Reversible Monadic Computing

Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 319 (2015) 217–237

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Reversible Monadic Computing

Chris Heunen[1,2]

Department of Computer Science
University of Oxford
United Kingdom

Martti Karvonen[3]

Department of Mathematics and Systems Analysis
Aalto University
Finland

Abstract

We extend categorical semantics of monadic programming to reversible computing, by considering monoidal closed dagger categories: the dagger gives reversibility, whereas closure gives higher-order expressivity. We demonstrate that Frobenius monads model the appropriate notion of coherence between the dagger and closure by reinforcing Cayley's theorem; by proving that effectful computations (Kleisli morphisms) are reversible precisely when the monad is Frobenius; by characterizing the largest reversible subcategory of Eilenberg–Moore algebras; and by identifying the latter algebras as measurements in our leading example of quantum computing. Strong Frobenius monads are characterized internally by Frobenius monoids.

Keywords: Frobenius monad, dagger category, reversible computing, quantum measurement

1 Introduction

The categorical concept of a *monad* has been tremendously useful in programming, as it extends purely functional programs with nonfunctional effects. For example, using monads one can extend a functional programming language with nondeterminism, probabilism, stateful computing, error handling, read-only environments, and input and output [51]. Haskell incorporates monads in its core language. On the theoretical side, there are satisfyingly clean categorical semantics. Simply typed λ-calculus, that may be regarded as an idealized functional programming language,

[1] Supported by the Engineering and Physical Sciences Research Council Fellowship EP/L002388/1. We thank an anonymous referee for Example 6.4, and Jorik Mandemaker, Sean Tull, and Maciej Pirog for helpful discussions.

[2] Email:heunen@cs.ox.ac.uk

[3] Email:martti.karvonen@aalto.fi

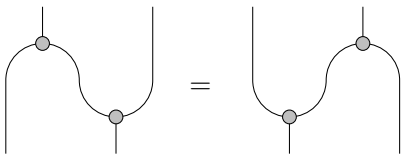
http://dx.doi.org/10.1016/j.entcs.2015.12.014

takes semantics in Cartesian closed categories [31]. The functional programming concept of a monad is modeled by the categorical concept of a monad [36].

In classical computation it is not always possible to reconstruct the input to an algorithm from its output. However, by using auxiliary bits, any classical computation can be turned into a *reversible* one [48]. Such a computation uses invertible primitive gates, and composition preserves invertibility. As discarding information requires work, reversible computations could in principle be implemented at higher speeds. The only operation costing power is the final discarding of auxiliary bits.

This is brought to a head in *quantum computing*, where any deterministic evolution of quantum bits is invertible, unlike the eventual measurement that converts quantum information to classical information. Another novelty in quantum computing is that it is impossible to copy or delete quantum information. This leads to a linear type theory of resources rather than a classical one [47]: quantum computing takes semantics in monoidal categories, rather than Cartesian ones [2].

Led by quantum computing, this article extends the categorical semantics of monadic programming to reversible computing. To allow for a linear type theory we consider monoidal closed categories. To allow for reversible computations, we consider *dagger categories*; in general these correspond to bidirectional computations rather than invertible ones, which in the quantum case comes down to the same thing. To allow for monadic effects, we introduce *Frobenius monads*. In the presence of a dagger, any monad gives rise to a comonad; a Frobenius monad is one that interacts with its comonad counterpart via the following *Frobenius law*:



(1)

Here we used the graphical calculus for monoidal categories [44,34], that will be explained further in Section 2, along with several examples. [4]

Our main contribution is to take reversal as a primitive and so justify the claim that Frobenius monads are precisely the right notion as follows:

- Section 3 justifies the Frobenius law as a necessary (and sufficient) consequence of coherence between the dagger and closure. In a reversible setting, it is natural to consider *involutive* monoids. In a monoidal closed category, any monoid embeds into a canonical one by Cayley's theorem. We prove that this embedding preserves the involution induced by the dagger if and only if the monoid satisfies the Frobenius law. This derivation from first principles is a noncommutative generalization of [41, Theorem 4.3] with a new proof.
- Section 4 characterizes Frobenius monads *internally*. Monads are an *external* notion. A good example is the writer monad, that allows programs to keep auxiliary

[4] We often need to reason simultaneously about morphisms *in* a monoidal category and endofunctors *on* it. Unfortunately there is no sound and complete graphical proof calculus that would handle this yet. Therefore we cannot use the graphical calculus exclusively and also have to use traditional commutative diagrams.

output alongside the computation. These values accumulate according to some monoid. Any monoid gives rise to a strong monad, and Frobenius monoids give rise to strong Frobenius monads. In general this is merely an adjunction and not an equivalence, but we work out that the converse holds in the Frobenius setting. This is a noncommutative generalization of [41, Corollary 4.5]. It also generalizes the classic Eilenberg–Watts theorem from homological algebra to categories that are not necessarily abelian. As Frobenius monoids satisfy the very same law (1) as Frobenius monads, only interpreted in a category rather than by endofunctors on it, this also exhibits that reversible settings are closed under categorification.

- We show that the extension of reversible pure computations with effects modeled by a monad results in reversible effectful computations if and only if the monad is a Frobenius monad. More precisely, Section 5 shows that a monad on a dagger category is a Frobenius monad if and only if the dagger extends to the category of Kleisli algebras. This reinforces that Frobenius monads model the right notion of effects for reversible computing. Section 6 identifies the largest subcategory of all algebras with this property, which we call *Frobenius–Eilenberg–Moore* algebras. Section 7 exemplifies them in the quantum setting by arguing that they correspond precisely to measurements via *effect handlers* [42].

Frobenius monads have been studied before [46,32], and monads have been used as semantics for quantum computing before [15,4,3], but not in a dagger setting, except for [41] that deals with the commutative case abstractly. Conversely, reversible programming has been modeled in dagger categories [6], but not using monads. Daggers and monads were combined before in coalgebra [20,24], quantum programming languages programming languages [14,45], and matrix algebra [11]. The current work differs by systematically starting from first principles. We intend to fit probabilistic programming in our setup in future work.

2 Dagger categories

Let us model types as objects $A, B, C, \ldots$in a category, and computations as morphisms $f, g, h, \ldots$. To model composite types, we consider *monoidal categories*, where one can not only compose computations in sequence $A \xrightarrow{f} B \xrightarrow{g} C$, but also in parallel $A \otimes B \xrightarrow{f \otimes g} C \otimes D$. This much is standard [5]. To model *reversible* computations, we need an operation turning a computation $A \xrightarrow{f} B$ into a computation $B \to A$, such that reversing twice doesn't do anything.

Definition 2.1 A *dagger* is a functor $\dagger\colon \mathbf{C}^{\mathrm{op}} \to \mathbf{C}$ satisfying $A^\dagger = A$ on objects and $f^{\dagger\dagger} = f$ on morphisms. A *dagger category* is a category equipped with a dagger.

Dagger categories can behave quite different from ordinary (non-dagger) ones, see *e.g.* [49, 9.7]. They are especially useful as semantics for quantum computing [19]. Note that *reversible* computing does not mean computations are *invertible*. An invertible morphism f in a dagger category is *unitary* when $f^\dagger = f^{-1}$. Similarly, an endomorphism f is *self-adjoint* when $f = f^\dagger$. As a rule, any structure in sight should cooperate with the dagger.

Definition 2.2 A monoidal category is called a *monoidal dagger category* when $(f \otimes g)^\dagger = f^\dagger \otimes g^\dagger$, and all coherence isomorphisms $A \otimes (B \otimes C) \xrightarrow{\alpha} (A \otimes B) \otimes C$, $I \otimes A \xrightarrow{\lambda} A$, and $A \otimes I \xrightarrow{\rho} A$, are unitary. In a *symmetric monoidal dagger category* additionally the swap maps $A \otimes B \xrightarrow{\sigma} B \otimes A$ are unitary.

We will mainly consider the following two examples.

