QUANTUM INFORMATION EFFECTS

CHRIS HEUNEN* AND ROBIN KAARSGAARD†

Abstract. We study the two dual quantum information effects to manipulate the amount of information in quantum computation: hiding and allocation. The resulting type-and-effect system is fully expressive for irreversible quantum computing, including measurement. We provide universal categorical constructions that semantically interpret this arrow metalanguage with choice, starting with any rig groupoid interpreting the reversible base language. Several properties of quantum measurement follow in general, and we translate quantum flow charts into our language. The semantic constructions turn the category of unitaries between Hilbert spaces into the category of completely positive trace-preserving maps, and they turn the category of bijections between finite sets into the category of functions with chosen garbage. Thus they capture the fundamental theorems of classical and quantum reversible computing of Toffoli and Stinespring.

1. Introduction

Something is rotten in the state of quantum computing. It subsumes classical computing, which is generally irreversible, yet it is most often formulated as a reversible quantum circuit, with an irreversible quantum measurement as an afterthought. The conceptual status of this irreversible measurement remains mysterious. This is known as the measurement problem.

Classical computing itself is most often formulated as composed of irreversible operations. However, by the seminal works of Toffoli [**?**] and Bennett [**?**], and more recently by James and Sabry [**?**], we know that it can also be phrased in terms of reversible operations, as long as we consider systems to be open and interact with an environment that is eventually disregarded. This final part is important, as reversible computations alone (be they classical or quantum) cannot change the *amount* of information (as measured by an appropriate notion of *entropy*). To understand the nature of quantum measurement in computation requires us to close two conceptual gaps:

(i) from reversible classical computing to reversible quantum computing; and
(ii) from reversible quantum computing to irreversible quantum computing.

It may seem that much has to be added to a reversible language to make it suitable for quantum computing. Similarly, it may seem that much less can be expressed in purely reversible quantum computations than in arbitrary quantum computations with measurements. We argue, however, that both gaps are smaller than they may appear.

To do this, we start with the reversible combinator language Π, which governs classical reversible computation, and extend it with combinators for quantum phases and the quantum *Hadamard gate*. We call the result $\mathcal{U}\Pi$ ("yuppie"), because it is already approximately universal for reversible quantum computing with unitary gates.

To address (ii) we introduce two *quantum information effects* – computational effects that manipulate the amount of information – through two arrows [**?**]. The first computational effect allows *allocation* of auxiliary space on a hidden heap, leading to the arrow metalanguage $\mathcal{U}\Pi_a$ ("yuppie-a"). This calculus is approximately universal for quantum computing with *isometries* rather than unitaries. The second computational effect dually allows *hiding*, by disregarding specifically marked *garbage output*, leading to $\mathcal{U}\Pi_a^\chi$ ("yuppie-chi-a"). We prove that this calculus is approximately universal for *arbitrary quantum computations*, including measurement.

Thus we have an arrow metalanguage that, with two simple computational effects on top of a pure reversible model, is fully expressive for irreversible quantum computing. All the allocation and hiding is tracked by the type system, and so allows us to compile an irreversible quantum program into an explicit reversible quantum circuit.

* Supported by the Engineering and Physical Sciences Research Council Fellowship No. EP/R044759/1.
† Supported by the Independent Research Fund Denmark under DFF-International Postdoc Fellowship No. 0131-00025B.

We provide matching categorical semantics via surprisingly simple concrete constructions that have very general universal properties. Vanilla Π may be interpreted in rig categories: categories with two monoidal structures $(\otimes, \oplus)$, where the product $(\otimes)$ distributes over the sum $(\oplus)$. We will interpret $\mathcal{U}\Pi$ in the category **Unitary** of Hilbert spaces and unitaries. This is a choice of canonical model: all that is needed is a rig category with morphisms to interpret phase gates and the Hadamard gate (which we will see is equivalent to having a notion of *superposition*).

For $\mathcal{U}\Pi_a$ we provide a free construction that turns a rig category $\mathbf{C}$ into a new one $R[\mathbf{C}]$ where the unit for the sum becomes initial. Then $R[\mathbf{Unitary}]$ is the category **Isometry** of Hilbert spaces and isometries. Dually, we interpret $\mathcal{U}\Pi_a^\chi$ via a free construction L making a monoidal unit terminal. Now $L[\mathbf{Isometry}]$ is the category **CPTP** of arbitrary (irreversible) quantum channels. Classically, R transforms the category of bijections between sets into that of injections, and in turn L transforms that into arbitrary functions with chosen garbage. This lets us reformulate Toffoli's *fundamental theorem of reversible computing* [**?**] as a purely categorical statement.

Surprising mileage is obtained from these simple constructions, as we prove in general several properties of measurement that can be expressed entirely as semantic equivalences between program fragments. For example, we show that measurement commutes with injections and projections, and that measurement is idempotent. More generally, we show that our setting can interpret (noniterative) *quantum flow charts* [**?**]. All constructions and translations in this paper are formalised in (heavily extended) Glasgow Haskell (see `https://github.com/rkaarsgaard/upi`)

Related work. Classical information effects are due to [**?**]. Quantum programming languages are an active research topic [**?**, **?**, **?**, **?**, **?**]). In particular, quantum measurement has been studied extensively as a computational effect [**?**, **?**, **?**, **?**, **?**]) too. While proven practically useful, the precise meaning of *measurement-as-an-effect* has remained unclear, perhaps partly because of the wide-spread conscription to the view "quantum data, classical control" [**?**]. This work provides the missing origin story, by showing that measurement-as-an-effect arises through a sequence of arrow constructions that can be applied (and given precise meaning) to *any* rig groupoid. Thus our categorical constructions eliminate the need for involved functional-analytic semantics using operator algebras [**?**, **?**, **?**], and is much more general than earlier work specific to Hilbert spaces [**?**, **?**] and restriction categories [**?**].

Overview. Section **??** recalls background material. Next, Section **??** discusses Π, introduces the languages $\mathcal{U}\Pi$, $\mathcal{U}\Pi_a$, and $\mathcal{U}\Pi_a^\chi$, and proves expressivity theorems. Section **??** deals with categorical semantics: it recalls rig groupoids, introduces the L and R constructions, and proves that they respect $\otimes$ and give the appropriate setting to interpret $\oplus$ as an arrow with choice. Section **??** concerns universal properties of the L and R constructions, and shows that they encompass two fundamental results: Toffoli's fundamental theorem, and Stinespring's dilation theorem. In Section **??** we derive extra properties in the arrow metalanguage, generically in $L[R[\mathbf{C}]]$. Finally, Section **??** concludes and lists interesting directions for future work. Several proofs are relegated to the Supplementary Material.

2. Background

This section recalls the basics of quantum theory, monoidal categories, and information effects.

2.1. Quantum theory.

For more details we refer to [**?**, **?**].

