B are: 1) the *lift monad* $\mathcal{L}X = 1 + X$ modeling potential nontermination; 2) the *powerset monad* $\mathcal{P}$ modeling nondeterminism; and 3) the *subdistribution monad* $\mathcal{D}X = \{d : X \to [0,1] \mid \sum_x d(x) \le 1\}$ modeling probabilistic branching.

A feature of such a "branching" monad B is that its Kleisli category $\mathcal{K}\ell(B)$ is ω-$\mathbf{CPO}$ enriched. This feature is used for identifying a final coalgebra in $\mathcal{K}\ell(B)$; the latter turns out to be a fully abstract semantic domain for (execution-trace based) *trace semantics* for state-based systems. See [5].

C. Geometry of Interaction

Girard's *Geometry of Interaction (GoI)* [6] is an interpretation of linear logic in terms of dynamic information flow. Its spirit is close to that of the game-based interpretations of computation [10], [11]. Later, Abramsky, Haghverdi and Scott [7] worked on a categorical foundation of GoI and 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 [7]: among other conditions, a crucial one is that $\mathbb{C}$ is a *traced symmetric monoidal category (TSMC)* [9]. Then applying what they call the *GoI construction* $\mathcal{G}$—isomorphic to the Int*-construction* [9]—yields 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* [18], [19]—but not quite, lacking an appropriate operator for modeling the ! modality. A step ahead is taken in [7]: 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.

D. 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; we shall be based on its formulation found in [8]. It goes as follows. Starting from an LCA A, we define the category $\mathbf{PER}_A$ of *partial equivalence relations (PER)* 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* $c \in A$.[1]

To turn $\mathbf{PER}_A$ into a model of a typed linear λ-calculus (i.e. a linear category) one needs operators like $\otimes$, $\multimap$ and ! on $\mathbf{PER}_A$. They can be defined by "programming in untyped linear λ-calculus"—it is much like encoding pairs and natural numbers in the λ-calculus. See [8] for details.

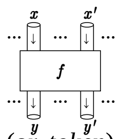
III. The Quantum Branching Monad

A. Background

The starting point of our development is Jacobs' observation [21] that relates: monads for branching (§II-B, used in coalgebraic *trace* semantics) and *traced* monoidal categories that appear in categorical GoI (§II-C).

The examples of a TSMC $\mathbb{C}$ in GoI [7] are divided into two groups: the *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 latter includes: the category $\mathbf{Pfn}$ of sets and partial functions; $\mathbf{Rel}_+$ of sets and binary relations; and $\mathbf{SRel}$ of measurable spaces and stochastic relations. These are, in fact, (close to) the Kleisli categories for the "branching" monads in §II-B. Generalizing this observation, Jacobs [21] proves that a monad B for branching—i.e. a monad on $\mathbf{Sets}$ with order enrichment, subject to some additional conditions—has its Kleisli category $\mathcal{K}\ell(B)$ traced monoidal, with $+$ being its monoidal structure.

[1]Another standard technique is to use *ω-sets* (also called *assemblies*) in place of PERs. This is done for LCAs in [20].



Let us elaborate on such Kleisli category $\mathcal{K}\ell(B)$. We look at it as a category of *piping*. An arrow[2] $f : X \twoheadrightarrow Y$ in $\mathcal{K}\ell(B)$ is understood as a bunch of pipes, with $|X|$-many entrances and $|Y|$-many exits.[3] The pipes are where a *particle* (or *token*) runs through.

According to the choice of a monad B, different "branching" of such pipes is allowed. With $B = \mathrm{id}$ each entrance $x \in X$ is connected to a unique exit $y = f(x)$: a token entering at x is led to the unique exit $f(x)$. When $B = \mathcal{L}$ a pipe can be "stuck" or "looped": a token entering at x may not come out. When $B = \mathcal{P}$ a pipe can branch into multiple ones, with one entrance connected to possibly multiple exits.

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

Our road-map is (see Fig. 1): we fix a branching monad B; then after some steps we obtain a category $\mathbf{PER}_A$ that models linear λ-calculus. For additional features of a calculus (such as nondeterminism) we would need corresponding structures in the ingredients—ultimately in the monad B.

For our purpose, therefore, we shall introduce a branching monad $\mathcal{Q}$ that supports "quantum branching." In an arrow in $\mathcal{K}\ell(\mathcal{Q})$ thought of as piping, a token that runs through is not simply a particle any more but is a quantum state now.

Definition III.1 (The monad $\mathcal{Q}$)**.** The *quantum branching monad* $\mathcal{Q} : \mathbf{Sets} \to \mathbf{Sets}$ is defined as follows. On objects,

$$\mathcal{Q}X = \Bigl\{ c : X \to \prod_{m,n\in\mathbb{N}} \mathrm{QO}_{m,n} \;\Bigm|\; \text{the trace condition (1)} \Bigr\}$$

where the *trace condition* is:

$$\sum_{x\in X}\sum_{n\in\mathbb{N}} \mathsf{tr}\bigl[\bigl(c(x)\bigr)_{m,n}(\rho)\bigr] \le 1 \;\;,\; \forall m \in \mathbb{N},\ \forall \rho \in D_m. \tag{1}$$

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 $c \in \mathcal{Q}X$, $y \in Y$:

$$\bigl(\,(\mathcal{Q}f)(c)(y)\,\bigr)_{m,n} := \textstyle\sum_{x\in f^{-1}(\{y\})}\bigl(\,c(x)\,\bigr)_{m,n}\;. \tag{2}$$

As for the monad structure, its unit $\eta_X : X \to \mathcal{Q}X$ is:

$$\bigl(\,\eta_X(x)(x')\,\bigr)_{m,n} := \begin{cases} \mathcal{I}_m & \text{if } x = x' \text{ and } m = n, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

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

$$\bigl(\,\mu_X(\gamma)(x)\,\bigr)_{m,n} := \sum_{c\in\mathcal{Q}X}\sum_{k\in\mathbb{N}}\bigl(c(x)\bigr)_{k,n} \circ \bigl(\gamma(c)\bigr)_{m,k}. \tag{4}$$

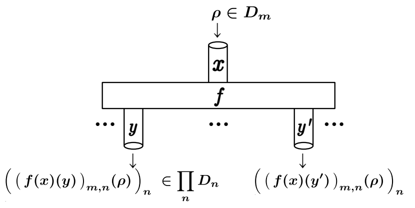
The QO $\bigl(c(x)\bigr)_{k,n} \circ \bigl(\gamma(c)\bigr)_{m,k}$ on the RHS is the *sequential composition* of QOs.

In [14, Appendix B] we prove that the sums in (2) and (4) exist, as well as that $\mathcal{Q}$ is indeed a functor, and is a monad.

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

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

Let us look at a Kleisli arrow $f : X \nrightarrow Y$ as piping. Each entrance $x \in X$ is ready for an incoming token $\rho \in D_m$ of any finite dimension m. Such a token gives rise to one outcoming token; however its exit can be any exit $y \in Y$ and the quantum state associated with the token can be of any finite dimension $n \in N$. The n-dim. quantum state that is to come out of y, is $\bigl(f(x)(y) \bigr)_{m,n}(\rho) \in D_n$.


$\rho \in D_m$
x
f
y
y'
$\Bigl(\bigl(f(x)(y) \bigr)_{m,n}(\rho) \Bigr)_n \in \prod_n D_n$
$\Bigl(\bigl(f(x)(y') \bigr)_{m,n}(\rho) \Bigr)_n$

The trace condition (1) now reads:

$$\sum_{y \in Y} \sum_{n \in \mathbb{N}} \mathsf{tr}\Bigl[\bigl(f(x)(y)\bigr)_{m,n}(\rho)\Bigr] \leq 1 \ , \ \forall m, \rho. \tag{5}$$

The value $\mathsf{tr}\Bigl[\bigl(f(x)(y)\bigr)_{m,n}(\rho)\Bigr]$ is the (observational) probability with which a token ρ entering at x leads to an n-dim. token at y. Such values must add up to at most 1; this is (5).

The composition $\odot$ of Kleisli arrows sequentially connects piping, one after another. See [14, Appendix B].

The monad $\mathcal{Q}$ indeed satisfies the conditions in [21]—equipped with a suitable order—so that the Kleisli category $\mathcal{K}\ell(\mathcal{Q})$ is a traced symmetric monoidal category (TSMC).