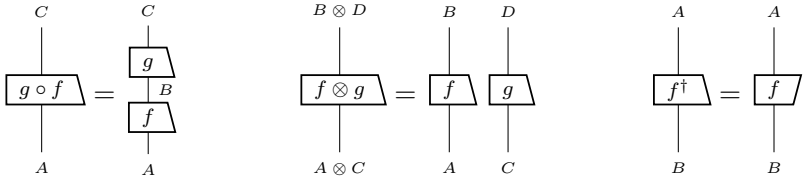
Example 2.3 The symmetric monoidal dagger category **Rel** has sets as objects. Morphisms $A \to B$ are *relations* $R \subseteq A \times B$, with composition $S \circ R = \{(a,c) \mid \exists b\colon (a,b) \in R, (b,c) \in S\}$. The dagger is given by $R^\dagger = \{(b,a) \mid (a,b \in R)\}$, and the monoidal structure is given by Cartesian products. We may think of **Rel** as modeling *nondeterministic computation* [22].

Example 2.4 The symmetric monoidal dagger category **FHilb** has finite-dimensional complex *Hilbert spaces* as objects and linear maps as morphisms. The dagger is given by adjoints: $f^\dagger$ is the unique linear function satisfying $\langle f(x) \,|\, y\rangle = \langle x \,|\, f^\dagger(y)\rangle$; in terms of matrices it is the conjugate transpose. The monoidal structure is given by tensor products of Hilbert spaces. This models *quantum computation* [2].

There are many other examples. Reversible probabilistic computation is modelled by the category of doubly stochastic maps [7, 2.3.5]; this generalizes to labelled Markov chains [38]. Universal constructions can generate examples with specific properties [40]. Finally, one can formally add daggers to a category in a free or cofree way [16, 3.1.17 and 3.1.19]. We will be interested in the following way to turn a monoidal dagger category into a new one of endofunctors on the old one. It could be regarded as modeling *second-order computation*, because the computations in the new category may refer to computations in the old one (but not to themselves).

Example 2.5 A functor $\mathbf{C} \xrightarrow{F} \mathbf{D}$ between dagger categories is a *dagger functor* when $F(f^\dagger) = F(f)^\dagger$ on morphisms. Let $\mathbf{C}$ be a monoidal dagger category. If $F(A) \xrightarrow{\beta_A} G(A)$ is a natural transformation between dagger functors $\mathbf{C} \xrightarrow{F,G} \mathbf{C}$, then so is $G(A) \xrightarrow{\beta_A^\dagger} F(A)$. Thus the category $[\mathbf{C},\mathbf{C}]_\dagger$ of dagger functors $\mathbf{C} \to \mathbf{C}$ is again a monoidal dagger category by $G \otimes F = G \circ F$.

Monoidal dagger categories have a sound and complete *graphical calculus*, that we briefly recall; for more details, see [44]. A morphism $A \xrightarrow{f} B$ is represented as  , and composition, the tensor product, and the dagger, become:

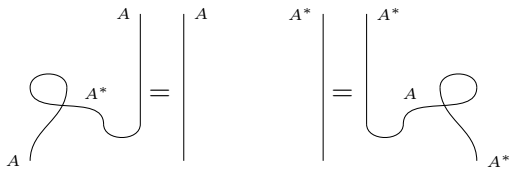


Notice that the output wire $B \otimes D$ of a morphism $A \xrightarrow{f} B \otimes D$ becomes a pair

of wires labelled B and D coming out of the box labelled f. Also, the dagger reflects in the horizontal axis, which is why we draw the boxes asymmetrically. Distinguished morphisms are often depicted with special diagrams instead of generic boxes as above. For example, the identity $A \to A$ is just the line $|$; the (identity on) the monoidal unit object I is drawn as the empty picture, and the swap map of symmetric monoidal categories becomes $\times$. Soundness and completeness means that any equality between morphisms one can prove algebraically using the axioms of monoidal dagger categories can equivalently and rigorously be proven graphically by isotopies of the graphical diagram.

To model *higher order computation*, we need function types. This is usually done by requiring *closed monoidal categories*, where the functors $-\otimes B$ have right adjoints $B \multimap -$. That is, there is a natural bijective correspondence between morphisms $B \otimes A \xrightarrow{f} C$ and their *curried* version $A \xrightarrow{\Lambda(f)} (B \multimap C)$. In the reversible setting of monoidal dagger categories, this closure operation should cooperate with the dagger: since $B \multimap C$ is the type of computations $B \xrightarrow{f} C$, and those computations can be reversed to $C \xrightarrow{f^\dagger} B$, there should be an operation $(B \multimap C) \to (C \multimap B)$ modelling this internally (we will see this in more detail in Section 3). Therefore we demand that $B \multimap -$ are dagger functors. It follows that they are not just right adjoint to $- \otimes B$, but also left adjoint. Now it is a small step to so-called *compact dagger categories* [33,27], which we make here for the sake of simplicity.

Definition 2.6 A *compact dagger category* is a symmetric monoidal dagger category in which every object A has a chosen dual object A^* and a morphism $I \xrightarrow{u} A^* \otimes A$, drawn as $\smile$, satisfying $\big((u^\dagger \circ \sigma) \otimes \mathrm{id}\big) \circ (\mathrm{id} \otimes u) = \mathrm{id}$ and its dual:



(2)

Compact dagger categories are automatically closed monoidal, with $(B \multimap C) = B^* \otimes C$. Think of dual objects B^* as *input* types, and primal objects C as *output* types. By convention we choose $A^{**} = A$ and $(A \otimes B)^* = B^* \otimes A^*$.

Our previous examples in fact already satisfy this closure property of *higher order computation*: **Rel** and **FHilb** are compact dagger categories as follows. In **Rel** we can take $A^* = A$ and $u = \{(*, (a, a)) \mid a \in A\}$ for $I = \{*\}$. In **FHilb** we can take H^* to be the dual Hilbert space of H; if H has an orthonormal basis $\{e_1, \ldots, e_n\}$, then H^* has an orthonormal basis $\{e_1^*, \ldots, e_n^*\}$, and we can take $u(1) = \sum_{i=1}^n e_i^* \otimes e_i$. There is also a *free compact dagger category* on a given (dagger) category **C** [1].

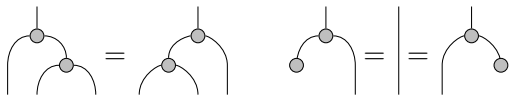
Let us conclude this preparatory section by contrasting reversible computing and *invertible computing*. A *groupoid* is a category where any morphism is invertible; it is always a dagger category with $f^\dagger = f^{-1}$. Any symmetric monoidal closed groupoid **G** is a so-called compact category with $A^* = (A \multimap I)$, as follows. Closure gives isomorphisms $(A \multimap B) \otimes A \xrightarrow{\mathrm{ev}} B$ for all objects A and B; in particular,

$I \cong A^* \otimes A$. The morphisms $\Lambda(\mathrm{ev})$ are isomorphisms $A \cong A^{**}$, making $\mathbf{G}$ into a so-called $*$-autonomous category [5]. Because $\mathbf{G}$ is symmetric monoidal, there are isomorphisms $A^* \otimes B^* \xrightarrow{\Lambda(\mathrm{ev}\otimes\mathrm{ev})} (A \otimes B)^*$, making $\mathbf{G}$ a compact category. However, this is not a compact dagger category unless all swap maps σ are identities.

3 Frobenius monoids

This section considers monoids in monoidal dagger categories. We will see that, in the higher order setting of closed monoidal categories, our rule of thumb that everything should cooperate with the dagger means considering *Frobenius monoids*.

Definition 3.1 A *monoid* in a monoidal category is an object A with morphisms : $A \otimes A \to A$ and : $I \to A$, satisfying:



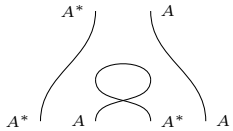
It is *commutative* when  =  $\circ\sigma$. A *Frobenius monoid* is a monoid in a monoidal dagger category satisfying (1). It is *special* when  $\circ$  $^\dagger = \mathrm{id}_A$.

A *comonoid* in $\mathbf{C}$ is a monoid in $\mathbf{C}^{\mathrm{op}}$. The Frobenius law (1) makes sense for pairs of a monoid and comonoid on the same object, and most of Section 4 holds in that generality. Each side of the Frobenius law (1) equals  $^\dagger \circ$  ; one of these equations is equivalent to (1). It is mostly motivated by observing that Frobenius monoids in specific categories are appropriate well-known mathematical structures.