2.1.1. *Pure quantum theory and bra-ket notation.* A quantum system is captured by a complex Hilbert space H. For example, qubits are modelled by $H = \mathbb{C}^2$. The pure states ψ range over the vectors in H of unit norm: $\|\psi\| = 1$. By convention, vectors are denoted as a *ket* $|\psi\rangle$. This is handy, because then the functional $H \to \mathbb{C}$ that maps ϕ to the inner product $\langle\psi|\phi\rangle$ can be denoted as the *bra* $\langle\psi|$. Pure dynamics of a quantum system are reversible. Evolution is given by a *unitary* linear map $U\colon H \to H$, meaning that U is a bijection that is *isometric*: $\langle U\phi|U\psi\rangle = \langle\phi|\psi\rangle$. More generally, any continuous linear function $f\colon H \to K$ has an *adjoint* $f^\dagger\colon K \to H$ satisfying $\langle f\phi|\psi\rangle = \langle\phi|f^\dagger\psi\rangle$. An isometry then satisfies $f^\dagger \circ f = \mathrm{id}$, and a unitary furthermore satisfies $f \circ f^\dagger = \mathrm{id}$.

Pure quantum theory subsumes reversible classical computation. Any finite set I generates a Hilbert space $\mathbb{C}^I$ of linear combinations of elements of I. Thus $\{|i\rangle \mid i \in I\}$ forms a basis of $\mathbb{C}^I$ that is moreover orthonormal: $\langle i|j\rangle$ is 1 when $i = j$ and vanishes otherwise. We call this basis of $\mathbb{C}^n$ induced by the set $\{1, 2, \ldots, n\}$ the *computational basis*. Any bijection of $\{1, 2, \ldots, n\}$ induces a unitary on $\mathbb{C}^n$ that preserves the computational basis.

If two quantum systems are modelled by Hilbert spaces H and K, the compound system is given by their tensor product $H \otimes K$. For example, a 3-qubit system is modelled by $\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 \simeq \mathbb{C}^8$. Similarly, if H and K evolve along unitaries U and V, then $H \otimes K$ evolves along $U \otimes V$.

2.1.2. *Mixed quantum theory.* A quantum computation (in the quantum circuit model) consists of composition of tensor products of unitary gates, which is entirely reversible. However, reading out the result of the computation requires a measurement, which is an irreversible operation. The standard model therefore considers *mixed states*. These are given by a *density matrix*, which is a linear function $\rho\colon H \to H$ such that $0 \leq \langle \rho(\psi)|\psi\rangle \leq 1$ for all $|\psi\rangle$. Thus any pure state $|\psi\rangle \in H$ is also a mixed state $|\psi\rangle\langle\psi| : H \to H$.

Mixed states no longer have reversible dynamics. Any unitary $U\colon H \to H$ still induces a map that takes a mixed state ρ to a mixed state $U^\dagger \circ \rho \circ U$. But now the appropriate dynamics allow more possibilities, generally given by so-called *completely positive trace-preserving (CPTP)* maps, also known as *quantum channels*. It is not important here to set out their definition. What is important is *Stinespring's dilation theorem*, which says that any CPTP map $H \to K$ may be factored as a pure evolution $H \to H \otimes G$ followed by a map $H \otimes G \to K$ given by $\rho \mapsto V^\dagger \circ \rho \circ V$ for an isometry V. That is, irreversible (mixed) quantum theory is contained within reversible (pure) quantum theory, as long as you allow an environment to play the role of auxiliary state space but disregard it.

2.1.3. *Superposition and measurement.* Superposition is the ability of a quantum state $|\rho\rangle$ to occupy several classical states $|b_1\rangle \dots |b_n\rangle$ at once, so long as no measurement occurs. Each classical state in a superposed state is weighted by a complex number α_i known as an *amplitude*. Once a system in superposition $|\psi\rangle = \sum_{i=1}^n \alpha_i |b_i\rangle$ is measured, it collapses to one of the classical states $|b_k\rangle$. The outcome of such a measurement is probabilistic, with the probability of observing $|b_i\rangle$ given by $|\langle b_i|\psi\rangle|^2$; this is called the *Born rule*. Using density matrices, measurement with respect to $\{|b_i\rangle\}_{i\in I}$ is represented by the *measurement instrument channel*

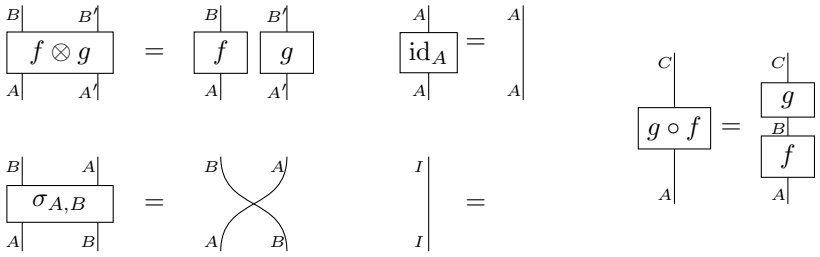
$$\rho \mapsto \sum_{i\in I} |b_i\rangle\langle b_i|\, \rho\, |b_i\rangle\langle b_i|$$

that sends quantum states to their post-measurement (mixed classical) states. For example, measuring a qubit $|\psi\rangle\langle\psi|$ for $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ with respect to $\{|0\rangle, |1\rangle\}$ results in the mixed state $|\alpha|^2 |0\rangle\langle 0| + |\beta|^2 |1\rangle\langle 1|$. The decay of a quantum state to a mixed classical state is known as *decoherence*, so this measurement process is sometimes called *decohering measurement*.

2.1.4. *Global and relative phase.* Recall that the complex conjugate of a complex number $\varphi = a + bi$ is $\overline{\varphi} = a - bi$. A *phase* is a complex number satisfying $\varphi \cdot \overline{\varphi} = 1$; equivalently, φ has norm 1. Quantum states that differ only by a *global phase*, $|\psi'\rangle = \varphi|\psi\rangle$, are indistinguishable, in that they have the same measurement statistics. But the phase difference between *parts* of states can be incredibly important. For example, the states $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ differ only by the phase -1 in their amplitude for $|1\rangle$. This difference in *relative phase* makes $|+\rangle$ and $|-\rangle$ orthogonal.

2.2. **Monoidal categories.** For our semantics, we will assume that the reader is familiar with the basic notions of categories and functors [**?**]. To set notation, recall that a category $\mathbf{C}$ is symmetric monoidal when it comes equipped with a tensor product $\otimes\colon \mathbf{C} \times \mathbf{C} \to \mathbf{C}$, a unit object I, isomorphisms $\lambda_A\colon I \otimes A \to A$ and $\rho_A\colon A \otimes I \to A$ called unitors, isomorphisms $\alpha_{A,B,C}\colon A \otimes (B \otimes C) \to (A \otimes B) \otimes C$ called associators, and isomorphisms $\sigma_{A,B}\colon A \otimes B \to B \otimes A$ called symmetries for all objects A, B, C, that satisfy certain coherence laws [**?**].

A category lets one compose morphisms 'in sequence'; a monoidal category additionally lets one compose morphisms 'in parallel'. This is expressed satisfyingly in the graphical calculus for monoidal categories. We draw a morphism as a box with an incoming wire labelled by its domain and an outgoing wire labelled by its codomain. Composition becomes stacking boxes vertically, whereas we draw the tensor product of boxes side by side. In particular, objects $A \otimes B$ may be drawn as a single wire labelled $A \otimes B$, or as two parallel wires labelled by A and B, and the nullary case of a wire labelled I is simply not drawn. The special morphisms $\sigma_{A,B}$ are drawn as crossing two wires. The coherence laws simply say that one may ignore the coherence isomorphisms graphically.