Definition III.2 (Order $\sqsubseteq$ on $\mathcal{Q}X$)**.** We endow the set $\mathcal{Q}X$ with the pointwise extension of the order in Def. II.5.

Theorem III.3. *The monad $\mathcal{Q}$ on* $\mathbf{Sets}$ *satisfies the condition [21, Requirements 4.7]. Therefore by [21, Prop. 4.8], $\mathcal{K}\ell(\mathcal{Q})$ is partially additive. In particular by [22, Chap. 3], $(\mathcal{K}\ell(\mathcal{Q}), +, 0)$ is a TSMC.* □

In [14, Appendix B] we spell out the condition [21, Requirements 4.7] and prove that it is satisfied.

C. A Linear Category via GoI and Realizability

A TSMC is a main ingredient of the notion of *GoI situation* in [7] (see §II-C). What are needed on top of it is a functor T for interpreting the ! modality, and a reflexive object U that would yield the carrier of an LCA. These can be provided to $\mathcal{K}\ell(\mathcal{Q})$ in the same way as to $\mathbf{Pfn} \cong \mathcal{K}\ell(\mathcal{L})$ and $\mathbf{Rel}_+ \cong \mathcal{K}\ell(\mathcal{P})$ done in [7]. See [14, Appendix B].

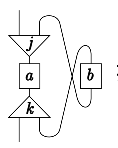
Theorem III.4. *The triple $\bigl(\mathcal{K}\ell(\mathcal{Q}),\, \mathbb{N}\cdot\_\,,\, \mathbb{N}\bigr)$ forms a GoI situation [7, Def. 4.1].* □

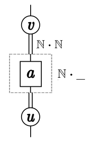
Therefore we can use [7, Prop. 4.2]—in which the Int-construction [9] plays an important role—to obtain an LCA.

Theorem III.5 (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). Its application operator $\cdot$ and its !-operator are concretely as follows.

$$a \cdot b \ := \ \mathsf{tr}^{\mathbb{N}}_{\mathbb{N},\mathbb{N}} \left(\begin{array}{l} \mathbb{N}+\mathbb{N} \xrightarrow{j} \mathbb{N} \xrightarrow{a} \\ \mathbb{N} \xrightarrow{k} \mathbb{N}+\mathbb{N} \\ \xrightarrow{\mathbb{N}+b} \mathbb{N}+\mathbb{N} \end{array} \right) =$$  ;

$$!\,a \ := \ \left(\begin{array}{l} \mathbb{N} \xrightarrow{v} \mathbb{N}\cdot\mathbb{N} \xrightarrow{\mathbb{N}\cdot a} \mathbb{N}\cdot\mathbb{N} \\ \xrightarrow{u} \mathbb{N} \end{array} \right) =$$  .

Here $j : \mathbb{N}+\mathbb{N} \cong \mathbb{N} : k$ and $u : \mathbb{N}\cdot\mathbb{N} \cong \mathbb{N} : v$ are (choices of) isomorphisms in $\mathbf{Sets}$ *embedded in $\mathcal{K}\ell(\mathcal{Q})$, like in [7, §5.1]. The above are string diagrams in the TSMC $\mathcal{K}\ell(\mathcal{Q})$.*

The LCA combinators

$$\begin{array}{lll} \mathsf{B}xyz = x(yz) & \mathsf{C}xyz = (xz)y & \mathsf{I}x = x \\ \mathsf{K}x\,!\,y = x & \mathsf{W}x\,!\,y = x\,!\,y\,!\,y & \mathsf{D}\,!\,x = x \\ \delta\,!\,x = !\,!\,x & \mathsf{F}\,!\,x\,!\,y = !(xy) & \end{array}$$

are defined in the same way as in [7, §4]. In fact, $A_{\mathcal{Q}}$ is affine*: it has the full* K *combinator s.t. $\mathsf{K}xy = x$.* □

Further, through the standard realizability technique we obtain a model for *typed* calculi. See e.g. [8, §2.1].

Definition III.6 (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 $|X| := \{x \mid (x,x) \in R\} = \{x \mid \exists y.\,(x,y) \in R\}$. Hence when restricted to its domain, X is an equivalence relation: therefore X can be thought of as a subset $|X| \subseteq A_{\mathcal{Q}}$ quotiented.

PERs over $A_{\mathcal{Q}}$ form a category $\mathbf{PER}_{\mathcal{Q}}$. Its arrow $X \to Y$ is defined to be an equivalence class of the PER

$$X \multimap Y := \Bigl\{ \, (c, c') \mid (x, x') \in X \Rightarrow (cx, c'x') \in Y \, \Bigr\} \ . \tag{6}$$

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."

Theorem III.7. *The category $\mathbf{PER}_{\mathcal{Q}}$ is a linear category [18], [19], 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* $\mathrm{der} : !\,X \to X$, $\delta : !\,X \to !\,!\,X$, $\varphi : !\,X \boxtimes !\,Y \to !(X \boxtimes Y)$ *and* $\varphi' : \mathrm{I} \to !\,\mathrm{I}$*) that is further equipped with monoidal natural transformations* $\mathrm{weak} : !\,X \to \mathrm{I}$ *and* $\mathrm{con} : !\,X \to !\,X \boxtimes !\,X$*, subject to certain conditions.*

Proof. By [8, Thm. 2.1]. See also Lem. V.2 later. □

We use $\boxtimes$ for the tensor product in $\mathbf{PER}_{\mathcal{Q}}$; it is distinguished from the tensor product of quantum states which we denote by $\otimes$. We will discuss this issue shortly in Rem. IV.1.

IV. A Quantum λ-Calculus $\mathbf{q}\lambda_\ell$

We introduce a new quantum λ-calculus $\mathbf{q}\lambda_\ell$. The subscript ℓ stands for "local."

Remark IV.1. The design of $\mathbf{q}\lambda_\ell$ is based on Selinger and Valiron's in [3]. One big difference is as follows. In [3], the type constructor $\otimes$ in linear λ-calculus ("multiplicative and") also plays a role of the tensor product of quantum states. Therefore the type $\mathtt{qbit} \otimes \mathtt{qbit}$ represents 2-qubit systems. This leads to clean syntax; and their ingenious operational semantics allows such double usage of $\otimes$.

Unfortunately, this design is not suited for the current style of semantics. While its reason is best observed on the technical level, a rough intuition is as follows. In our particle-style GoI semantics, a computation is like a game played by passing a single token (i.e. a quantum state) around. Now consider the combination $f \otimes g : A \otimes B \to C \otimes D$ of two computations f and g. It is the two games f and g played at the same time; for the sake of compositionality of the semantics the two games must be played separately without affecting each other. This separation is violated by the following program, valid in [3]:

$$\dfrac{\vdash \mathtt{EPR} : \mathtt{qbit} \otimes \mathtt{qbit} \qquad x : \mathtt{qbit} \vdash \mathtt{meas}\, x : \mathtt{bit} \qquad y : \mathtt{qbit} \vdash \mathtt{meas}\, y : \mathtt{bit}}{\vdash \mathtt{let}\,\langle x, y\rangle = \mathtt{EPR}\,\mathtt{in}\,\langle \mathtt{meas}\, x, \mathtt{meas}\, y\rangle : \mathtt{bit} \otimes \mathtt{bit}}$$

where $\mathtt{EPR}$ is the EPR pair $(|00\rangle + |11\rangle)/\sqrt{2}$. The result of $\mathtt{meas}\, x : \mathtt{qbit} \multimap \mathtt{bit}$ *does* affect that of $\mathtt{meas}\, y$.

Our design choice is hence to separate $\otimes$ for quantum states from the type constructor $\boxtimes$ for linear logic. In fact $\otimes$ 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*.