Example 3.2 Frobenius monoids in **FHilb** correspond to finite-dimensional *C*-algebras* [50, Theorem 4.6]. These play a major role in quantum computing [28], but also as semantics for labelled Markov processes with bisimulations [35,43,30,37] and as operational semantics of probabilistic languages [12,13]. Commutative Frobenius monoids in **FHilb** therefore correspond to orthonormal bases when special [9].

Example 3.3 Frobenius monoids in **Rel** correspond to (small) *groupoids* [18,39], which are important to invertible computing.

Example 3.4 In a compact dagger category, $A^* \otimes A$ is a Frobenius monoid with  $= u$, and  being the *pair of pants*:

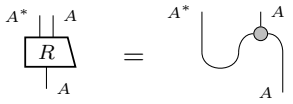


This is precisely the monoid $A \multimap A$ of computations $A \to A$ under composition.

Pair of pants are universal, as the following generalization of Cayley's theorem shows. A *monoid homomorphism* f satisfies $= f \circ$  and $f \circ$ = $\circ (f \otimes f)$.

Lemma 3.5 *Any monoid* $(A, \curvearrowleft, \downarrow)$ *in a compact category allows a monic monoid homomorphism* R *into* $A^* \otimes A$.

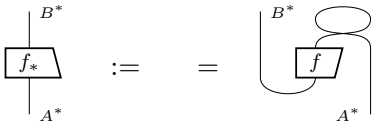
Proof. The following is a monoid homomorphism by (1):



(3)

It is monic because it has a left inverse $\big((\downarrow)^\dagger \otimes \mathrm{id}\big) \circ R$. □

We will prove that the Cayley embedding of the previous lemma respects daggers precisely when the monoid is a Frobenius monoid. To make precise what it means to respect daggers, we need to internalize the operation $f \mapsto f^\dagger$ from $A \xrightarrow{f} A$ to the monoid $A \multimap A$. But the former might not be a well-defined morphism; for example, in **FHilb**, taking conjugate transpose matrices is *anti*-linear, not linear, and hence a morphism $(A \multimap A) \to (A \multimap A)^*$ rather than an endomorphism. In a compact category, this is modeled by



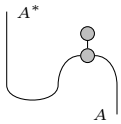
for $A \xrightarrow{f} B$. The operation $f \mapsto f^\dagger$ additionally is contravariant: $(g \circ f)^\dagger = f^\dagger \circ g^\dagger$. So for it to be a monoid homomorphism the codomain has to have opposite multiplication as the domain.

Lemma 3.6 *If* $(A, \curvearrowleft, \downarrow)$ *is a monoid in a compact category, then so is* $(A^*, \curvearrowleft_*, \downarrow_*)$*, called the* opposite monoid.

Proof. The functor $f \mapsto f_*$ is (strong) monoidal. □

Definition 3.7 A monoid $(A, \curvearrowleft, \downarrow)$ in a compact dagger category is an *involutive monoid* when it is equipped with an *involution*: a monoid homomorphism $A \xrightarrow{i} A^*$ satisfying $i_* \circ i = \mathrm{id}$. A *homomorphism of involutive monoids* is a monoid homomorphism $A \xrightarrow{f} B$ satisfying $i \circ f = f_* \circ i$.

Note that there is a canonical choice of involution:

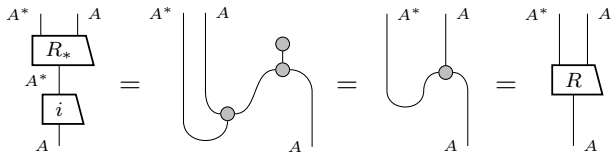


(4)

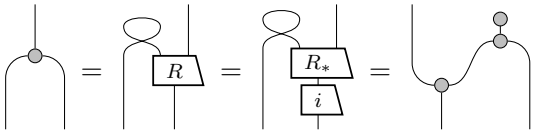
For the groupoids of Example 3.3, it is $g \mapsto g^{-1}$. For the C*-algebras of Example 3.2, it is $a \mapsto a^*$. The following theorem justifies the Frobenius law from first principles, generalizing [41, Theorem 4.3] noncommutatively.

Theorem 3.8 *A monoid in a compact dagger category is a Frobenius monoid if and only if* (4) *makes it involutive and* (3) *a homomorphism of involutive monoids.*

Proof. Write $(A, \curvearrowright, \downarrow)$ for the monoid, and i for (4). If A is a Frobenius monoid, it follows from (1) that i is indeed an involution. Observe that the involution on $A^* \otimes A$ is the identity because of our convention $(A \otimes B)^* = B^* \otimes A^*$. So (3) preserves involutions when $R_* \circ i = R$:

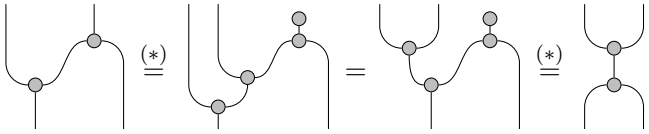


Conversely, assuming $R_* \circ i = R$:



(∗)

Hence, by associativity:



But this is equivalent to (1). □

4 Frobenius monads

A monad is a functor $\mathbf{C} \xrightarrow{T} \mathbf{C}$ with natural transformations $T(T(A)) \xrightarrow{\mu_A} T(A)$ and $A \xrightarrow{\eta_A} T(A)$ satisfying certain laws. It is well-known that monads are precisely monoids in categories of functors $\mathbf{C} \to \mathbf{C}$: Definition 3.1 unfolds to the monad laws

$$\mu_A \circ T(\mu_A) = \mu_A \circ \mu_{T(A)},$$
$$\mu_A \circ T(\eta_A) = \mathrm{id}_{T(A)} = \mu_A \circ \eta_{T(A)}.$$

There is a dual notion of a comonad. Daggers make any monoid (monad) give rise to a comonoid (comonad). Thus the Frobenius law (1) lifts to monads as follows.

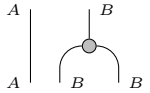
Definition 4.1 A *Frobenius monad* on a dagger category $\mathbf{C}$ is a Frobenius monoid in $[\mathbf{C}, \mathbf{C}]_\dagger$; explicitly, a monad (T, μ, η) on $\mathbf{C}$ with $T(f^\dagger) = T(f)^\dagger$ and

$$T(\mu_A) \circ \mu^\dagger_{T(A)} = \mu_{T(A)} \circ T(\mu^\dagger_A).$$

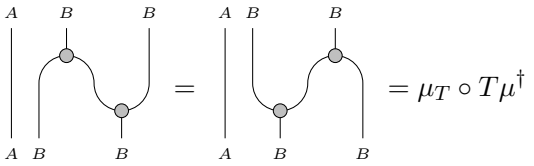
It is *special* when $\mu_A \circ \mu^\dagger_A = \mathrm{id}_{T(A)}$.

Frobenius monads have been studied before by Street [46,32]. His definition does not take daggers into account, and concerns a monad rather than a monad-comonad pair. However, the natural generalization of the above definition to (non-dagger) monad-comonad pairs results in an equivalent notion to the one studied by Street. The primary example of a Frobenius monad is taking tensor products with a Frobenius monad.

Example 4.2 If $(B, \curvearrowright, \downarrow)$ is a Frobenius monoid in a monoidal dagger category $\mathbf{C}$, then the functor $\mathbf{C} \xrightarrow{-\otimes B} \mathbf{C}$, given by $A \mapsto A \otimes B$ and $f \mapsto f \otimes \mathrm{id}$, is a Frobenius monad on $\mathbf{C}$ with:

$\mu_A =$  $\eta_A =$ 

Proof. The Frobenius monad law simply comes down to the Frobenius monoid law:

$T\mu \circ \mu_T^\dagger =$  $= \mu_T \circ T\mu^\dagger$

The monad laws become the monoid laws. Taking $A = I$, we thus see that $- \otimes B$ is a Frobenius monad if and only if B is a Frobenius monoid. □

This section characterizes Frobenius monads of this form. There are, however, also other Frobenius monads, as in the following example.