Of special interested are symmetric monoidal categories whose tensor unit is initial or terminal. If there is a unique morphism $A \to I$ for any object A, the monoidal category is called *affine*, and if there is a unique morphism $I \to A$ for any object A, it is called *coaffine*.

A functor $F\colon \mathbf{C} \to \mathbf{D}$ between symmetric monoidal categories is (strong) monoidal when it is equipped with isomorphisms $F(A) \otimes F(B) \simeq F(A \otimes B)$ and $F(I) \simeq I$ that respect the coherence isomorphisms of $\mathbf{C}$ and $\mathbf{D}$. It is *strict* monoidal when these isomorphisms are in fact identities. Monoidal functors between (co)affine categories automatically preserve the terminal (initial) object.

2.3. **Information effects.** Classical computation, embodied by functions on finite sets, is irreversible, because applying a function in general loses information. This can be made precise via the Shannon entropy $H = -\sum p_i \log p_i$ that measures how surprising variable is when it takes value i with probability p_i. The functions that preserve Shannon entropy are precisely bijections. This direct connection between information preservation and reversibility is a consequence of *Landauer's principle* [**?**].

The central idea of information effects [**?**] is that this irreversible model of computation arises from a reversible (bijective) model of computation, together with computational effects that can implicitly duplicate and erase information. Thus irreversible programs are reversible instructions governed by an arrow metalanguage that tracks interaction with a global environment.

Quantum theory, embodied by quantum channels between finite-dimensional Hilbert spaces, is also irreversible. The information content of a quantum state ρ can be made precise by von Neumann entropy $S = -\mathrm{tr}(\rho \log \rho)$. In this case, the information-preserving maps are also the reversible ones: those of the form $\rho \mapsto U^\dagger \circ \rho \circ U$ for unitary U. This mirrors the classical connection between information preservation and reversibility.

3. Three generations of yuppie

Π is a reversible combinator language introduced in [**?**, **?**] to study strongly typed reversible *classical* programming. Many extensions exist, such as partiality and iteration [**?**, **?**], fractional types [**?**, **?**], negative types [**?**], and higher combinators [**?**, **?**]. This section introduces a quantum extension to Π, and shows it to be approximately universal for unitaries, the canonical model of pure quantum computation (without measurement). We then use two arrow constructions to extend this with the quantum information effects of *allocation* and *hiding* to arrive at an arrow metalanguage which we prove approximately universal for quantum channels, the canonical model of full quantum computation (with measurement).

3.1. **Reversible classical combinators: Π.** The syntax and type system of the unextended calculus Π is shown in Figure **??**. It comprises a small set of invertible, first-order, strongly typed polymorphic combinators on data constructed from (classical) sum and products types, as well as their units 0 and 1. These combinators enable data of sum and product type to be swapped (sending inl x to inr x and vice versa for sums, and (x, y) to (y, x) for products), reassociated, and have their respective units added and removed in the usual way. Products can also be distributed over sums (and back again) as usual. Finally, these combinators can be composed in sequence $c_1 \fatsemi c_2$ and in parallel using both $+$ and $\times$. That is, $c_1 \times c_2$ takes a pair (x, y) and produces the pair $(c_1\ x, c_2\ x)$, while $c_1 + c_2$ takes inl x to inl $(c_1 x)$ and inr y to inr $(c_2 y)$.

Aside from the base combinators, a pair of useful derived combinators $midswap^+ : (b_1 + b_2) + (b_3 + b_4) \leftrightarrow (b_1 + b_3) + (b_2 + b_4)$ and $midswap^\times : (b_1 \times b_2) \times (b_3 \times b_4) \leftrightarrow (b_1 \times b_3) \times (b_2 \times b_4)$ can be defined as

$$midswap^+ = assoc^+ \fatsemi (id + associ^+) \fatsemi (id + (swap^+ + id)) \fatsemi (id + assoc^+) \fatsemi associ^+$$
$$midswap^\times = assoc^\times \fatsemi (id \times associ^\times) \fatsemi (id \times (swap^\times \times id)) \fatsemi (id \times assoc^\times) \fatsemi associ^\times \quad .$$

The definition and use of derived combinators should be taken as no more than aliasing, or macro definition and expansion; in particular, (mutually) recursive systems of derived combinators are *not* permitted.

Syntax

$$b ::= 0 \mid 1 \mid b + b \mid b \times b \qquad \text{(base types)}$$

$$t ::= b \leftrightarrow b \qquad \text{(combinator types)}$$

$$\begin{aligned} a ::= {} & id \mid swap^+ \mid unit^+ \mid uniti^+ \mid assoc^+ \mid associ^+ \\ & \mid swap^\times \mid unit^\times \mid uniti^\times \mid assoc^\times \mid associ^\times \\ & \mid distrib \mid distribi \mid distribo \mid distriboi \qquad \text{(atomic combinators)} \end{aligned}$$

$$d ::= midswap^+ \mid midswap^\times \qquad \text{(derived combinators)}$$

$$c ::= a \mid c \fatsemi c \mid c + c \mid c \times c \qquad \text{(combinators)}$$

Typing rules

$$\begin{array}{rcccl} id & : & b \leftrightarrow b & : & id \\ swap^+ & : & b_1 + b_2 \leftrightarrow b_2 + b_1 & : & swap^+ \\ unit^+ & : & b + 0 \leftrightarrow b & : & uniti^+ \\ assoc^+ & : & (b_1 + b_2) + b_3 \leftrightarrow b_1 + (b_2 + b_3) & : & associ^+ \\ swap^\times & : & b_1 \times b_2 \leftrightarrow b_2 \times b_1 & : & swap^\times \\ unit^\times & : & b \times 1 \leftrightarrow b & : & uniti^\times \\ assoc^\times & : & (b_1 \times b_2) \times b_3 \leftrightarrow b_1 \times (b_2 \times b_3) & : & associ^\times \\ distrib & : & b_1 \times (b_2 + b_3) \leftrightarrow (b_1 \times b_2) + (b_1 \times b_3) & : & distribi \\ distribo & : & b \times 0 \leftrightarrow 0 & : & distriboi \\ midswap^+ & : & (b_1 + b_2) + (b_3 + b_4) \leftrightarrow (b_1 + b_3) + (b_2 + b_4) & : & midswap^+ \\ midswap^\times & : & (b_1 \times b_2) \times (b_3 \times b_4) \leftrightarrow (b_1 \times b_3) \times (b_2 \times b_4) & : & midswap^\times \end{array}$$

$$\frac{c_1 : b_1 \leftrightarrow b_2 \quad c_2 : b_2 \leftrightarrow b_3}{c_1 \fatsemi c_2 : b_1 \leftrightarrow b_3} \qquad \frac{c_1 : b_1 \leftrightarrow b_3 \quad c_2 : b_2 \leftrightarrow b_4}{c_1 + c_2 : b_1 + b_2 \leftrightarrow b_3 + b_4} \qquad \frac{c_1 : b_1 \leftrightarrow b_3 \quad c_2 : b_2 \leftrightarrow b_4}{c_1 \times c_2 : b_1 \times b_2 \leftrightarrow b_3 \times b_4}$$