Definition IV.2 (The calculus $\mathbf{q}\lambda_\ell$)**.** The *types* of $\mathbf{q}\lambda_\ell$ are:

$$A, B ::= n\text{-}\mathtt{qbit} \mid {!\,A} \mid A \multimap B \mid \top \mid A \boxtimes B \mid A + B \ ,$$
$$\text{with conventions } \mathtt{qbit} := 1\text{-}\mathtt{qbit} \text{ and } \mathtt{bit} := \top + \top \ .$$

The *terms* of $\mathbf{q}\lambda_\ell$ are:

$$\begin{aligned}
&M, N, P ::= \\
&\quad x \mid \lambda x^A.M \mid MN \mid \langle M, N\rangle \mid * \mid \\
&\quad \mathtt{let}\,\langle x^A, y^B\rangle = M\,\mathtt{in}\,N \mid \mathtt{let}\,* = M\,\mathtt{in}\,N \mid \\
&\quad \mathtt{inj}_\ell^B\, M \mid \mathtt{inj}_r^A\, M \mid \\
&\quad \mathtt{match}\,P\,\mathtt{with}\,(x^A \mapsto M \mid y^B \mapsto N) \mid \\
&\quad \mathtt{letrec}\,f^A x = M\,\mathtt{in}\,N \mid \\
&\quad \mathtt{new}\,|0\rangle \mid \mathtt{meas}_i^{n+1} \mid U \mid \mathtt{cmp}_{m,n}\ , \\
&\quad \text{with conventions } \mathtt{tt} := \mathtt{inj}_\ell^\top(*) \text{ and } \mathtt{ff} := \mathtt{inj}_r^\top(*)\ .
\end{aligned}$$

Here $m, n \in \mathbb{N}$, $i \in [1, n+1]$; U is a $2^k \times 2^k$ unitary matrix, for some $k \in \mathbb{N}$; and A and B are types. The terms are almost the same as in [3]; the additional *composition* operator $\mathtt{cmp}$ will have the type m-$\mathtt{qbit} \boxtimes n$-$\mathtt{qbit} \multimap (m+n)$-$\mathtt{qbit}$, embedding nonentangled states as possibly entangled states.

For typing, we employ the same subtype relation $<:$ as in [3] for taking care of the $!$ modality. The rules that derive $<:$ are as in Table I. Here $(*)$ stands for the side condition $(n = 0 \Rightarrow m = 0)$.

The typing rules are as in Table I. There Δ, Γ, etc. denote (unordered) *contexts*. For a context $\Delta = (x_1 : A_1, \dots, x_m : A_m)$, $!\,\Delta$ denotes $(x_1 : {!\,A_1}, \dots, x_m : {!\,A_m})$. The side condition $(\dagger)$ stands for: an entry $x : D$ in the context Γ, Γ_1 or Γ_2 never has a type D of the form $D = {!\,E}$. That is, such a type $!\,E$ occurs only in $!\,\Delta$. In the rule (Ax.2), c is a constant and its *default type* A_c is defined as in Table I. 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.

Besides Rem. IV.1, among the design choices made for $\mathbf{q}\lambda_\ell$ are: 1) bound variables and injections have explicit type labels; 2) weakening is only allowed for types of the form $!\,A$. One reason for these choices is so that Lem. V.21 holds.

V. A Denotational Model

A. *Type Constructors in* $\mathbf{PER}_Q$

In what follows, an element of the LCA A_Q is often designated by an untyped linear λ-term. This is allowed due to combinatory completeness (see e.g. [8]).

Definition V.1. We introduce these combinators in A_Q.

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

Recall that the full K combinator is available in A_Q. Obviously: $\mathsf{P}xy = \mathsf{P}x'y'$ implies $x = x'$ and $y = y'$.

Lemma V.2. *The category* $\mathbf{PER}_Q$ *is a symmetric monoidal closed category with the following operations.*

$$\begin{aligned}
&X \boxtimes Y := \big\{ (\mathsf{P}xy, \mathsf{P}x'y') \,\big|\, (x, x') \in X \wedge (y, y') \in Y \big\}\ , \\
&\mathrm{I} := \{(\mathsf{I}, \mathsf{I})\}\ , \qquad X \multimap Y := (\textit{see (6)}).
\end{aligned}$$

Moreover, the monoidal unit I *is final (i.e. terminal) in* $\mathbf{PER}_Q$*.*

The category has a linear exponential comonad $!$ *[8]:*

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

where the latter $!$*'s are from Thm. III.5. In particular, its comonad structure is realized by the combinators* D *and* δ*.*

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

$$\begin{aligned}
X \times Y := \big\{ &(\mathsf{P}k_1(\mathsf{P}k_2u), \mathsf{P}k_1'(\mathsf{P}k_2'u')) \,\big| \\
&(k_1u, k_1'u') \in X \wedge (k_2u, k_2'u') \in Y \big\}\ , \\
X + Y := \big\{ &(\mathsf{PK}x, \mathsf{PK}x') \,\big|\, (x, x') \in X \big\} \\
\cup \big\{ &(\mathsf{P}\bar{\mathsf{K}}y, \mathsf{P}\bar{\mathsf{K}}y') \,\big|\, (y, y') \in Y \big\}\ .
\end{aligned}$$

Proof. Straightforward; see e.g. [8], [20]. $\square$

Logically, $\boxtimes$ is "multiplicative and"; $\times$ is "additive and."

Lemma V.3. *In* $\mathbf{PER}_Q$ *we have canonical isomorphisms*

$$!(X + Y) \cong {!\,X} + {!\,Y}\ ,\ !!\,X \cong {!\,X}\ ,\ !(X \boxtimes Y) \cong {!\,X} \boxtimes {!\,Y}\ ;$$

$$\dfrac{(*)}{!^n\, k\text{-}\mathtt{qbit} <: !^m\, k\text{-}\mathtt{qbit}}\ (k\text{-}\mathtt{qbit}) \qquad \dfrac{(*)}{!^n \top <: !^m \top}\ (\top)$$

$$\dfrac{A_1 <: B_1 \quad A_2 <: B_2 \quad (*)}{!^n(A_1 \boxdot A_2) <: !^m(B_1 \boxdot B_2)}\ (\boxdot) \text{ with } \boxdot \in \{\boxtimes, +\}$$

$$\dfrac{B_1 <: A_1 \quad A_2 <: B_2 \quad (*)}{!^n(A_1 \multimap A_2) <: !^m(B_1 \multimap B_2)}\ (\multimap)$$

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

$$\dfrac{\Delta \vdash M : !^n\, A}{\Delta \vdash \mathtt{inj}^B_\ell M : !^n(A+B)}\ (+.\text{I}_1)$$

$$\dfrac{\Delta \vdash N : !^n\, B}{\Delta \vdash \mathtt{inj}^A_r N : !^n(A+B)}\ (+.\text{I}_2)$$

$$\dfrac{!\,\Delta, \Gamma_1 \vdash P : !^n(A+B) \quad \begin{array}{c} !\,\Delta, \Gamma_2, x : !^n\, A \vdash M : C \\ !\,\Delta, \Gamma_2, y : !^n\, B \vdash N : C \end{array}}{\begin{array}{l} !\,\Delta, \Gamma_1, \Gamma_2 \\ \qquad \vdash \mathtt{match}\, P\, \mathtt{with}\, (x^{!^n\, A} \mapsto M \mid y^{!^n\, B} \mapsto N) : C \end{array}}\ (+.\text{E}), (\dagger)$$

$$\dfrac{x : A, \Delta \vdash M : B}{\Delta \vdash \lambda x^A.M : A \multimap B}\ (\multimap.\text{I}_1)$$

$$\dfrac{x : A, !\,\Delta \vdash M : B}{!\,\Delta \vdash \lambda x^A.M : !^n(A \multimap B)}\ (\multimap.\text{I}_2)$$

$$\dfrac{!\,\Delta, \Gamma_1 \vdash M : A \multimap B \quad !\,\Delta, \Gamma_2 \vdash N : A}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash MN : B}\ (\multimap.\text{E}), (\dagger)$$

$$\dfrac{!\,\Delta, \Gamma_1 \vdash M_1 : !^n\, A_1 \quad !\,\Delta, \Gamma_2 \vdash M_2 : !^n\, A_2}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash \langle M_1, M_2 \rangle : !^n(A_1 \boxtimes A_2)}\ (\boxtimes.\text{I}), (\dagger)$$

$$\dfrac{}{!\,\Delta \vdash * : !^n \top}\ (\top.\text{I})$$

$$\dfrac{\begin{array}{c} !\,\Delta, \Gamma_2, x_1 : !^n\, A_1, x_2 : !^n\, A_2 \vdash N : A \\ !\,\Delta, \Gamma_1 \vdash M : !^n(A_1 \boxtimes A_2) \end{array}}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash \mathtt{let}\, \langle x_1^{!^n\, A_1}, x_2^{!^n\, A_2} \rangle = M\, \mathtt{in}\, N : A}\ (\boxtimes.\text{E}), (\dagger)$$