Example 4.3 Consider the monoid $\mathbf{Rel}(\mathbb{N}, \mathbb{N})$ of all relations $\mathbb{N} \to \mathbb{N}$ as a single-object category. The following define a Frobenius monad on this category:

$$\begin{aligned} T(R) =& \{(2m, 2n) \mid (m, n) \in R\} \\ & \cup \{(2m+1, 2n+1) \mid (m, n) \in R\} \\ \eta =& \{(2n, 2n+1) \mid n \in \mathbb{N}\} \\ \mu =& \{(4n, 2n) \mid n \in \mathbb{N}\} \cup \{(4n+3, 2n+1) \mid n \in \mathbb{N}\} \end{aligned}$$

The functor $- \otimes B$ comes with a natural transformation $\alpha_{-,-,B}$, making it a strong functor. This natural transformation respects the monoid structure on B. Before recording some folklore results, we first define what this means for monads.

Definition 4.4 A functor F between monoidal categories is *strong* when it is equipped with a natural transformation $A \otimes F(B) \xrightarrow{\mathrm{st}_{A,B}} F(A \otimes B)$ satisfying $\mathrm{st} \circ \alpha = F(\alpha) \circ \mathrm{st} \circ (\mathrm{id} \otimes \mathrm{st})$ and $F(\lambda) \circ \mathrm{st} = \lambda$. A *morphism of strong functors* is a natural transformation $F \xrightarrow{\beta} G$ satisfying $\beta \circ \mathrm{st} = \mathrm{st} \circ (\mathrm{id} \otimes \beta)$ A *strong monad* is a monad (T, μ, η) that is a strong functor satisfying $\mathrm{st} \circ (\mathrm{id} \otimes \mu) = \mu \circ T(\mathrm{st}) \circ \mathrm{st}$ and $\mathrm{st} \circ (\mathrm{id} \otimes \eta) = \eta$. A *morphism of strong monads* is a natural transformation, which is a morphism of the underlying monads and the underlying strong functors.

Proposition 4.5 *Let* $\mathbf{C}$ *be a monoidal category. The operations* $B \mapsto - \otimes B$ *and* $T \mapsto T(I)$ *define an adjunction between monoids in* $\mathbf{C}$ *and strong monads on* $\mathbf{C}$*, with* $B \mapsto - \otimes B$ *being the left adjoint.*

Proof. See [52]. The unit of the adjunction is $I \otimes B \xrightarrow{\lambda_B} B$. The counit is determined by $A \otimes T(I) \xrightarrow{T(\rho) \circ \mathrm{st}} T(A)$. □

In the case of symmetric monoidal categories, there is also a notion of commutativity for strong monads [29,23]. Given a strong monad T, one can define a natural transformation $T(A) \otimes B \xrightarrow{\mathrm{st}'_{A,B}} T(A \otimes B)$ by $T(\sigma_{B,A}) \circ st_{B,A} \circ \sigma_{T(A),B}$, and

$$dst_{A,B} := \mu_{A\otimes B} \circ T(st'_{A,B}) \circ st_{T(A),B}$$
$$dst'_{A,B} := \mu_{A\otimes B} \circ T(st_{A,B}) \circ st'_{A,T(B)}$$

A strong monad is *commutative* when these coincide. Proposition 4.5 restricts to an adjunction between commutative monoids and commutative monads [52].

Definition 4.6 A *costrong functor* $\mathbf{C} \xrightarrow{F} \mathbf{D}$ between monoidal categories is a functor that is strong when considered as a functor $\mathbf{C}^{\mathrm{op}} \to \mathbf{D}^{\mathrm{op}}$. Explicitly, it has a natural transformation $F(A \otimes B) \xrightarrow{\mathrm{cst}_{A,B}} A \otimes F(B)$ satisfying $F(\lambda) = \lambda \circ \mathrm{cst}_{I,A}$ and $\mathrm{cst}_{A\otimes B,C} \circ F(\alpha) = \alpha \circ (\mathrm{id} \otimes \mathrm{cst}_{B,C}) \circ \mathrm{cst}_{A,B\otimes C}$. A *morphism of costrong functors* is a natural transformation $F \xrightarrow{\beta} G$ satisfying $\mathrm{cst} \circ \beta = (\mathrm{id} \otimes \beta) \circ \mathrm{cst}$. A *costrong comonad* is a comonad (T, δ, ε) that is a costrong functor, such that $(\mathrm{id} \otimes \varepsilon) \circ \mathrm{cst} = \varepsilon$ and $\mathrm{cst} \circ T(\mathrm{cst}) \circ \delta = (\mathrm{id} \otimes \delta) \circ \mathrm{cst}$. A *morphism of costrong comonads* is a natural transformation, which is a morphism of the underlying comonads and the underlying costrong functors.

Corollary 4.7 *Let* $\mathbf{C}$ *be a monoidal category. The operations* $B \mapsto - \otimes B$ *and* $T \mapsto T(I)$ *form an adjunction between comonoids in* $\mathbf{C}$ *and costrong comonads on* $\mathbf{C}$*, but this time* $B \mapsto - \otimes B$ *is the right adjoint.* □

In our reversible setting of dagger categories, any strong monad T is automatically a costrong comonad under $\mathrm{cst} = \mathrm{st}^\dagger$, $\delta = \mu^\dagger$, and $\varepsilon = \eta^\dagger$. According to our motto that everything in sight should cooperate with the dagger, the reverse cst of st should in fact be its inverse, leading to the following definition.

Definition 4.8 A *strong Frobenius monad* on a monoidal dagger category $\mathbf{C}$ is a Frobenius monad (T, μ, η) that is simultaneously a strong monad, such that each st_A is unitary. A *morphism of strong Frobenius monads* is just a morphism of the underlying strong monads.

The following theorem promotes the adjunction of Proposition 4.5 and Corollary 4.7 into an equivalence in the dagger setting. It generalizes [41, Theorem 4.5] noncommutatively. It also generalizes the classic Eilenberg–Watts theorem, that characterizes certain endofunctors on abelian categories as being of the form $- \otimes B$ for a monoid B, to monoidal dagger categories; note that there are monoidal dagger categories that are not abelian, such as **Rel** and **Hilb** [17, Appendix A].

Theorem 4.9 *Let* $\mathbf{C}$ *be a monoidal dagger category. The operations* $B \mapsto - \otimes B$ *and* $T \mapsto T(I)$ *define an equivalence between Frobenius monoids in* $\mathbf{C}$ *and strong Frobenius monads on* $\mathbf{C}$*.*

Proof. We already saw in Example 4.2 that $B \mapsto - \otimes B$ preserves the Frobenius law. We prove that $T \mapsto T(I)$ preserves the Frobenius law, too, in Lemma A.2 in the Appendix. It remains to prove that they form an equivalence. Clearly the unit of the adjunction, $I \otimes B \xrightarrow{\lambda_B} B$, is a natural isomorphism. To prove that the counit $A \otimes T(I) \xrightarrow{T(\rho)\circ\mathrm{st}} T(A)$ is also a natural isomorphism, notice that by definition it is a morphism of strong monads. In Lemma A.3 in the Appendix we prove that it is also a morphism of comonads. But homomorphisms of Frobenius monoids must be isomorphisms by Lemma A.1. □

The previous theorem restricts to an equivalence between commutative/special Frobenius monoids and commutative/special strong Frobenius monads (see Corollary A.4 in the Appendix).

One might think it too strong to require st to be unitary. The following counterexample shows that Theorem 4.9 would fail if we abandoned that requirement.

Example 4.10 Let's call a Frobenius monad *rather strong* when it is simultaneously a strong monad. The operations of Theorem 4.9 do not form an adjunction between Frobenius monoids and rather strong Frobenius monads, because the counit of the adjunction would not be a well-defined morphism. To produce a counterexample where the counit does not preserve comultiplication comes down to finding a rather strong Frobenius monad with $T(\eta_A) \circ \eta_A \neq \mu_A^\dagger \circ \eta_A$ for some A. This is the case when T is $- \otimes B$ for a Frobenius monoid B with $\delta \otimes \delta \neq (\cap)^\dagger \circ \delta$. Such Frobenius monoids certainly exist: if G is any nontrivial group, regarded as a Frobenius monoid in **Rel** via Example 3.3, then $\delta \otimes \delta$ is the relation $\{(*, (1,1))\}$, but $(\cap)^\dagger \circ \delta = \{(*, (g, g^{-1})) \mid g \in G\}$.