FIGURE 1. The syntax and type system of Π.

$$\begin{array}{rclrcl} inv(id) & = & id & inv(swap^+) & = & swap^+ \\ inv(unit^+) & = & uniti^+ & inv(uniti^+) & = & unit^+ \\ inv(assoc^+) & = & associ^+ & inv(associ^+) & = & assoc^+ \\ inv(swap^\times) & = & swap^\times & inv(unit^\times) & = & uniti^\times \\ inv(uniti^\times) & = & unit^\times & inv(assoc^\times) & = & associ^\times \\ inv(associ^\times) & = & assoc^\times & inv(distrib) & = & distribi \\ inv(distribi) & = & distrib & inv(distribo) & = & distriboi \\ inv(distriboi) & = & distribo & inv(phase_\varphi) & = & phase_{\overline{\varphi}} \\ inv(hadamard) & = & hadamard & inv(c_1 \fatsemi c_2) & = & inv(c_2) \fatsemi inv(c_1) \\ inv(c_1 + c_2) & = & inv(c_1) + inv(c_2) & inv(c_1 \times c_2) & = & inv(c_1) \times inv(c_2) \end{array}$$

FIGURE 2. The inversion meta-combinator *inv* in $(\mathcal{U})\Pi$.

Π takes semantics in *rig groupoids* (see Section **??**), the canonical choice being the category **FinBij** of finite sets and bijective functions. Indeed, Π is universal for finite bijective functions; Figure **??** shows the implementations of the universal gate set $\{PX, CNOT, TOFFOLI\}$ [**?**].

3.1.1. *Inversion.* Our presentation of Π differs slightly from [**?**]: our syntax does not include an inversion combinator *inv c*; instead we derive it as a metacombinator (in Figure **??**). This avoids some superfluous syntax – e.g., $inv(c_1 + c_2)$ and $(inv\ c_1) + (inv\ c_2)$ are equivalent, as are $inv(id)$ and id – but results in a higher number of base combinators. Some basic well-behavedness properties can be straightforwardly shown by induction, summarised as follows.

Proposition 1. *Let c be a* $(\mathcal{U})\Pi$ *combinator. Then:*

(i) $c\colon b_1 \leftrightarrow b_2$ *implies* $inv(c)\colon b_2 \leftrightarrow b_1$*, and*
(ii) $inv(inv(c)) = c$*.*

3.2. **Reversible quantum combinators: $\mathcal{U}\Pi$.** $\mathcal{U}\Pi$ ("yuppie") extends Π with notions of phase and superposition, in the form of the $phase_\varphi$ and *hadamard* combinators. Figure **??** shows the syntax and

Syntax

$$a ::= \cdots \mid \mathit{phase}_{\varphi} \mid \mathit{hadamard} \qquad \text{(atomic combinators)}$$

Typing rules

$$\begin{array}{ccccc} \mathit{phase}_{\varphi} & : & 1 \leftrightarrow 1 & : & \mathit{phase}_{\overline{\varphi}} \\ \mathit{hadamard} & : & 1+1 \leftrightarrow 1+1 & : & \mathit{hadamard} \end{array}$$

FIGURE 3. The syntax and typing rules of $\mathcal{U}\Pi$ in addition to those in Π (see Figure **??**).

types of this small extension. While the types of $\mathcal{U}\Pi$ remain the same as in Π, in $\mathcal{U}\Pi$ they are entirely quantum rather than (as in Π) entirely classical. For example, where $1+1$ in Π is the type of *bits*, in $\mathcal{U}\Pi$ it is the type of *qubits* (with no way of forming the type of bits). $\mathcal{U}\Pi$ canonically takes semantics in the category **Unitary** of finite dimensional Hilbert spaces and unitaries. The full treatment of these semantics is given in Section **??**, but later in this section we will show that $\mathcal{U}\Pi$ is *approximately universal* for unitaries.

Phases correspond with unitaries $\mathbb{C} \to \mathbb{C}$. Since $\mathbb{C}$ is the tensor unit in **Unitary**, we can express an arbitrary phase φ through the combinator $\mathit{phase}_{\varphi} \colon 1 \leftrightarrow 1$. This will allow us to form quantum *phase gates* like S and T, and to multiply a combinator $c \colon b_1 \leftrightarrow b_2$ by an arbitary phase φ as:

$$\varphi \bullet c = \mathit{uniti}^{\times} \fatsemi c \times \mathit{phase}_{\varphi} \fatsemi \mathit{unit}^{\times}$$

We include *all* phases, yielding an uncountable number of phase combinators, even though a finite number of phases suffice for approximate universality up to a global phase. We find including all of them to be the more principled solution, especially when an appropriate finite subset (such as $\{\pm i, \pm 1, \cos(\frac{\pi}{4}) \pm i \sin(\frac{\pi}{4})\}$) can be chosen in a concrete implementation without detriment.

Superpositions are introduced by means of the *hadamard* combinator (of type $1+1 \leftrightarrow 1+1$, or $\mathit{Qbit} \leftrightarrow \mathit{Qbit}$), named after the Hadamard gate from which it takes its semantics:

$$|0\rangle \mapsto \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad\qquad |1\rangle \mapsto \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

It introduces uniform superpositions of states in the computational basis. Though effective, it can be argued that this combinator is not conceptually clean: all of the Π combinators are parametrically polymorphic and pertain to *structure* rather than *behaviour*, but *hadamard* is monomorphic, and pertains specifically to the behaviour of qubits. To mend this, we could instead introduce a parametrically polymorphic combinator $\mathit{superposition} : b + b \leftrightarrow b + b$ with semantics:

$$\mathrm{inl}(|\psi\rangle) \mapsto \tfrac{1}{\sqrt{2}}(\mathrm{inl}(|\psi\rangle) + \mathrm{inr}(|\psi\rangle)) \qquad\qquad \mathrm{inr}(|\psi\rangle) \mapsto \tfrac{1}{\sqrt{2}}(\mathrm{inl}(|\psi\rangle) - \mathrm{inr}(|\psi\rangle))$$

Now *hadamard* is just the *superposition* combinator on the type $1+1 \leftrightarrow 1+1$. Interestingly, the two are equivalent in the presence of the other combinators, as *superposition* can be derived:

$$\mathit{superposition} = (\mathit{uniti}^{\times} + \mathit{uniti}^{\times}) \fatsemi \mathit{distribi} \fatsemi (\mathit{id} \times \mathit{hadamard}) \fatsemi \mathit{distrib} \fatsemi (\mathit{unit}^{\times} + \mathit{unit}^{\times})$$

It also follows from this definition that *superposition*, like *hadamard*, is self-inverse. Whether the *hadamard* or the *superposition* combinator is taken as primal thus comes down to preference; there is no difference in expressivity, and one is easily derived from the other.

3.2.1. *Expressiveness.* We have taken an established combinator calculus for reversible *classical* computing, and extended it only slightly with two quantum combinators modelling phase and superposition. One may ask whether this extension is sufficient to express all of reversible *quantum* computing. This question contains a number of subtleties, not least because there are systems of quantum computing which do include concepts of both phase and superposition, but can nevertheless be efficiently simulated by purely classical means (e.g., the Clifford gate set *without* T).