$$\dfrac{!\,\Delta, \Gamma_1 \vdash M : \top \quad !\,\Delta, \Gamma_2 \vdash N : A}{!\,\Delta, \Gamma_1, \Gamma_2 \vdash \mathtt{let} * = M\, \mathtt{in}\, N : A}\ (\top.\text{E}), (\dagger)$$

$$\dfrac{\begin{array}{c} !\,\Delta, \Gamma, f : !(A \multimap B) \vdash N : C \\ !\,\Delta, f : !(A \multimap B), x : A \vdash M : B \end{array}}{!\,\Delta, \Gamma \vdash \mathtt{letrec}\, f^{A \multimap B} x = M\, \mathtt{in}\, N : C}\ (\text{rec}), (\dagger)$$

$$\begin{array}{rcl} A_{\mathtt{new}|0\rangle} & := & \mathtt{qbit} \\ A_{\mathtt{meas}^{n+1}_i} & := & (n+1)\text{-}\mathtt{qbit} \multimap (\mathtt{bit} \boxtimes n\text{-}\mathtt{qbit}) \text{ for } n \geq 1 \\ A_{\mathtt{meas}^1_1} & := & \mathtt{qbit} \multimap \mathtt{bit} \\ A_U & := & n\text{-}\mathtt{qbit} \multimap n\text{-}\mathtt{qbit} \quad \text{for a } 2^n \times 2^n \text{ matrix } U \\ A_{\mathtt{cmp}_{m,n}} & := & (m\text{-}\mathtt{qbit} \boxtimes n\text{-}\mathtt{qbit}) \multimap (m+n)\text{-}\mathtt{qbit} \end{array}$$

Table I
TYPING RULES FOR $\mathbf{q}\lambda_\ell$

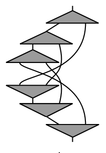
therefore ! on $\mathbf{PER}_\mathcal{Q}$ is idempotent and strong monoidal; it also preserves coproducts. Additionally, as in any linear category:

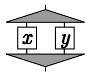
$$\begin{array}{ll} !(X \times Y) \cong !\,X \boxtimes !\,Y \ , & (X+Y) \boxtimes Z \cong X \boxtimes Z + Y \boxtimes Z \ , \\ \mathrm{I} \multimap X \cong X \ , & (X+Y) \multimap Z \cong (X \multimap Z) \times (Y \multimap Z) \ . \end{array}$$

Proof. See [14, Appendix D]. □

Using another pairing combinator $\dot{\mathsf{P}}$ we obtain a different "implementation" $\dot{\times}$ of products. The merit of $\dot{\times}$ is that it exhibits better order-theoretic properties; we will need them for recursion. In contrast, P enjoys a useful combinatorial property: $(\mathsf{P}xy)z = zxy$.

Definition V.4 (Combinator $\dot{\mathsf{P}}$, binary product $X \dot{\times} Y$)**.** We define $\dot{\mathsf{P}} \in A_\mathcal{Q}$ by the string diagram in $\mathcal{K}\ell(\mathcal{Q})$ shown top on the right. The triangles are $j : \mathbb{N} + \mathbb{N} \cong \mathbb{N} : k$ in Thm. III.5. Then $\dot{\mathsf{P}}xy$ becomes as shown bottom on the right. It is straightforward to write projections $\dot{\mathsf{P}}_\mathsf{l}, \dot{\mathsf{P}}_\mathsf{r}$ for $\dot{\mathsf{P}}$; also conversion combinators $\mathsf{C}_{\mathsf{P} \mapsto \dot{\mathsf{P}}}$—with $\mathsf{C}_{\mathsf{P} \mapsto \dot{\mathsf{P}}}(\mathsf{P}xy) = \dot{\mathsf{P}}xy$—and $\mathsf{C}_{\dot{\mathsf{P}} \mapsto \mathsf{P}}$. We define $X \dot{\times} Y$ by replacing P by $\dot{\mathsf{P}}$ in $X \times Y$ (Lem. V.2).

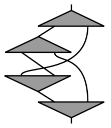




Lemma V.5. *We have a canonical natural isomorphism $X \times Y \overset{\cong}{\Rightarrow} X \dot{\times} Y$ in $\mathbf{PER}_\mathcal{Q}$, realized using $\mathsf{C}_{\mathsf{P} \mapsto \dot{\mathsf{P}}}$. In what follows we shall use $\times$ and $\dot{\times}$ interchangeably. That is, we suppress use of $\mathsf{C}_{\mathsf{P} \mapsto \dot{\mathsf{P}}}$ and $\mathsf{C}_{\dot{\mathsf{P}} \mapsto \mathsf{P}}$.* □

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

Definition V.6 (Combinator A)**.** We define $\mathsf{A} \in A_\mathcal{Q}$ by the string diagram in $\mathcal{K}\ell(\mathcal{Q})$ shown on the right. The triangles are $j : \mathbb{N} + \mathbb{N} \cong \mathbb{N} : k$ in Thm. III.5. It satisfies: $\mathsf{A}xy = x \odot y$, where $\odot$ denotes composition of arrows in $\mathcal{K}\ell(Q)$.



Definition V.7 (Combinators Q_ρ, Q_U, $\mathsf{Q}^{N+1}_{|0_i\rangle}$, $\mathsf{Q}^{N+1}_{|1_i\rangle}$)**.** We define the elements $\mathsf{Q}_\rho, \mathsf{Q}_U, \mathsf{Q}^{N+1}_{|0_i\rangle} \in A_\mathcal{Q}$ as follows. Here $N \in \mathbb{N}$, $\rho \in D_N$, U is an $N \times N$ unitary matrix, and $i \in [1, N+1]$.

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

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

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

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

In the definition of Q_ρ, $\otimes$ denotes tensor product of matrices. Q_ρ is such that an incoming token σ comes out of the same pipe, with its state composed with ρ. In particular $1 \in D_1$ comes out as ρ. In $\mathsf{Q}^{N+1}_{|0_i\rangle}$, $\langle 0_i | \_ | 0_i \rangle$ denotes the projection operator from $D_{2^{N+1}}$ to D_{2^N} that projects the i-th qubit to $|0\rangle$. A combinator $\mathsf{Q}^{N+1}_{|1_i\rangle} \in A_\mathcal{Q}$ is defined in the same way, using $\langle 1_i | \_ | 1_i \rangle$.

Definition V.8 ($[\![N\text{-}\mathtt{qbit}]\!]$,$[\![\mathtt{bit}]\!]$)**.** For each $N \in \mathbb{N}$ we define a PER $[\![N\text{-}\mathtt{qbit}]\!]$ by:

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

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

A PER $[\![\mathtt{bit}]\!]$ is defined to be $\mathrm{I}+\mathrm{I}$ (see Lem. V.2).

Lemma V.9 (Combinators $\mathsf{U}_U, \mathsf{Pr}^{N+1}_{|0_i\rangle}, \mathsf{Pr}^{N+1}_{|1_i\rangle}$)**.** *We define*

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

Then we have, for ρ, σ, U of suitable dimensions,

$$\begin{array}{ll} \mathsf{AQ}_\rho \mathsf{Q}_\sigma = \mathsf{Q}_{\rho\otimes\sigma} \ , & \mathsf{U}_U \mathsf{Q}_\rho = \mathsf{Q}_{U\rho U^\dagger} \ , \\ \mathsf{Pr}^{N+1}_{|0_i\rangle}\mathsf{Q}_\sigma = \mathsf{Q}_{\langle 0_i|\sigma|0_i\rangle} \ , & \mathsf{Pr}^{N+1}_{|1_i\rangle}\mathsf{Q}_\sigma = \mathsf{Q}_{\langle 1_i|\sigma|1_i\rangle} \ . \end{array}$$

C. Continuation Monad T

Our categorical model $\mathbf{PER}_\mathcal{Q}$ further employs a monad T; the interpretation of a type judgment $\Delta \vdash M : A$ will be an arrow $[\![\Delta]\!] \to T[\![A]\!]$. It 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)*. Informally, the reason for this design choice 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 we need 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 §II-B) would not do. One explanation is as follows. Think of $\mathtt{meas}^2_1 : 2\text{-}\mathtt{qbit} \multimap \mathtt{bit} \boxtimes \mathtt{qbit}$: 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 $\mathsf{tr}\big[\,(\langle 0_1|\_|0_1\rangle \otimes \mathcal{I}_2)\rho\big]$; use of a naive "probability distribution" monad requires calculation of this probability. The calculation traces out the second qubit, destroying it and leaving it inept for further quantum procedures. Another explanation is: naive interpretation of $\mathtt{meas}^2_1$ has the codomain $\mathtt{bit} \otimes \mathtt{qbit}$—with entanglement—rather than the desired codomain $\mathtt{bit} \boxtimes \mathtt{qbit}$.