5 Kleisli algebras

One of the standard categorical constructions when given a monad T is to consider the category $\mathbf{C}_T$ of its Kleisli algebras. In monadic programming, this category gives semantics for computations with effects modeled by T, whereas the base category $\mathbf{C}$ only gives semantics for pure computations [25]. In this section we show that if T is a Frobenius monad, then $\mathbf{C}_T$ is a dagger category. In fact we also show the converse, under a natural condition about cooperation with daggers. Thus effects modeled by a monad can be added without leaving the setting of reversible computations precisely when the monad is a Frobenius monad.

Definition 5.1 If $\mathbf{C} \xrightarrow{T} \mathbf{C}$ is a monad, its *Kleisli category* $\mathbf{C}_T$ is defined as follows. Objects are the same as in $\mathbf{C}$. A morphism $A \to B$ in $\mathbf{C}_T$ is a morphism $A \xrightarrow{f} T(B)$ in $\mathbf{C}$. Identities are given by η, and composition of g and f in $\mathbf{C}_T$ is given by $\mu \circ T(g) \circ f$.

There is a forgetful functor $\mathbf{C}_T \to \mathbf{C}$ given by $A \mapsto T(A)$ on objects and $f \mapsto$

$\mu \circ T(f)$ on morphisms. It has a left adjoint $\mathbf{C} \to \mathbf{C}_T$ given by $A \mapsto A$ on objects and $f \mapsto \eta \circ f$ on morphisms.

We now show that for Frobenius monads the Kleisli construction preserves daggers.

Lemma 5.2 *If T is a Frobenius monad on a dagger category $\mathbf{C}$, then $\mathbf{C}_T$ carries a dagger that commutes with the canonical functors $\mathbf{C}_T \to \mathbf{C}$ and $\mathbf{C} \to \mathbf{C}_T$.*

Proof. A straightforward calculation establishes that

$$\big(A \xrightarrow{f} T(B)\big) \;\mapsto\; \big(B \xrightarrow{\eta} T(B) \xrightarrow{\mu^\dagger} T^2(B) \xrightarrow{T(f^\dagger)} T(A)\big)$$

is a dagger on $\mathbf{C}_T$ commuting with the canonical functors $\mathbf{C} \to \mathbf{C}_T$ and $\mathbf{C}_T \to \mathbf{C}$.□

The following theorem proves a converse of the previous lemma, under the natural condition that the "reverse identity morphisms" of the Kleisli category equal their own dagger. This gives another characterization of Frobenius monads, in terms of reversibility of their effectful computations.

Theorem 5.3 *A monad T on a dagger category $\mathbf{C}$ is a Frobenius monad if and only if $\mathbf{C}_T$ has a dagger such that:*

- *the functors $\mathbf{C} \to \mathbf{C}_T$ and $\mathbf{C}_T \to \mathbf{C}$ are dagger functors;*
- *the morphisms $\mu_A^\dagger \colon T(A) \to T^2(A)$ of $\mathbf{C}$ are self-adjoint when regarded as morphisms $T(A) \to T(A)$ of $\mathbf{C}_T$.*

Proof. One direction follows from Lemma 5.2 and the observation that with that dagger the morphism $\mu_A^\dagger \colon T(A) \to T^2(A)$ is self-adjoint in $\mathbf{C}_T$. For the other direction, we wish to show that the following diagram commutes for arbitrary A.

$$\begin{array}{ccc} T^2(A) & \xrightarrow{T(\mu_A^\dagger)} & T^3(A) \\ {\scriptstyle \mu^\dagger_{T(A)}}\downarrow & & \downarrow{\scriptstyle \mu_{T(A)}} \\ T^3(A) & \xrightarrow[T(\mu_A)]{} & T^2(A) \end{array}$$

Write $\mathbf{C} \xrightarrow{F} \mathbf{C}_T$ and $\mathbf{C}_T \xrightarrow{G} \mathbf{C}$ for the canonical functors. Note that if we consider $\mathrm{id}_{T^2(A)}$ and $\eta_{T(A)} \circ \mu_A$ as morphisms of $\mathbf{C}_T$, then we have $G(\mathrm{id}_{T^2(A)}) = \mu_{T(A)}$, $G(\eta_{T(A)} \circ \mu_A) = T(\mu_A)$, and $F(\mu_A) = \eta_{T(A)} \circ \mu_A$. As G is a dagger functor, we have found preimages of all the morphisms in the diagram. More explicitly, we know that

$$\begin{aligned} G\big(\mathrm{id}_{T^2(A)} \circ F(\mu_A^\dagger)\big) &= \mu_{T(A)} \circ T(\mu_A^\dagger), \\ G\big(F(\mu_A) \circ \mathrm{id}^\dagger_{T^2(A)}\big) &= T(\mu_A) \circ \mu^\dagger_{T(A)}. \end{aligned}$$

Hence it suffices to show $\mathrm{id}_{T^2(A)} \circ F(\mu_A^\dagger) = F(\mu_A) \circ \mathrm{id}^\dagger_{T^2(A)}$. As the left hand side is the dagger of the right hand side and $\mu^\dagger$ is self-adjoint in $\mathbf{C}_T$, it suffices to show

that either equals $\mu^\dagger$. The following calculation does this for the left-hand side:

$$\begin{aligned}\mathrm{id}_{T^2(A)} \circ F(\mu_A^\dagger) &= \mu_{T(A)} \circ T(\mathrm{id}_{T^2(A)}) \circ \eta_{T^2(A)} \circ \mu_A^\dagger \\ &= \mu_{T(A)} \circ \eta_{T^2(A)} \circ \mu_A^\dagger = \mu_A^\dagger\end{aligned}$$

This completes the proof. □

Kleisli categories of commutative monads on symmetric monoidal categories are again symmetric monoidal [10]. This extends to the reversible setting.

Theorem 5.4 *If T is a commutative strong Frobenius monad on a symmetric monoidal dagger category $\mathbf{C}$, then $\mathbf{C}_T$ is a symmetric monoidal dagger category.*

Proof. The monoidal structure on $\mathbf{C}_T$ is given by $A \otimes_T B = A \otimes B$ on objects and by $f \otimes_T g = \mathrm{dst} \circ (f \otimes g)$ on morphisms. The coherence isomorphisms of $\mathbf{C}_T$ are images of those in $\mathbf{C}$ under the functor $\mathbf{C} \to \mathbf{C}_T$. This functor preserves daggers and hence unitaries, making all coherence isomorphisms of $\mathbf{C}_T$ unitary. It remains to check that the dagger on $\mathbf{C}_T$ satisfies $(f \otimes_T g)^\dagger = f^\dagger \otimes_T g^\dagger$. By Theorem 4.9, T is isomorphic to $- \otimes T(I)$, and it is straightforward to check that this induces an isomorphism between the respective Kleisli categories that preserves daggers and monoidal structure on the nose. Thus it suffices to check that this equation holds on $\mathbf{C}_{-\otimes T(I)}$, which can be done with a straightforward graphical argument. □

6 Frobenius–Eilenberg–Moore algebras

The other canonical standard categorical construction when given a monad T is to consider the category $\mathbf{C}^T$ of its Eilenberg–Moore algebras. In monadic programming, these are understood to expand effectful computations to pure computations [25]. This section identifies the largest full subcategory of $\mathbf{C}^T$ that is still reversible.

Definition 6.1 An *Eilenberg–Moore algebra* (A, a) for a monad T is a morphism $T(A) \xrightarrow{a} A$ satisfying $a \circ T(a) = a \circ \mu$ and $a \circ \eta = \mathrm{id}$. A *morphism of Eilenberg–Moore algebras* $(A, a) \to (B, b)$ is a morphism $A \xrightarrow{f} B$ satisfying $b \circ T(f) = f \circ a$. These form a category $\mathbf{C}^T$.

We will again need cooperation of such algebras with daggers when present.