Figure **??** shows the implementation of a variety of reversible quantum (and classical) gates in $\mathcal{U}\Pi$. Note the meta-combinator *ctrl*, which produces a combinator for the usual *controlled gate* for a combinator corresponding to a gate c. Briefly, *ctrl* c takes $(|0\rangle, |\psi\rangle)$ to $(|0\rangle, |\psi\rangle)$, and $(|1\rangle, |\psi\rangle)$ to $(|1\rangle, c(|\psi\rangle))$.

Using these representations of quantum gates, it can be shown that $\mathcal{U}\Pi$ is *approximately universal* for reversible quantum computing: it can approximate any unitary (on a space of dimension 2^n) to arbitrary precision (measured by the operator norm). The proof is in the Supplementary Material.

Theorem 2. *$\mathcal{U}\Pi$ is approximately universal for $2^n \times 2^n$ unitaries: For any unitary U and $\delta > 0$ there exists a $\mathcal{U}\Pi$ combinator u such that $\|U - [\![u]\!]\|_{\mathrm{op}} < \delta$.*

$$
\begin{array}{lcl@{\qquad}lcl@{\qquad}lcl}
px & : & Qbit \leftrightarrow Qbit & py & : & Qbit \leftrightarrow Qbit & pz & : & Qbit \leftrightarrow Qbit \\
px & = & swap^{+} & py & = & swap^{+} \fatsemi (phase_{-i} + phase_{i}) & pz & = & id + phase_{-1}
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl}
s & : & Qbit \leftrightarrow Qbit & t & : & Qbit \leftrightarrow Qbit \\
s & = & id + phase_{i} & t & = & id + phase_{e^{\frac{i\pi}{4}}}
\end{array}
$$

$$
\begin{array}{lcl}
ctrl\ c & : & b \leftrightarrow b \to Qbit \times b \leftrightarrow Qbit \times b \\
ctrl\ c & = & swap^{\times} \fatsemi distrib \fatsemi (unit^{\times} + unit^{\times}) \fatsemi (id + c) \fatsemi (uniti^{\times} + uniti^{\times}) \fatsemi distribi \fatsemi swap^{\times}
\end{array}
$$

$$
\begin{array}{lcl@{\qquad}lcl@{\qquad}lcl}
cnot & : & Qbit^2 \leftrightarrow Qbit^2 & toffoli & : & Qbit^3 \leftrightarrow Qbit^3 & fredkin & : & Qbit^3 \leftrightarrow Qbit^3 \\
cnot & = & ctrl\ px & toffoli & = & ctrl\ cnot & fredkin & = & ctrl\ swap^{\times}
\end{array}
$$

FIGURE 4. The implementation of a variety of quantum gates in $\mathcal{U}\Pi$. We use $Qbit^n$ as shorthand for the n-fold product of the qubit type $Qbit = 1 + 1$ with itself.

Syntax

$$
\begin{array}{lr}
b ::= 0 \mid 1 \mid b + b \mid b \times b & \text{(base types)} \\
t ::= b \rightarrowtail b & \text{(combinator types)} \\
c ::= lift\ u & \text{(combinator constructor)} \\
d ::= c \mid iso \mid arr\ u \mid d \ggg d \mid first\ d \mid second\ d \mid left\ d \mid right\ d & \\
\quad \mid d \ast\!\ast\!\ast d \mid d +\!+\!+ d \mid inhab \mid inl \mid inr \mid alloc \mid clone & \text{(combinators)}
\end{array}
$$

Typing rules

$$
\frac{u : b_1 + b_3 \leftrightarrow b_2}{lift\ u : b_1 \rightarrowtail b_2} \qquad \frac{u : b_1 \leftrightarrow b_2}{arr\ u : b_1 \rightarrowtail b_2} \qquad \frac{d_1 : b_1 \rightarrowtail b_2 \quad d_2 : b_2 \rightarrowtail b_3}{d_1 \ggg d_2 : b_1 \rightarrowtail b_3}
$$

$$
\frac{d : b_1 \rightarrowtail b_2}{first\ d : b_1 \times b_3 \rightarrowtail b_2 \times b_3} \qquad \frac{d : b_1 \rightarrowtail b_2}{second\ d : b_3 \times b_1 \rightarrowtail b_3 \times b_2}
$$

$$
\frac{d : b_1 \rightarrowtail b_2}{left\ d : b_1 + b_3 \rightarrowtail b_2 + b_3} \qquad \frac{d : b_1 \rightarrowtail b_2}{right\ d : b_3 + b_1 \rightarrowtail b_3 + b_2}
$$

$$
\frac{d_1 : b_1 \rightarrowtail b_3 \quad d_2 : b_2 \rightarrowtail b_4}{d_1 +\!+\!+ d_2 : b_1 + b_2 \rightarrowtail b_3 + b_4} \qquad \frac{d_1 : b_1 \rightarrowtail b_3 \quad d_2 : b_2 \rightarrowtail b_4}{d_1 \ast\!\ast\!\ast d_2 : b_1 \times b_2 \rightarrowtail b_3 \times b_4} \qquad \frac{b \text{ inhabited}}{inhab : 1 \rightarrowtail b}
$$

$$
\frac{}{alloc : 0 \rightarrowtail a} \qquad \frac{}{inl : a \rightarrowtail a + b} \qquad \frac{}{inr : b \rightarrowtail a + b} \qquad \frac{}{clone : a \rightarrowtail a \times a}
$$

$$
\frac{}{1 \text{ inhabited}} \qquad \frac{b_1 \text{ inhabited} \quad b_2 \text{ inhabited}}{b_1 \times b_2 \text{ inhabited}} \qquad \frac{b_1 \text{ inhabited}}{b_1 + b_2 \text{ inhabited}} \qquad \frac{b_2 \text{ inhabited}}{b_1 + b_2 \text{ inhabited}}
$$

FIGURE 5. The syntax and type system of the arrow metalanguage $\mathcal{U}\Pi_a$.

3.3. **Quantum combinators with allocation: the arrow metalanguage $\mathcal{U}\Pi_a$.** Next we extend $\mathcal{U}\Pi$ with an *allocation effect* $alloc : 0 \rightarrowtail a$, yielding the language of $\mathcal{U}\Pi_a$ (*"yuppie-a"*). This effect is introduced by letting combinators $b_1 \rightarrowtail b_2$ in $\mathcal{U}\Pi_a$ be given by invertible combinators with a *heap* of type b_3: that is, as $\mathcal{U}\Pi$ combinators of type $b_1 + b_3 \leftrightarrow b_2$. Analogous to [**?**], this enables the type system to track the information effects.

This small extension will allow us to define a *classical cloning* combinator that clones classical states exactly, and sends quantum states $|\psi\rangle$ to $\sqrt{|\psi\rangle} \otimes \sqrt{|\psi\rangle}$; this will be crucial later on in deriving a combinator for decohering measurement in $\mathcal{U}\Pi_a^{\chi}$.