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. For a result type R, we take that of complete binary trees with each edge labeled by a real number $p \in [0,1]$—obtained as a final coalgebra.

Lemma V.10 (The result type R)**.** *The endofunctor $F = [\![\mathtt{bit}]\!] \multimap ([\![0\text{-}\mathtt{qbit}]\!] \dot{\times} \_)$ on $\mathbf{PER}_\mathcal{Q}$ has a final coalgebra. We denote it by $r : R \xrightarrow{\cong} FR$.*

Proof. See [14, Appendix E]. □

It is standard that $T := (\_ \multimap R) \multimap R$ is a strong monad. We will also need the following map.

Definition V.11 (mult)**.** We define a map $\mathrm{mult} : [\![0\text{-}\mathtt{qbit}]\!] \boxtimes R \to R$ in $\mathbf{PER}_\mathcal{Q}$ to be such that: it receives $p \in [0,1]$ and a tree t, and returns the tree t with each label in it multiplied by p. We can implement such a function by writing a coalgebra c whose carrier is $[\![0\text{-}\mathtt{qbit}]\!] \boxtimes R$; see [14, Appendix E].

D. Fixed Point Operator

We shall interpret recursion in $\mathbf{q}\lambda_\ell$ using the ω-CPO structure of $A_\mathcal{Q}$; this is like in [23]. The proofs for §V-D are found in [14, Appendix E].

Lemma V.12 ($A_\mathcal{Q}$ is an ω-CPO)**.** *The set $A_\mathcal{Q}$ is an ω-CPO with $\bot$, by the ω-$\mathbf{CPO}$ enriched structure $\sqsubseteq$ of $\mathcal{K}\ell(\mathcal{Q})$ (Thm. III.3). Furthermore: 1) application $\cdot : A^2_\mathcal{Q} \to A_\mathcal{Q}$ and $!$ are continuous; and 2) $\bot \cdot a = \bot$.* □

Definition V.13 (Admissible PER)**.** A PER $U \in \mathbf{PER}_\mathcal{Q}$ is said to be *admissible* if: 1) $(\bot,\bot) \in U$ for the least element $\bot \in A_\mathcal{Q}$ 2) $(x_i, y_i) \in U$, $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$ and $y_0 \sqsubseteq y_1 \sqsubseteq \cdots$ implies $(\sup_i x_i, \sup_i y_i) \in U$.

Lemma V.14. *For each X, Y, $X \multimap TY$ is admissible.* □

Definition V.15 (Fixed point operator)**.** Let $U, X \in \mathbf{PER}_\mathcal{Q}$, and $f : !\,U \boxtimes !\,X \to U$ be an arrow. Further assume that U is admissible. We define f's *fixed point* $\mathrm{fix}(f) : !\,X \to U$ as follows. Let c be a code of f. We define $c_0, c_1, \ldots \in |\,!\,X \multimap U|$ by

$$c_0 := \bot \ ; \qquad c_{n+1} := \text{ the canonical code of}$$
$$!\,X \xrightarrow{\mathrm{con}} !\,X \boxtimes !\,X \xrightarrow{\delta\boxtimes\mathrm{id}} !\,!\,X \boxtimes !\,X \xrightarrow{![c_n]\boxtimes\mathrm{id}} !\,U \boxtimes !\,X \xrightarrow{[c]} U \ .$$

Since U is admissible and $\bot \cdot x = \bot$, $c_0 = \bot$ is a valid code. By induction we can show that $c_0 \sqsubseteq c_1 \sqsubseteq \cdots$; since $!\,X \multimap U$ is admissible its supremum $\sup_i c_i$ belongs to the domain $|\,!\,X \multimap U|$. We define $\mathrm{fix}(f) := [\,\sup_i c_i\,]$.

E. Interpretation

Definition V.16 (Interpretation of types)**.** For each $\mathbf{q}\lambda_\ell$-type A, we assign $[\![A]\!] \in \mathbf{PER}_\mathcal{Q}$ as follows, using the constructors in Lem. V.2. For base types, $[\![N\text{-}\mathtt{qbit}]\!]$ is as in Def. V.8.

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

Definition V.17 (Interpretation of $<:$)**.** To each derivation of the subtype relation $\Pi \Vdash A <: B$ (Def. IV.2), we can by induction assign an arrow $[\![\Pi]\!] : [\![A]\!] \to [\![B]\!]$ in $\mathbf{PER}_\mathcal{Q}$ using der and δ (Thm. III.7).

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

Definition V.18 (Interpretation of constants)**.** For a constant c in $\mathbf{q}\lambda_\ell$, an arrow $[\![c]\!] : \mathrm{I} \to [\![A_c]\!]$ in $\mathbf{PER}_\mathcal{Q}$ is as follows.[4]

[4]Note that $[\![c]\!]$ is not $\mathrm{I} \to T[\![A_c]\!]$; this is because a constant c can always have the type $!\,A_c$. See (Ax.2) in Table I and its interpretation in Table II.

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

$$[\![(n+1)\text{-}\mathtt{qbit}]\!] \boxtimes (([\![\mathtt{bit}]\!] \boxtimes [\![n\text{-}\mathtt{qbit}]\!]) \multimap R) \xrightarrow{m} R \ . \tag{7}$$

This can be obtained from the following map, using R's fixed point property ($R \overset{\cong}{\Rightarrow} [\![\mathtt{bit}]\!] \multimap ([\![0\text{-}\mathtt{qbit}]\!] \times R)$) and Lem. V.3.

$$\begin{aligned} &\left(\begin{array}{l} [\![(n+1)\text{-}\mathtt{qbit}]\!] \boxtimes ([\![n\text{-}\mathtt{qbit}]\!] \multimap R)^{\times 2} \\ +[\![(n+1)\text{-}\mathtt{qbit}]\!] \boxtimes ([\![n\text{-}\mathtt{qbit}]\!] \multimap R)^{\times 2} \end{array} \right) \\ &\xrightarrow{\mathsf{Pr}^{n+1}_{|0_i\rangle} \boxtimes \pi_\ell + \mathsf{Pr}^{n+1}_{|1_i\rangle} \boxtimes \pi_r} \left(\begin{array}{l} [\![n\text{-}\mathtt{qbit}]\!] \boxtimes ([\![n\text{-}\mathtt{qbit}]\!] \multimap R) \\ +[\![n\text{-}\mathtt{qbit}]\!] \boxtimes ([\![n\text{-}\mathtt{qbit}]\!] \multimap R) \end{array} \right) \\ &\xrightarrow{\mathsf{ev}+\mathsf{ev}} R + R \xrightarrow{[\langle \mathsf{Q}_0, \mathrm{id}\rangle, \langle \mathsf{Q}_0, \mathrm{id}\rangle]} [\![0\text{-}\mathtt{qbit}]\!] \times R \ . \end{aligned}$$

On the last line Q_0 denotes $R \xrightarrow{\nabla} \mathrm{I} \xrightarrow{[\lambda x. x\mathsf{Q}_0]} [\![0\text{-}\mathtt{qbit}]\!]$, with ∇ denoting the unique arrow.

For $c = \mathtt{meas}_1^1$, by transpose we need

$$[\![\mathtt{qbit}]\!] \boxtimes ([\![\mathtt{bit}]\!] \multimap R) \xrightarrow{m'} R \ , \tag{8}$$

which we similarly obtain from the following composite.

$$\begin{aligned} &[\![\mathtt{qbit}]\!] \boxtimes R^{\times 2} + [\![\mathtt{qbit}]\!] \boxtimes R^{\times 2} \\ &\xrightarrow{\mathsf{Pr}^1_{|0\rangle} \boxtimes \pi_\ell + \mathsf{Pr}^1_{|1\rangle} \boxtimes \pi_r} [\![0\text{-}\mathtt{qbit}]\!] \boxtimes R + [\![0\text{-}\mathtt{qbit}]\!] \boxtimes R \\ &\xrightarrow{[\langle \mathsf{Q}_0, \mathrm{mult}\rangle, \langle \mathsf{Q}_0, \mathrm{mult}\rangle]} [\![0\text{-}\mathtt{qbit}]\!] \times R \ ; \end{aligned}$$

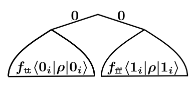
here Q_0 is the same as above; mult is from Def. V.11.