Definition 6.2 Let T be a monad on a dagger category $\mathbf{C}$. A *Frobenius–Eilenberg–Moore algebra*, or *FEM-algebra* for short, is an Eilenberg–Moore algebra (A, a) that makes the following diagram commute.

$$\begin{array}{ccc} T(A) & \xrightarrow{T(a)^\dagger} & T^2(A) \\ {\scriptstyle \mu^\dagger}\downarrow & & \downarrow{\scriptstyle \mu} \\ T^2(A) & \xrightarrow[T(a)]{} & T(A) \end{array} \tag{5}$$

We call this the *Frobenius law* for Eilenberg–Moore algebras.

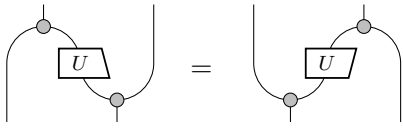
Example 6.3 The Kleisli category $\mathbf{C}_T$ of any monad T sits inside $\mathbf{C}^T$ as the *free algebras* $(T(A), \mu_A)$. If T is a Frobenius monad on a dagger category $\mathbf{C}$, any free algebra is an FEM-algebra.

Proof. The Frobenius law for the free algebra is the Frobenius law of the monad. □

There are many EM-algebras that are not FEM-algebras; a family of examples can be derived from [41, Theorem 6.4]. Here is a concrete example.

Example 6.4 Let $A = \mathbb{M}_2(\mathbb{C})$ be the Hilbert space of 2-by-2-matrices, with inner product $\langle a, b\rangle = \frac{1}{2}\operatorname{Tr}(a^\dagger \circ b)$. Matrix multiplication gives a map $m\colon A \otimes A \to A$ making A a Frobenius monoid in **FHilb**, so that $T = - \otimes A$ is a Frobenius monad. Let $u \in A$ be a unitary matrix, and define $U\colon A \to A$ by $U(a) = u^\dagger \circ a \circ u$. Now $U^\dagger(a) = u \circ a \circ u^\dagger$ and U is an endomorphism of the monoid A, making $h = m \circ (\mathrm{id} \otimes U)$ an EM-algebra. It is an FEM-algebra if and only if $u = u^\dagger$.

Proof. The Frobenius law (5) means:

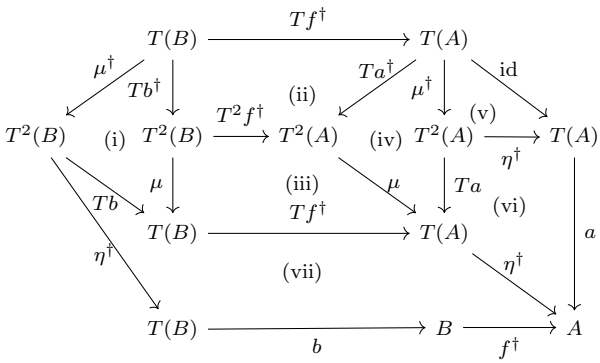


This comes down to $U = (U^*)^\dagger$, that is, $u = u^\dagger$. □

The following two results highlight the importance of FEM-algebras to daggers. First, extending from pure computations to FEM–computations is still reversible.

Proposition 6.5 *Let T be a Frobenius monad on a dagger category $\mathbf{C}$. The dagger on $\mathbf{C}$ induces a dagger on the category of FEM-algebras of T.*

Proof. Let $f\colon (A, a) \to (B, b)$ be a morphism of FEM-algebras; we have to show that $f^\dagger$ is a morphism $(B, b) \to (A, a)$. It suffices to show that $b \circ T(f) = f \circ a$ implies $a \circ T(f^\dagger) = f^\dagger \circ b$. Consider the following diagram:



Region (i) is the Frobenius law of (B, b); commutativity of (ii) follows from the assumption that f is a morphism $(A, a) \to (B, b)$ by applying T and $\dagger$; (iii) is naturality of μ; (iv) is the Frobenius law of (A, a); (v) commutes since T is a comonad; (vi) and (vii) commute by naturality of $\eta^\dagger$. □

Second, FEM–computations are the largest class that stays reversible.

Theorem 6.6 *FEM-algebras form the largest full subcategory of* $\mathbf{C}^T$ *containing* $\mathbf{C}_T$ *that carries a dagger commuting with the forgetful functor* $\mathbf{C}^T \to \mathbf{C}$.

Proof. Suppose that an EM-algebra (A, a) is such that for any free algebra $(T(B), \mu_B)$ and any morphism $f : T(B) \to A$, f is a morphism of EM-algebras $(T(B), \mu_B) \to (A, a)$ iff $f^\dagger$ is a morphism $(A, a) \to (T(B), \mu_B)$ of EM-algebras. Now (A, a) being an EM-algebra implies that a is a morphism $(T(A), \mu_A) \to (A, a)$. Thus by assumption $a^\dagger$ is a morphism $(A, a) \to (T(A), \mu_A)$, which implies that (A, a) is an FEM-algebra. □

7 Quantum measurement

This final section exemplifies the relevance of FEM-algebras to quantum computation, by indicating how quantum measurement fits neatly in effectful functional programming as *handlers* of Frobenius monads [42,26].

Example 7.1 Let B be a finite-dimensional Hilbert space. A choice of orthonormal basis makes B a commutative Frobenius monoid in **FHilb** via Example 3.2. Hence $T = - \otimes B$ is a (commutative strong) Frobenius monad on **FHilb** by Theorem 4.9.

Traditionally, effectful computations are modelled as morphisms in the Kleisli category [36,51]. In the above example, those are just morphisms $A \to A \otimes B$ in **FHilb**. Quantum measurements are indeed morphisms of this type, but they satisfy more requirements, such as von Neumann's *projection postulate*: repeating a measurement is equivalent to copying the outcome of the first measurement. These requirements make the dagger of the morphism $A \to A \otimes B$ precisely an FEM-algebra, see [8, Theorems 1.5 and 1.6]. [5] The following proposition summarizes.

Proposition 7.2 *Quantum measurements with outcomes modeled by a commutative strong Frobenius monad on* **FHilb** *correspond precisely to its FEM-algebras.* □

Consider the exception monad T that adds exceptions from a set E to a computation by $T(A) = A + E$. Intercepting exceptions means executing a computation f_e for each $e \in E$, and a computation f if no exception is raised. Thus a *handler* for T specifies an EM-algebra (A, a) and a map $f \colon A \to A$ making the triangle left

[5] Technically, the monad has to be lifted to a category of so-called completely positive maps, see [8].

below commute.

This extends to arbitrary algebraic effects T [42]. In particular, it makes sense for quantum measurement, as in the right diagram above. The Frobenius monad $-\otimes B$ modeling quantum measurement with outcomes in B is similar to 'raising exceptions B', the vertical arrows are Kleisli morphisms, and the lower right *handling construct* is an FEM-algebra $A \otimes B \xrightarrow{a} A$ that 'handles exceptions B'; it involves the unique dashed arrow, that is induced by the free property of the Kleisli algebra $A \otimes B$, and is a morphism of FEM-algebras by Example 6.3. Intuitively, Kleisli morphisms $A \to T(B)$ are constructors that 'build' an effectful computation, whereas FEM algebras $T(B) \to B$ are destructors that 'handle' the effects.

Thus in general, effectful reversible computation takes place in the category of FEM-algebras of a Frobenius monad, rather than its subcategory of Kleisli algebras. See also [21] for a similar reasoning in different language.

8 Conclusion

We have proposed Frobenius monads as the appropriate notion to model computational effects in the reversible setting of dagger categories. We have justified their definition from first principles, characterized them internally, shown that their Kleisli categories are again reversible, and identified the largest reversible subcategory of their Eilenberg–Moore categories. As an example we phrased quantum measurement in the category of such Frobenius–Eilenberg–Moore algebras.

More examples should be studied. Specifically, noncommutative Frobenius monoids on **FHilb** might induce monads modelling partial quantum measurement. Also, the relationship between nondeterministic computation in **Rel** and groupoids should be explored. Finally, we leave probabilistic computation to future work.

References

[1] Abramsky, S., *Abstract scalars, loops, and free traced and strongly compact closed categories*, Algebra and Coalgebra in Computer Science (2005), p. 2005.

[2] Abramsky, S. and B. Coecke, *A categorical semantics of quantum protocols*, in: *Logic in Computer Science 19* (2004), pp. 415–425.

[3] Altenkirch, T. and J. Grattage, *A functional quantum programming language*, in: *Logic in Computer Science* (2005), pp. 249–258.