$\mathcal{U}\Pi_a$ canonically takes semantics in the category **Isometry** of Hilbert spaces and isometries: in Section **??**, we will see how a categorical model of $\mathcal{U}\Pi$ can be extended universally to model of $\mathcal{U}\Pi_a$, and Section **??** shows how this construction connects the canonical model of $\mathcal{U}\Pi$ to that of $\mathcal{U}\Pi_a$. Now we show that the approximate universality theorem for $\mathcal{U}\Pi$ with its unitary semantics extends to an approximate universality for $\mathcal{U}\Pi_a$ with its semantics in isometries.

Figure **??** gives an over view of $\mathcal{U}\Pi_a$. It is an arrow metalanguage [**?**, **?**, **?**] built atop $\mathcal{U}\Pi$: it has the same base types as $\mathcal{U}\Pi$, but introduces a new, irreversible combinator type $b \rightarrowtail b$ (reflecting the fact that combinators in $\mathcal{U}\Pi_a$ are no longer invertible). All combinators in $\mathcal{U}\Pi_a$ are constructed from

combinators in $\mathcal{U}\Pi$ by means of the *lift* constructor, following the type rule:

$$\frac{u : b_1 + b_3 \leftrightarrow b_2}{\mathit{lift}\ u : b_1 \rightarrowtail b_2}$$

So a combinator in $\mathcal{U}\Pi_a$ corresponds to a combinator in $\mathcal{U}\Pi$ with a hidden *heap* of type b_3. Section **??** will discuss that some quotienting is needed for this construction to behave; we defer further details about the semantics until then, including the arrow laws.

For this to constitute an arrow, we must produce meta-combinators *arr*, $\ggg$, and *first*. To start, *arr* must lift a $\mathcal{U}\Pi$ combinator u to a *pure* $\mathcal{U}\Pi_a$ one, free of effects. To do this, we assign it the trivial heap 0 and remove it before proceeding with u:

$$\mathit{arr}\ u = \mathit{lift}(\mathit{unit}^+ \fatsemi u)$$

In Figure **??**, *iso* refers to atomic combinators of $\mathcal{U}\Pi$ brought into $\mathcal{U}\Pi_a$ by applying *arr* to them. We write, for example, swap^+ in $\mathcal{U}\Pi_a$ to refer to $\mathit{arr}(\mathit{swap}^+)$, and so on.

To compose combinators $\mathit{lift}(u_1) : b_1 \rightarrowtail b_2$ and $\mathit{lift}(u_2) : b_2 \rightarrowtail b_3$ with heaps of type b_4 and b_4', we must track both heaps. The result will be a lifted $\mathcal{U}\Pi$ combinator of type $b_1 + (b_4 + b_4') \leftrightarrow b_3$ which permit u_1 and u_2 access to their parts of the heap accordingly:

$$\mathit{lift}(u_1) \ggg \mathit{lift}(u_2) = \mathit{lift}(\mathit{associ}^+ \fatsemi (u_1 + \mathit{id}) \fatsemi u_2)$$

The final combinator related to *lift* defining an arrow is *first*, allowing two arrows (each with their own information effects) to be executed in parallel. We define

$$\mathit{lift}(u_1) \ast\!\ast\!\ast \mathit{lift}(u_2) = \mathit{lift}(\mathit{associ}^+ \fatsemi \mathit{distribi} + \mathit{distribi} \fatsemi \mathit{swap}^\times + \mathit{swap}^\times \fatsemi \mathit{distribi} \fatsemi \mathit{swap}^\times \fatsemi u_1 \times u_2)$$

and derive $\mathit{first}\ d = d \ast\!\ast\!\ast \mathit{arr}(\mathit{id})$ and $\mathit{second}\ d = \mathit{arr}(\mathit{id}) \ast\!\ast\!\ast d$ as usual. That is, given $\mathit{lift}(u_1)$ and $\mathit{lift}(u_1)$ with $u_1 : b_1 + b_3 \leftrightarrow b_2$ and $u_2 : b_1' + b_3' \leftrightarrow b_2'$, this defines their product by choosing the heap to be $(b_3 \times b_1') + ((b_1 \times b_3') + (b_3 \times b_3'))$, as we then have:

$$\begin{aligned}(b_1 \times b_1') + ((b_1 \times b_3') + ((b_3 \times b_1') + (b_3 \times b_3'))) &\cong ((b_1 \times b_1') + (b_1 \times b_3')) + ((b_3 \times b_1') + (b_3 \times b_3')) \\ &\cong (b_1 \times (b_1' + b_3')) + (b_3 \times (b_1' + b_3')) \\ &\cong (b_1 + b_3) \times (b_1' + b_3')\end{aligned}$$

Make *lift* an *arrow with choice* by defining a combinator $d_1 +\!\!+\!\!+ d_2$ giving a choice between d_1 and d_2:

$$\mathit{lift}(u_1) +\!\!+\!\!+ \mathit{lift}(u_2) = \mathit{lift}((u_1 + u_2) \fatsemi \mathit{midswap}^+)$$

From this we can derive $\mathit{left}\ d = d +\!\!+\!\!+ \mathit{arr}(\mathit{id})$ and $\mathit{right}\ d = \mathit{arr}(\mathit{id}) +\!\!+\!\!+ d$.

What can we do with this arrow metalanguage? Firstly, we can construct the promised allocation combinator $0 \rightarrowtail a$ by lifting the $\mathcal{U}\Pi$ map $0 + a \leftrightarrow a$ that removes the additive unit on the left, i.e., $\mathit{alloc} = \mathit{lift}(\mathit{swap}^+ \fatsemi \mathit{unit}^+)$. From this we can recover injections *inl* and *inr* as

$$\mathit{inl} = \mathit{arr}(\mathit{uniti}^+) \ggg \mathit{right}(\mathit{alloc})$$

and analogously for *inr*, though we can also define them more simply as $\mathit{inl} = \mathit{lift}(\mathit{id})$ and $\mathit{inr} = \mathit{lift}(\mathit{swap}^+)$. Another crucial application of allocation is *classical cloning*.

3.3.1. *Classical cloning.* Classical can be copied, quantum data cannot. While there is a program that inputs a piece of classical data and outputs two copies of that data, no such program exists for quantum data; this is the *no cloning theorem* [**?**, **?**].

In light of this, it may come as a bit of a surprise that we can derive a combinator *clone* satisfying $[\![\mathit{clone}]\!](|0\rangle) = |00\rangle$ and $[\![\mathit{clone}]\!](|1\rangle) = |11\rangle$. After all, wouldn't that imply $[\![\mathit{clone}]\!](|\phi\rangle) = |\phi\rangle \otimes |\phi\rangle$ for a qubit $|\phi\rangle$? Fortunately not! To see this, consider a superposed state $|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.

$$\begin{aligned}[\![\mathit{clone}]\!](|\phi\rangle) &= [\![\mathit{clone}]\!](\alpha\,|0\rangle + \beta\,|1\rangle) = [\![\mathit{clone}]\!](\alpha\,|0\rangle) + [\![\mathit{clone}]\!](\beta\,|1\rangle) \\ &= \alpha\,[\![\mathit{clone}]\!](|0\rangle) + \beta\,[\![\mathit{clone}]\!](|1\rangle) = \alpha\,|00\rangle + \beta\,|11\rangle\end{aligned}$$

Now, $|00\rangle$ and $|11\rangle$ are shorthands for $|0\rangle \otimes |0\rangle$ and $|1\rangle \otimes |1\rangle$, and the tensor product of Hilbert spaces satisfies $s(|u\rangle \otimes |v\rangle) = (s\,|u\rangle) \otimes |v\rangle = |u\rangle \otimes (s\,|v\rangle)$ for all scalars s and vectors $|u\rangle \otimes |v\rangle$ in $U \otimes V$. This means for example $\alpha\,|00\rangle = (\sqrt{\alpha}\,|0\rangle) \otimes (\sqrt{\alpha}\,|0\rangle)$ which is distinct from $(\alpha\,|0\rangle) \otimes (\alpha\,|0\rangle)$ except when $\alpha = 1$. So applying *clone* to $|\phi\rangle$ does *not* gives two copies $|\phi\rangle \otimes |\phi\rangle$, but rather two copies of $|\phi\rangle$ with all amplitudes (in the computational basis) replaced by their *square roots*.