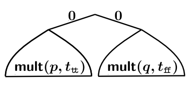
For the other constants (take transpose when necessary):

$$\begin{aligned} [\![\mathtt{new}|0\rangle]\!] &:= \quad \mathrm{I} \xrightarrow{[\lambda x. x\mathsf{Q}_{|0\rangle\langle 0|}]} [\![\mathtt{qbit}]\!] \\ [\![U]\!] &:= \quad [\![n\text{-}\mathtt{qbit}]\!] \xrightarrow{[\mathsf{U}_U]} [\![n\text{-}\mathtt{qbit}]\!] \xrightarrow{\eta} T[\![n\text{-}\mathtt{qbit}]\!] \\ [\![\mathtt{cmp}_{m,n}]\!] &:= \quad [\![m\text{-}\mathtt{qbit}]\!] \boxtimes [\![n\text{-}\mathtt{qbit}]\!] \xrightarrow{[\lambda w. w\mathsf{A}]} \\ &\qquad [\![(m+n)\text{-}\mathtt{qbit}]\!] \xrightarrow{\eta} T[\![(m+n)\text{-}\mathtt{qbit}]\!] \end{aligned}$$

Here we used Lem. V.2 and Def. V.7.

The idea for $[\![\mathtt{meas}_i^{n+1}]\!]$ is as follows. Let $n \geq 1$ and take the map m in (7); roughly its input is a triple $(\rho, f_{\mathtt{tt}}, f_{\mathtt{ff}})$ with $\rho \in D_{2^{n+1}}$ and $f_{\mathtt{tt}}, f_{\mathtt{ff}} : D_{2^n} \to R$. Then m's output is the tree shown below on the left. We simply put 0 as the labels on the depth one edges; 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.


0
0
$f_{\mathtt{tt}}\langle 0_i|\rho|0_i\rangle$
$f_{\mathtt{ff}}\langle 1_i|\rho|1_i\rangle$


0
0
$\mathbf{mult}(p, t_{\mathtt{tt}})$
$\mathbf{mult}(q, t_{\mathtt{ff}})$

When there is only one qubit left, we finally compute actual probabilities. Take m' in (8); its input is roughly a triple $(\rho, t_{\mathtt{tt}}, t_{\mathtt{ff}})$ with $\rho \in D_2$ and trees $t_{\mathtt{tt}}, t_{\mathtt{ff}} \in R$. Let $p = \langle 0|\rho|0\rangle$ and $q = \langle 1|\rho|1\rangle$ be probabilities; then what m' returns is the tree above on the right. Recall that $\mathrm{mult}(p, \_)$ multiplies all the labels of the input tree by p.

This way we only generate edges with its label 0. This is no problem: once we supply trees with nonzero labels as $t_{\mathtt{tt}}$ and $t_{\mathtt{ff}}$ above, we observe nonzero probabilities.

The rest of the definition is straightforward.

Definition V.19 (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 V.20 (Interpretation of type judgments)**.** For each derivation $\Pi \Vdash \Delta \vdash M : A$ of a type judgment in $\mathbf{q}\lambda_\ell$, we inductively assign an arrow $[\![\Pi]\!] : [\![\Delta]\!] \to T[\![A]\!]$ as in Table II. There some obvious elements are omitted: we write ∇ in place of $\nabla \boxtimes \mathrm{id}$, $[\![M]\!]$ 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 Def. V.15.

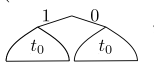
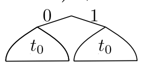
Lemma V.21. *The interpretation* $[\![\Pi \Vdash \Delta \vdash M : A]\!]$ *does not depend on the choice of* Π. *That is,* $[\![\Delta \vdash M : A]\!]$ *is well-defined if the judgment is derivable.*

Proof. See [14, Appendix F]. $\square$

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

Definition V.22 (Trees $t_{\mathtt{tt}}, t_{\mathtt{ff}}$, and test)**.** We define trees $t_0, t_{\mathtt{tt}}, t_{\mathtt{ff}} : \mathrm{I} \to R$ by:

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

$t_{\mathtt{tt}} :=$  , $t_{\mathtt{ff}} :=$  .

Indeed, it is straightforward to write down an F-coalgebra (with its carrier $3 \cdot \mathrm{I}$) that gives rise to such trees by finality.

We denote by test the arrow $\mathrm{I} \to ([\![\mathtt{bit}]\!] \multimap R)$ such that: $\mathtt{tt} \mapsto t_{\mathtt{tt}}$ and $\mathtt{ff} \mapsto t_{\mathtt{ff}}$.

Definition V.23 (Operation prob on trees)**.** For each arrow $t : \mathrm{I} \to R$ thought of as a tree, we define $\mathrm{prob}(t) \in \mathbb{R}^2$ by:

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

To be precise, "edges going down-left" means "obtained by supplying $b_0, b_1, \ldots, b_n$, and finally $\mathtt{tt}$." For example, $\mathrm{prob}(t_{\mathtt{tt}}) = (1, 0)$ and $\mathrm{prob}(t_{\mathtt{ff}}) = (0, 1)$.

Definition V.24 (Denotation relation $\curlyvee$)**.** We define a relation $\curlyvee$ 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:

$$\mathrm{I} \xrightarrow{\cong} \mathrm{I} \boxtimes \mathrm{I} \xrightarrow{\mathrm{test} \boxtimes [\![\vdash M : \mathtt{bit}]\!]} ([\![\mathtt{bit}]\!] \multimap R) \boxtimes T[\![\mathtt{bit}]\!] \xrightarrow{\mathsf{ev}} R \ . \tag{9}$$

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

$$\boxed{\text{Ax.1}}\quad \llbracket !\,\Delta\rrbracket \boxtimes \llbracket A\rrbracket \xrightarrow{\nabla} \llbracket A\rrbracket \xrightarrow{\llbracket A<:A'\rrbracket} \llbracket A'\rrbracket \xrightarrow{\eta} T\llbracket A'\rrbracket$$

$$\boxed{\text{Ax.2}}\quad \llbracket !\,\Delta\rrbracket \xrightarrow{\nabla} \mathrm{I} \xrightarrow{\varphi'} !\,\mathrm{I} \xrightarrow{!\llbracket c\rrbracket\ \text{(cf. Def. V.18)}} !\llbracket A_c\rrbracket \xrightarrow{\eta} T\,!\llbracket A_c\rrbracket \xrightarrow{T\llbracket !\,A_c<:A\rrbracket} T\llbracket A\rrbracket$$

$$\boxed{+.\mathrm{I}_1}\quad \llbracket \Delta\rrbracket \xrightarrow{\llbracket M\rrbracket} T(!^n\llbracket A\rrbracket) \xrightarrow{T\,!^n\,\kappa_\ell} T\,!^n(\llbracket A\rrbracket + \llbracket B\rrbracket)$$

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

$$\boxed{+.\mathrm{E}}\quad \llbracket !\,\Delta\rrbracket \boxtimes \llbracket \Gamma_1\rrbracket \boxtimes \llbracket \Gamma_2\rrbracket \xrightarrow{\mathrm{con},\llbracket P\rrbracket} T(!^n(\llbracket A\rrbracket + \llbracket B\rrbracket)) \boxtimes \llbracket !\,\Delta\rrbracket \boxtimes \llbracket \Gamma_2\rrbracket \xrightarrow{\mathrm{str},\text{Lem. V.3}} T(!^n\llbracket A\rrbracket \boxtimes \llbracket !\,\Delta\rrbracket \boxtimes \llbracket \Gamma_2\rrbracket + !^n\llbracket B\rrbracket \boxtimes \llbracket !\,\Delta\rrbracket \boxtimes \llbracket \Gamma_2\rrbracket) \xrightarrow{T[\llbracket M\rrbracket,\llbracket N\rrbracket]} TTC \xrightarrow{\mu} TC$$

$$\boxed{\multimap.\mathrm{I}_1}\quad \llbracket \Delta\rrbracket \xrightarrow{\llbracket M\rrbracket^\wedge} \llbracket A \multimap B\rrbracket \xrightarrow{\eta} T\llbracket A \multimap B\rrbracket$$