[4] Altenkirch, T. and A. S. Green, "Semantics Techniques in Quantum Computation," Cambridge University Press, 2010 pp. 173–205.

[5] Barr, M. and C. Wells, "Category Theory for Computing Science," Prentice–Hall, 1990.

[6] Bowman, W. J., R. P. James and A. Sabry, *Dagger traced symmetric monoidal categories and reversible programming*, Reversible Computation (2011).

[7] Coecke, B., E. O. Paquette and D. Pavlovic, "Semantic Techniques in Quantum Computation," Cambridge University Press, 2009 pp. 29–69.

[8] Coecke, B. and D. Pavlovic, "Quantum measurements without sums," Taylor and Francis, 2008 pp. 559–596.

[9] Coecke, B., D. Pavlović and J. Vicary, *A new description of orthogonal bases*, Mathematical Structures in Computer Science **23** (2012), pp. 555–567.

[10] Day, B., *On closed category of functors II*, in: *Sydney Category Theory Seminar*, number 420 in Lecture Notes in Mathematics, 1974, pp. 20–54.

[11] de Vos, A. and S. de Baerdemacker, *Matrix calculus for classical and quantum circuits*, ACM Journal on Emerging Technologies in Computing Systems **11** (2014), p. 9.

[12] Di Pierro, A., C. Hankin and H. Wiklicky, *Quantitative relations and approximate process equivalences*, in: *CONCUR 14*, Lecture Notes in Computer Science **2761**, 2003, pp. 508–522.

[13] Di Pierro, A. and H. Wiklicky, *Operator algebras and the operational semantics of probabilistic languages*, in: *MFCSIT 3*, Electronic Notes in Theoretical Computer Science **161**, 2006, pp. 131–150.

[14] Green, A. S., P. L. Lumsdaine, N. J. Ross, P. Selinger and B. Valiron, *A scalable quantum programming language*, ACM SIGPLAN Notices **48** (2013), pp. 333–342.

[15] Hasuo, I. and N. Hoshino, *Semantics of higher-order quantum computation via geometry of interaction*, Logic in Computer Science (2011), pp. 237–246.

[16] Heunen, C., "Categorical quantum models and logics," Ph.D. thesis, Radboud University Nijmegen (2009).

[17] Heunen, C., *An embedding theorem for Hilbert categories*, Theory and Applications of Categories **22** (2009), pp. 321–344.

[18] Heunen, C., I. Contreras and A. Cattaneo, *Relative Frobenius algebras are groupoids*, Journal of Pure and Applied Algebra **217** (2013), pp. 114–124.

[19] Heunen, C. and J. Vicary, "Categories for Quantum Theory: An Introduction," Oxford University Press, 2015.

[20] Jacobs, B., *Involutive categories and monoids, with a GNS-correspondence*, Foundations of Physics **42** (2012), pp. 874–895.

[21] Jacobs, B., *On block structures in quantum computation*, in: *Mathematical Foundations of Program Semantics*, Electronic Notes in Theoretical Computer Science **298**, 2013, pp. 233–255.

[22] Jacobs, B. and J. Rutten, *A tutorial on (co)algebras and (co)induction*, EATCS Bulletin **62** (1997), pp. 222–259.

[23] Jacobs, B. P. F., *Semantics of weakening and contraction*, Annals of Pure and Applied Logic **69** (1994), pp. 73–106.

[24] Jacobs, B. P. F., *Coalgebraic walks, in quantum and Turing computation*, FoSSaCS, Lecture Notes in Computer Science **6604** (2011), pp. 12–26.

[25] Jacobs, B. P. F., C. Heunen and I. Hasuo, *Categorical semantics for arrows*, Journal of Functional Programming **19** (2009), pp. 403–438.

[26] Kammar, O., S. Lindley and N. Oury, *Handlers in action*, in: *International Conference on Functional Programming XVIII* (2013), pp. 145–148.

[27] Kelly, G. M. and M. L. Laplaza, *Coherence for compact closed categories*, Journal of Pure and Applied Algebra **19** (1980), pp. 193–213.

[28] Keyl, M., *Fundamentals of quantum information theory*, Physical Reports **369** (2002), pp. 431–548.

[29] Kock, A., *Strong functors and monoidal monads*, Archiv der Mathematik **23** (1972), pp. 113–120.

[30] Kozen, D., *Semantics of probabilistic programs*, Journal of Computer and System Sciences **22** (1981), pp. 328–350.

[31] Lambek, J. and P. Scott, "Introduction to higher order categorical logic," Cambridge University Press, 1986.

[32] Lauda, A., *Frobenius algebras and ambidextrous adjunctions*, Theory and Applicatoins of Categories **16** (2006), pp. 84–122.

[33] Lindner, H., *Adjunctions in monoidal categories*, Manuscripta Mathematica **26** (1978), pp. 123–139.

[34] Marsden, D., *Category theory using string diagrams*, arXiv:1401.7220 (2014).

[35] Mislove, M., J. Ouaknine, D. Pavlovic and J. Worrell, *Duality for labelled Markov processes*, in: *FOSSACS 7*, Lecture Notes in Computer Science **2987**, 2004, pp. 393–407.

[36] Moggi, E., *Notions of computation and monads*, Information and Computation **93** (1991), pp. 55–92.

[37] Moshier, M. A. and D. Petrişan, *A duality theorem for real C*-algebras*, in: *CALCO 3*, Lecture Notes in Computer Science **5728**, 2009, pp. 284–299.

[38] Panangaden, P., "Labelled Markov processes," Imperial College Press, 2009.

[39] Pavlović, D., *Quantum and classical structures in nondeterministic computation*, in: P. B. et al., editor, *Third International symposium on Quantum Interaction*, Lecture Notes in Artificial Intelligence **5494** (2009), pp. 143–157.

[40] Pavlovic, D., *Relating toy models of quantum computation: comprehension, complementarity and dagger mix autonomous categories*, in: *Quantum Physics and Logic VI*, Electronic Notes in Theoretical Computer Science **270**, 2009, pp. 121–139.

[41] Pavlovic, D., *Geometry of abstraction in quantum computation*, in: *Classical and Quantum Information Assurance Foundations and Practice*, number 09311 in Dagstuhl Seminar Proceedings, 2010.

[42] Plotkin, G. D. and M. Pretnar, *Handling algebraic effects*, Logical Methods in Computer Science **9** (2013), p. 23.

[43] Saheb-Djahromi, N., *Cpo's of measure for nondeterminism*, Theoretical Computer Science **12** (1980), pp. 19–37.

[44] Selinger, P., *A survey of graphical languages for monoidal categories*, number 813 in Lecture Notes in Physics (2009), pp. 289–356.

[45] Selinger, P. and B. Valiron, *A lambda calculus for quantum computation with classical control*, Mathematical Structures in Computer Science **16** (2006), pp. 527–552.

[46] Street, R., *Frobenius monads and pseudomonoids*, Journal of Mathematical Physics **45** (2004), pp. 3930–3948.

[47] Szabo, M. E., "Algebra of Proofs," Number 88 in Studies in Logic and the Foundations of Mathematics, North-Holland, 1978.

[48] Toffoli, T., *Reversible computing*, Automata, Languages and Programming **85** (1980), pp. 632–644.

[49] Univalent Foundations Program, T., "Homotopy Type Theory: Univalent Foundations of Mathematics," http://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

[50] Vicary, J., *Categorical formulation of quantum algebras*, Communications in Mathematical Physics **304** (2011), pp. 765–796.

[51] Wadler, P., *Comprehending monads*, Mathematical Structures in Computer Science **2** (1992), pp. 461–493.

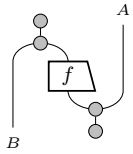
[52] Wolff, H., *Monads and monoids on symmetric monoidal closed categories*, Archiv der Mathematik **24** (1973), pp. 113–120.

A Proofs

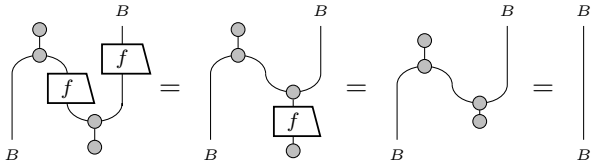
This appendix verifies steps used in proofs in Section 4.