Define $\mathit{clone} : b \rightarrowtail b \times b$ by induction on the structure of b. The base cases are $\mathit{clone}_0 = \mathit{arr}(\mathit{distriboi})$ and $\mathit{clone}_1 = \mathit{arr}(\mathit{uniti}^\times)$. Products are cloned inductively by rearranging:

$$\mathit{clone}_{b \times b'} = (\mathit{clone}_b \ast\!\ast\!\ast \mathit{clone}_b') \ggg \mathit{arr}(\mathit{midswap}^\times)$$

Syntax

$$b ::= 0 \mid 1 \mid b + b \mid b \times b \qquad \text{(base types)}$$

$$t ::= b \rightsquigarrow b \qquad \text{(combinator types)}$$

$$c ::= \mathit{lift}\ v \qquad \text{(combinator constructor)}$$

$$d ::= c \mid \mathit{iso} \mid \mathit{arr}\ v \mid d \ggg d \mid \mathit{first}\ d \mid \mathit{second}\ d \mid \mathit{left}\ d \mid \mathit{right}\ d$$
$$\mid d \ast\!\!\ast\!\!\ast d \mid d +\!\!+\!\!+ d \mid \mathit{inhab} \mid \mathit{inl} \mid \mathit{inr} \mid \mathit{alloc} \mid \mathit{clone}$$
$$\mid \mathit{discard} \mid \mathit{fst} \mid \mathit{snd} \mid \mathit{merge} \mid \mathit{measure} \qquad \text{(combinators)}$$

Typing rules

$$\frac{v : b_1 \rightarrowtail b_2 \times b_3 \quad b_3 \text{ inhabited}}{\mathit{lift}\ v : b_1 \rightsquigarrow b_2} \qquad \frac{v : b_1 \rightarrowtail b_2}{\mathit{arr}\ v : b_1 \rightsquigarrow b_2} \qquad \frac{d_1 : b_1 \rightsquigarrow b_2 \quad d_2 : b_2 \rightsquigarrow b_3}{d_1 \ggg d_2 : b_1 \rightsquigarrow b_3}$$

$$\frac{d : b_1 \rightsquigarrow b_2}{\mathit{first}\ d : b_1 \times b_3 \rightsquigarrow b_2 \times b_3} \qquad \frac{d : b_1 \rightsquigarrow b_2}{\mathit{second}\ d : b_3 \times b_1 \rightsquigarrow b_3 \times b_2}$$

$$\frac{d : b_1 \rightsquigarrow b_2}{\mathit{left}\ d : b_1 + b_3 \rightsquigarrow b_2 + b_3} \qquad \frac{d : b_1 \rightsquigarrow b_2}{\mathit{right}\ d : b_3 + b_1 \rightsquigarrow b_3 + b_2}$$

$$\frac{d_1 : b_1 \rightsquigarrow b_3 \quad d_2 : b_2 \rightsquigarrow b_4}{d_1 +\!\!+\!\!+ d_2 : b_1 + b_2 \rightsquigarrow b_3 + b_4} \qquad \frac{d_1 : b_1 \rightsquigarrow b_3 \quad d_2 : b_2 \rightsquigarrow b_4}{d_1 \ast\!\!\ast\!\!\ast d_2 : b_1 \times b_2 \rightsquigarrow b_3 \times b_4} \qquad \frac{b \text{ inhabited}}{\mathit{inhab} : 1 \rightsquigarrow b}$$

$$\frac{}{\mathit{alloc} : 0 \rightsquigarrow a} \qquad \frac{}{\mathit{inl} : a \rightsquigarrow a + b} \qquad \frac{}{\mathit{inr} : b \rightsquigarrow a + b} \qquad \frac{}{\mathit{clone} : a \rightsquigarrow a \times a}$$

$$\frac{}{\mathit{discard} : a \rightsquigarrow 1} \qquad \frac{}{\mathit{fst} : a \times b \rightsquigarrow a} \qquad \frac{}{\mathit{snd} : a \times b \rightsquigarrow b} \qquad \frac{}{\mathit{merge} : a + a \rightsquigarrow a} \qquad \frac{}{\mathit{measure} : a \rightsquigarrow a}$$

FIGURE 6. The syntax and type system of the arrow metalanguage $\mathcal{U}\Pi_a^\chi$ (rules for inhabitation appear in Figure **??**).

Sums are cloned inductively, tagging accordingly, and factoring:

$$\mathit{clone}_{b+b'} = (\mathit{clone}_b +\!\!+\!\!+ \mathit{clone}_{b'}) \ggg ((\mathit{inl} +\!\!+\!\!+ \mathit{id}) \ast\!\!\ast\!\!\ast (\mathit{inr} +\!\!+\!\!+ \mathit{id})) \ggg \mathit{arr}(\mathit{distribi}).$$

Interestingly, though the languages and semantics are different, cloning is defined *precisely* as for classical information effects [**?**].

3.3.2. *Inhabitation.* Later, we will need a notion of *inhabited types* in $\mathcal{U}\Pi_a$. By a type b being inhabited in $\mathcal{U}\Pi_a$, we mean that there is a combinator $1 \rightarrowtail b$. For inhabited types b, construct canonical inhabitants as follows. First, $\mathit{inhab}_1 = \mathit{id}$. The inhabitant of a product type $b \times b'$ is the product $\mathit{inhab}_{b\times b'} = \mathit{uniti}^\times \ggg \mathit{inhab}_b \ast\!\!\ast\!\!\ast \mathit{inhab}_{b'}$ of inhabitants. Finally, a sum $b + b'$ is inhabited if either b or b' is: if b is inhabited set $\mathit{inhab}_{b+b'} = \mathit{inhab}_b \ggg \mathit{inl}$, and if b is not inhabited but b' is, $\mathit{inhab}_{b+b'} = \mathit{inhab}_{b'} \ggg \mathit{inr}$.

3.3.3. *Expressiveness.* We can now extend the expressiveness theorem for $\mathcal{U}\Pi$ to one for $\mathcal{U}\Pi_a$. The proof can be found in the Supplementary Material.