$$\boxed{\multimap.\mathrm{I}_2}\quad \llbracket !\,\Delta\rrbracket \xrightarrow{\delta} \llbracket !^{n+1}\,\Delta\rrbracket \xrightarrow{!^n(\llbracket M\rrbracket^\wedge)} \llbracket !^n(A \multimap B)\rrbracket \xrightarrow{\eta} T\llbracket !^n(A \multimap B)\rrbracket$$

$$\boxed{\multimap.\mathrm{E}}\quad \llbracket !\,\Delta\rrbracket \boxtimes \llbracket \Gamma_1\rrbracket \boxtimes \llbracket \Gamma_2\rrbracket \xrightarrow{\mathrm{con},\llbracket M\rrbracket,\llbracket N\rrbracket} T(\llbracket A\rrbracket \multimap T\llbracket B\rrbracket) \boxtimes T\llbracket A\rrbracket \xrightarrow{\mathrm{str}'} T((\llbracket A\rrbracket \multimap T\llbracket B\rrbracket) \boxtimes T\llbracket A\rrbracket) \xrightarrow{T\mathrm{str}} TT((\llbracket A\rrbracket \multimap T\llbracket B\rrbracket) \boxtimes \llbracket A\rrbracket) \xrightarrow{\mathsf{ev},\mu} T\llbracket B\rrbracket$$

$$\boxed{\boxtimes.\mathrm{I}}\quad \llbracket !\,\Delta\rrbracket \boxtimes \llbracket \Gamma_1\rrbracket \boxtimes \llbracket \Gamma_2\rrbracket \xrightarrow{\mathrm{con},\llbracket M_1\rrbracket,\llbracket M_2\rrbracket} T\,!^n\llbracket A_1\rrbracket \boxtimes T\,!^n\llbracket A_2\rrbracket \xrightarrow{\mathrm{str}',\ \text{and then}\ \mathrm{str},\mu} T(!^n\llbracket A_1\rrbracket \boxtimes !^n\llbracket A_2\rrbracket) \xrightarrow{\text{Lem. V.3},\mu} T\,!^n(\llbracket A_1\rrbracket \boxtimes \llbracket A_2\rrbracket)$$

$$\boxed{\top.\mathrm{I}}\quad \llbracket !\,\Delta\rrbracket \xrightarrow{\nabla} \mathrm{I} \xrightarrow{\varphi'} !\,\mathrm{I} \xrightarrow{\delta,\mathrm{der}} !^n\,\mathrm{I} \xrightarrow{\eta} T\,!^n\,\mathrm{I}$$

$$\boxed{\boxtimes.\mathrm{E}}\quad \llbracket !\,\Delta\rrbracket \boxtimes \llbracket \Gamma_1\rrbracket \boxtimes \llbracket \Gamma_2\rrbracket \xrightarrow{\mathrm{con},\llbracket M\rrbracket} T\,!^n(\llbracket A_1\rrbracket \boxtimes \llbracket A_2\rrbracket) \boxtimes \llbracket !\,\Delta\rrbracket \boxtimes \llbracket \Gamma_2\rrbracket \xrightarrow{\text{Lem. V.3},\mathrm{str}'} T(!^n\llbracket A_1\rrbracket \boxtimes !^n\llbracket A_2\rrbracket \boxtimes \llbracket !\,\Delta\rrbracket \boxtimes \llbracket \Gamma_2\rrbracket) \xrightarrow{\llbracket N\rrbracket,\mu} T\llbracket A\rrbracket$$

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

$$\boxed{\text{rec}}\quad \llbracket !\,\Delta\rrbracket \boxtimes \llbracket \Gamma\rrbracket \xrightarrow{\mathrm{con},\delta} !\llbracket !\,\Delta\rrbracket \boxtimes \llbracket !\,\Delta\rrbracket \boxtimes \llbracket \Gamma\rrbracket \xrightarrow{!\,\mathrm{fix}(\llbracket M\rrbracket^\wedge)} !\llbracket A \multimap B\rrbracket \boxtimes \llbracket !\,\Delta\rrbracket \boxtimes \llbracket \Gamma\rrbracket \xrightarrow{\llbracket N\rrbracket} TC\ ,\ \text{where}$$
$$\llbracket !\,\Delta\rrbracket \boxtimes !(\llbracket A\rrbracket \multimap T\llbracket B\rrbracket) \xrightarrow{\llbracket M\rrbracket^\wedge} (\llbracket A\rrbracket \multimap T\llbracket B\rrbracket)$$

Table II
INTERPRETATION OF TYPE JUDGMENTS

VI. Operational Semantics and Adequacy

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

Definition VI.1 (Extended $\mathbf{q}\lambda_\ell$)**.** For operational semantics, we extend $\mathbf{q}\lambda_\ell$-terms by additional two sets of constants:

$$\begin{array}{ll} \mathtt{new}\,\rho & \text{for each } n \in \mathbb{N} \text{ and } \rho \in D_{2^n}; \\ \mathtt{abort}_{A'} & \text{for each type } A'. \end{array}$$

Their default types are: $A_{\mathtt{new}\,\rho} := n\text{-}\mathtt{qbit}$ for $\rho \in D_{2^n}$; $A_{\mathtt{abort}_{A'}} := A'$. The interpretation $\llbracket \mathtt{new}\,\rho\rrbracket$ is obvious (cf. Def. V.18); that of $\mathtt{abort}_A$ is $[\bot] : \mathrm{I} \to T\llbracket A\rrbracket$.

We also introduce the following shorthands for "`letrec` with counters." They do not contain actual `letrec`.

$$\begin{array}{l} \mathtt{letrec}^0\, f^{A\multimap B}x = M\,\mathtt{in}\,N \quad := \quad N[\mathtt{abort}_{A\multimap B}/f]\ ; \\ \mathtt{letrec}^{n+1}\, f^{A\multimap B}x = M\,\mathtt{in}\,N \quad := \\ \qquad\qquad N[\,\lambda x^A.\mathtt{letrec}^n\, f^{A\multimap B}x = M\,\mathtt{in}\,M/f\,]\ . \end{array}$$

Definition VI.2 (Value, evaluation context)**.** The *values* and *evaluation contexts* of $\mathbf{q}\lambda_\ell$ are defined in a standard way.

$$\begin{array}{l} V, V_1, V_2\ ::=\ x \mid \lambda x^A.M \mid \langle V_1, V_2\rangle \mid * \mid \\ \quad \mathtt{inj}^B_\ell\, V \mid \mathtt{inj}^A_r\, V \mid \mathtt{new}\,\rho \mid \mathtt{meas}^{n+1}_i \mid U \mid \mathtt{cmp}_{m,n}\ ; \\ E\ ::=\ [\_] \mid E[\,[\_]M\,] \mid E[\,V[\_]\,] \mid E[\,\langle [\_], M\rangle\,] \mid \\ \quad E[\,\langle V, [\_]\rangle\,] \mid E[\,\mathtt{let}\,\langle x^A, y^B\rangle = [\_]\,\mathtt{in}\,M\,] \mid \\ \quad E[\,\mathtt{let}\,* = [\_]\,\mathtt{in}\,N\,] \mid E[\,\mathtt{inj}^B_\ell[\_]\,] \mid E[\,\mathtt{inj}^A_r[\_]\,] \mid \\ \quad E[\,\mathtt{match}\,[\_]\,\mathtt{with}\,(x^A \mapsto M \mid y^B \mapsto N)\,] \end{array}$$

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

Definition VI.3 (Small-step semantics)**.** The *reduction rules* of $\mathbf{q}\lambda_\ell$ are defined in a standard way. Each reduction is labeled by a real number from $[0, 1]$.