Lemma A.1 *A monoid homomorphism between Frobenius monoids in a monoidal dagger category, that is also a comonoid homomorphism, is an isomorphism.*

Proof. Construct an inverse to $A \xrightarrow{f} B$ as follows:



The composite with f gives the identity in one direction:



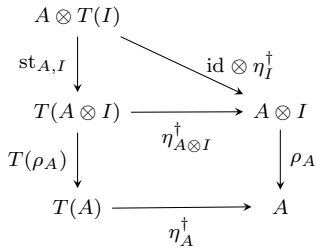
The third equality uses the Frobenius law (1) and unitality. The other composite is the identity by a similar argument. □

Lemma A.2 *The functor $T \mapsto T(I)$ preserves the Frobenius law.*

Proof. Consider the diagram in Fig. A.1. Region (i) commutes because T is a Frobenius monad, (ii) because $\mu^\dagger$ is natural, (iii) because ρ^{-1} is natural, (iv) because $\mathrm{st}^\dagger$ is natural, (v) is a consequence of T being a strong monad, (vi) commutes as ρ is natural, (vii) and (viii) because st is natural, (ix) commutes trivially and (x) because st is natural. Regions (ii)'-(x)' commute for dual reasons. Hence the outer diagram commutes, and $T \mapsto T(I)$ preserves the Frobenius law. □

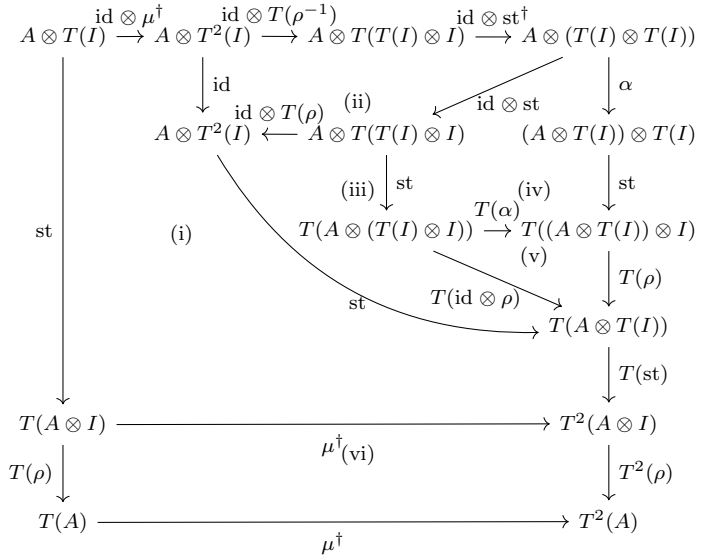
Lemma A.3 *If T is a strong Frobenius monad, the counit of the adjunction of Proposition 4.5 is a morphism of comonads.*

Proof. First we show that the counit of the adjunction preserves counits of the comonads. It suffices to see that



commutes. But the rectangle commutes because $\eta^\dagger$ is natural, and the triangle commutes because T is a strong monad and st is an isomorphism.

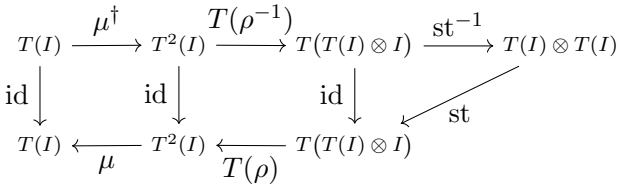
To see that that the counit of the adjunction preserves the comultiplication, consider the following diagram:



Commutativity of region (i) is a consequence of T being a strong monad, and st being an iso, (ii) commutes by definition, (iii) commutes as st is natural, (iv) because T is a strong functor, (v) by coherence and finally (vi) by naturality of $\mu^\dagger$. Hence the outer diagram commutes, and the counit of the adjunction preserves the comultiplication. □

Corollary A.4 *The equivalence of Theorem 4.9 restricts to an equivalence between special Frobenius monoids and special strong Frobenius monads.*

Proof. The commutative case follows from [52]. If the Frobenius monoid is special, so is the monad, by trivial graphical manipulation of Example 4.2. Conversely, if the Frobenius monad T is special, the following diagram commutes:



and so $T(I)$ is special. □

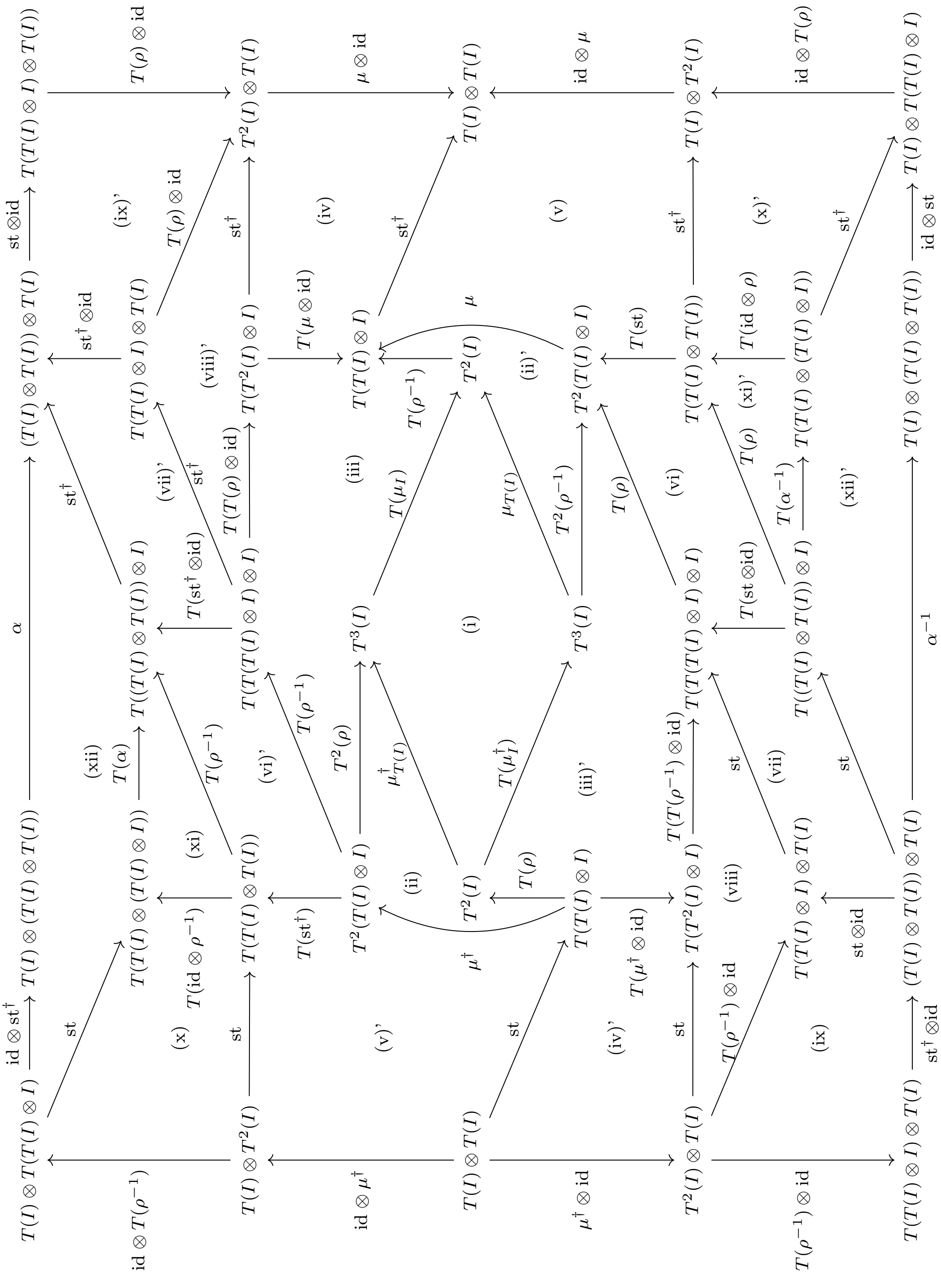


Fig. A.1. Diagram proving that $T \mapsto T(I)$ preserves the Frobenius law.