$$\begin{array}{l} E[\,(\lambda x^A.M)V\,] \to_1 E[\,M[V/x]\,] \\ E[\,\mathtt{let}\,\langle x^A, y^B\rangle = \langle V, W\rangle\,\mathtt{in}\,M\,] \to_1 E[\,M[V/x, W/y]\,] \\ E[\,\mathtt{let}\,* = *\,\mathtt{in}\,M\,] \to_1 E[\,M\,] \\ E[\,\mathtt{match}\,(\mathtt{inj}^B_\ell V)\,\mathtt{with}\,(x^{!^n\,A} \mapsto M \mid y^{!^n\,B} \mapsto N)\,] \\ \hfill \to_1 E[\,M[V/x]\,] \\ E[\,\mathtt{match}\,(\mathtt{inj}^A_r V)\,\mathtt{with}\,(x^{!^n\,A} \mapsto M \mid y^{!^n\,B} \mapsto N)\,] \\ \hfill \to_1 E[\,N[V/y]\,] \\ E[\,\mathtt{letrec}\,f^{A\multimap B}x = M\,\mathtt{in}\,N\,] \\ \qquad \to_1 E[\,N[\lambda x^A.\mathtt{letrec}\,f^{A\multimap B}x = M\,\mathtt{in}\,M/f]\,] \\ E[\,\mathtt{meas}^{n+1}_i(\mathtt{new}\,\rho)\,] \to_1 E[\,\langle\,\mathtt{tt}, \mathtt{new}\,\langle 0_i|\rho|0_i\rangle\,\rangle\,] \\ E[\,\mathtt{meas}^{n+1}_i(\mathtt{new}\,\rho)\,] \to_1 E[\,\langle\,\mathtt{ff}, \mathtt{new}\,\langle 1_i|\rho|1_i\rangle\,\rangle\,] \\ E[\,\mathtt{meas}^1_1(\mathtt{new}\,\rho)\,] \to_{\langle 0|\rho|0\rangle} E[\,\mathtt{tt}\,] \\ E[\,\mathtt{meas}^1_1(\mathtt{new}\,\rho)\,] \to_{\langle 1|\rho|1\rangle} E[\,\mathtt{ff}\,] \\ E[\,U(\mathtt{new}\,\rho)\,] \to_1 E[\,\mathtt{new}\,(U\rho)\,] \\ E[\,\mathtt{cmp}_{m,n}\langle \mathtt{new}\,\rho, \mathtt{new}\,\sigma\rangle\,] \to_1 E[\,\mathtt{new}\,(\rho \otimes \sigma)\,] \end{array}$$

Here M, N are terms, V, W are values and $n \geq 1$. The reductions involving $\mathtt{new}\,\rho$ occur only when the dimensions match. The measurement rules always give rise to two reductions in a pair (corresponding to $|0\rangle$ and $|1\rangle$); they are said to be the *partner* to each other.

An *evaluation* is a series of reductions.

Observe that the label p in reduction $\to_p$ is like a probability but not quite: from $\mathtt{meas}^2_i(\mathtt{new}\,\rho)$ there are two $\to_1$ reductions, to $\mathtt{new}\,\langle 0_i|\rho|0_i\rangle$ and to $\mathtt{new}\,\langle 1_i|\rho|1_i\rangle$. Again, probabilities are implicitly carried by the trace values.

Next we derive, from the small-step semantics, big-step semantics for `bit`-type closed terms. To handle recursion, we follow the standard method and use explicit counters.

Definition VI.4 (Big-step semantics)**.** For each $n \in \mathbb{N}$ we define a relation $\Downarrow^n$ between closed `bit`-terms M and pairs (p, q) of real numbers. This is by induction on n.

For $n = 0$, we set

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

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

$$M \Downarrow^{n+1} (p,q) \text{ if } M' \Downarrow^n (p,q)\ \ .$$

If M has a reduction $M \to_r N$ caused by the measurement rule, there is always its partner reduction $M \to_{r'} N'$. We set

$$M \Downarrow^{n+1} (rp + r'p', rq + r'q') \quad \text{if } N \Downarrow^n (p,q) \text{ and } N' \Downarrow^n (p',q').$$

Finally, we define a relation $\Downarrow$ by: $M \Downarrow (p,q)$ if

$$(p,q) = \sup\{\, (p',q') \mid M \Downarrow^n (p',q') \text{ for some } n \,\}\ \ ,$$

where $\sup$ is with respect to the pointwise order in $[0,1]^2$. It is easy to see that for each M and n, there is only one (p,q) such that $M \Downarrow^n (p,q)$; hence the same for $\Downarrow$.

Theorem VI.5 (Adequacy)**.** *For any closed* `bit`*-term M, we have $M \Downarrow (p,q)$ if and only if $M \curlyvee (p,q)$.*

Proof. The proof and some lemmas (such as normalization for the recursion-free fragment) are in [14, Appendix F]. □

Acknowledgment

Thanks are due to Peter Selinger for a number of insightful comments; to Kazuyuki Asada, Bart Jacobs, Phil Scott and the anonymous reviewers for useful discussions and comments; and to Andy Pitts for his ETAPS'07 talk that inspired the current work. The second author is supported by JSPS Research Fellowships for Young Scientists.

References

[1] P. Selinger, "Towards a quantum programming language," *Math. Struct. in Comp. Sci.*, vol. 14, no. 4, pp. 527–586, 2004.

[2] P. Selinger and B. Valiron, "On a fully abstract model for a quantum linear functional language: (extended abstract)," *Elect. Notes in Theor. Comp. Sci.*, vol. 210, pp. 123–137, 2008.

[3] ——, "Quantum lambda calculus," in *Semantic Techniques in Quantum Computation*, S. Gay and I. Mackie, Eds. Cambridge Univ. Press, 2009, pp. 135–172.

[4] Y. Delbecque and P. Panangaden, "Game semantics for quantum stores," *Elect. Notes in Theor. Comp. Sci.*, vol. 218, pp. 153–170, 2008.

[5] I. Hasuo, B. Jacobs, and A. Sokolova, "Generic trace semantics via coinduction," *Logical Methods in Comp. Sci.*, vol. 3, no. 4:11, 2007.

[6] J.-Y. Girard, "Geometry of interaction I: Interpretation of system F," in *Logic Colloquium 88*, R. F. et al., Ed. North-Holland, 1989, pp. 221–260.

[7] S. Abramsky, E. Haghverdi, and P. Scott, "Geometry of interaction and linear combinatory algebras," *Math. Struct. in Comp. Sci.*, vol. 12, no. 5, pp. 625–665, 2002.

[8] S. Abramsky and M. Lenisa, "Linear realizability and full completeness for typed lambda-calculi," *Ann. Pure & Appl. Logic*, vol. 134, no. 2–3, pp. 122–168, 2005.

[9] A. Joyal, R. Street, and D. Verity, "Traced monoidal categories," *Math. Proc. Cambridge Phil. Soc.*, vol. 119(3), pp. 425–446, 1996.

[10] S. Abramsky, R. Jagadeesan, and P. Malacaria, "Full abstraction for PCF," *Inf. & Comp.*, vol. 163, no. 2, pp. 409–470, 2000.

[11] J. M. E. Hyland and C.-H. L. Ong, "On full abstraction for PCF: I, II, and III," *Inf. & Comp.*, vol. 163, no. 2, pp. 285–408, 2000.

[12] I. Mackie, "The geometry of interaction machine," in *POPL '95: Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. New York, NY, USA: ACM, 1995, pp. 198–208.

[13] P. Scott, "Tutorial on geometry of interaction," Tutorial talk at FMCS 2004, 2004, slides available online.

[14] I. Hasuo and N. Hoshino, "Semantics of higher-order quantum computation via geometry of interaction," Extended version, to appear in RIMS Preprints, April 2011.

[15] K. Kraus, *States, effects and operations. Fundamental notions of quantum theory*, ser. Lect. Notes Phys. Springer-Verlag, 1983, vol. 190.

[16] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[17] E. Moggi, "Notions of computation and monads," *Inf. & Comp.*, vol. 93(1), pp. 55–92, 1991.

[18] G. M. Bierman, "What is a categorical model of intuitionistic linear logic," in *Typed Lambda Calculi and Applications*, ser. Lect. Notes Comp. Sci., M. Dezani-Ciancaglini and G. Plotkin, Eds., no. 902. Springer, Berlin, 1995, pp. 78–93.

[19] P. N. Benton and P. Wadler, "Linear logic, monads and the lambda calculus," in *LICS*, 1996, pp. 420–431.

[20] N. Hoshino, "Linear realizability," in *CSL*, ser. Lect. Notes Comp. Sci., J. Duparc and T. A. Henzinger, Eds., vol. 4646. Springer, 2007, pp. 420–434.

[21] B. Jacobs, "From coalgebraic to monoidal traces," in *Coalgebraic Methods in Computer Science (CMCS 2010)*, ser. Elect. Notes in Theor. Comp. Sci., vol. 264. Elsevier, Amsterdam, 2010.

[22] E. Haghverdi, "A categorical approach to linear logic, geometry of proofs and full completeness," Ph.D. dissertation, Univ. of Ottawa, 2000.

[23] M. Abadi and G. D. Plotkin, "A per model of polymorphism and recursive types," in *LICS*. IEEE Computer Society, 1990, pp. 355–365.