Theorem 3. *$\mathcal{U}\Pi_a$ is approximately universal for isometries: For any $2^n \times 2^m$ isometry V and $\delta > 0$ there exists a $\mathcal{U}\Pi_a$ combinator v such that $\|V - [\![v]\!]\|_{\mathrm{op}} < \delta$.*

3.4. **Quantum combinators with hiding and allocation: the arrow metalanguage $\mathcal{U}\Pi_a^\chi$.** We finally extend $\mathcal{U}\Pi_a$ with an additional information effect to *hide* information via a combinator $\mathit{discard}{:}b \rightsquigarrow 1$, giving us the language of $\mathcal{U}\Pi_a^\chi$ (*"yuppie-chi-a"*). Dually to how allocation was introduced in $\mathcal{U}\Pi_a$, discarding is introduced in $\mathcal{U}\Pi_a^\chi$ by letting combinators $b_1 \rightsquigarrow b_2$ be given by $\mathcal{U}\Pi_a$ combinators of type $b_1 \rightarrowtail b_2 \times b_3$, where we think of b_3 as the type of *garbage* produced by the combinator. In order to be able to produce a choice metacombinator, however, we need to make the additional assumption that this garbage is inhabited. This is a very mild assumption, since garbage can always be chosen to be inhabited.

The hiding combinator allows projections $\mathit{fst} : b_1 \times b_2 \rightsquigarrow b_1$ and $\mathit{snd} : b_1 \times b_2 \rightsquigarrow b_2$ to be defined. When combined with the classical cloning combinator inherited from $\mathcal{U}\Pi_a$, we show that a combinator $\mathit{measure} : b \rightsquigarrow b$ for decohering measurement can be derived.

$\mathcal{U}\Pi_a^\chi$ takes its canonical semantics in the category **CPTP** of Hilbert spaces and quantum channels, and as with $\mathcal{U}\Pi_a$, we will show in Section **??** how a model of $\mathcal{U}\Pi_a$ can be extended to one of $\mathcal{U}\Pi_a^\chi$ by a universal construction, connecting isometries to quantum channels (more on this in Section **??**). We also

extend the approximate universality theorem of $\mathcal{UΠ}_a$ to one showing approximate universality of $\mathcal{UΠ}_a^\chi$ combinators with respect to quantum channels.

Like $\mathcal{UΠ}_a$, $\mathcal{UΠ}_a^\chi$ is an arrow metalanguage extending $\mathcal{UΠ}_a$ (see Figure **??** for an overview). It uses the same base types as $\mathcal{UΠ}$ and $\mathcal{UΠ}_a$, but introduces a new combinator type $b \rightsquigarrow b$ to distinguish $\mathcal{UΠ}_a^\chi$ combinators at the type level. All combinators in $\mathcal{UΠ}_a^\chi$ are constructed from $\mathcal{UΠ}_a$ combinators using the *lift* constructor

$$\frac{v : b_1 \rightarrowtail b_2 \times b_3 \quad b_3 \text{ inhabited}}{\mathit{lift}\ v : b_1 \rightsquigarrow b_2} \ .$$

The definition of the arrow metacombinators *arr*, $\ggg$, and *first* are bound to look very familiar, as they are defined dually to those in $\mathcal{UΠ}_a$ (indeed, we will see in Section **??** that the two constructions are dual in a formal sense). To turn a $\mathcal{UΠ}_a$ combinator $b_1 \rightarrowtail b_2$ into a pure $\mathcal{UΠ}_a^\chi$ combinator $b_1 \rightsquigarrow b_2$ can be done by assigning it the trivial (and trivially inhabited) garbage of 1,

$$\mathit{arr}(v) = \mathit{lift}(v \ggg \mathit{uniti}^\times) \ .$$

Combinators of type $b_1 \rightsquigarrow b_2$ and $b_2 \rightsquigarrow b_3$ with garbage of type b_4 and b_4' respectively can be composed by

$$\mathit{lift}(v_1) \ggg \mathit{lift}(v_2) = \mathit{lift}(v_1 \ggg (v_2 \ast\!\!\ast\!\!\ast\ \mathit{id}) \ggg \mathit{assoc}^\times)$$

resulting in a combinator with garbage $b_4' \times b_4$. For the final arrow combinator *first* allowing parallel execution of arrows, we define $\ast\!\!\ast\!\!\ast$ to simply run the underlying $\mathcal{UΠ}_a$ combinators in parallel and swap the garbage into the right position as necessary,

$$\mathit{lift}(v_1) \ast\!\!\ast\!\!\ast\ \mathit{lift}(v_2) = \mathit{lift}((v_1 \ast\!\!\ast\!\!\ast\ v_2) \ggg \mathit{midswap}^\times) \ ,$$

such that the garbage of $d_1 \ast\!\!\ast\!\!\ast\ d_2$ is the product of the garbages of d_1 and d_2 respectively. We derive $\mathit{first}(d) = d \ast\!\!\ast\!\!\ast\ \mathit{id}$ and $\mathit{second}(d) = \mathit{id} \ast\!\!\ast\!\!\ast\ d$. All of these definitions are straightforwardly seen to preserve the inhabitation requirement on garbage.

Defining the choice metacombinator $+\!+\!+$ is a bit more tricky, and it turns out to be easier to define *left* and derive $+\!+\!+$ and *right* from it. The idea is to exploit distributivity and inhabitation of garbage: if $d : b_1 \rightsquigarrow b_2$ produces garbage of type b_4 and the identity produces garbage of type 1, we can use the inhabitation of b_4 to turn the trivial garbage into garbage of type b_4 via $\mathit{inhab} : 1 \rightsquigarrow b_4$, and then distribute out on the right to get something of the required type $(b_2 + b_3) \times b_4$. This gives us the definition

$$\mathit{left}(\mathit{lift}\ v) = \mathit{lift}((v +\!+\!+ \mathit{uniti}^\times) \ggg (\mathit{id} +\!+\!+ (\mathit{id} \ast\!\!\ast\ \mathit{inhab})) \ggg (\mathit{swap}^\times +\!+\!+ \mathit{swap}^\times) \ggg \mathit{distribi} \ggg \mathit{swap}^\times)$$

from which we derive *right* and $+\!+\!+$ as usual [**?**] as

$$\mathit{right}(d) = \mathit{swap}^+ \ggg \mathit{left}(d) \ggg \mathit{swap}^+ \qquad d_1 +\!+\!+ d_2 = \mathit{left}(d_1) \ggg \mathit{right}(d_2) \ .$$

The combinators *alloc*, *inl*, *inr*, *clone*, and *inhab* related to the allocation effect from $\mathcal{UΠ}_a$, as well as all of the base combinators of $\mathcal{UΠ}$ lifted to $\mathcal{UΠ}_a$, can be further lifted to $\mathcal{UΠ}_a^\chi$ by applying *arr* to them (these are denoted by *iso* in Figure **??**).

Information hiding is introduced in $\mathcal{UΠ}_a^\chi$ by means of the effectful $\mathit{discard} : b \rightsquigarrow 1$ combinator. Some finesse is required to manage the inhabitation requirement on garbage, however. On all types aside from 0, *discard* is given by lifting the $\mathcal{UΠ}_a$ combinator $b \rightarrowtail 1 \times b$ that adds the multiplicative unit on the left,

$$\mathit{discard} = \mathit{lift}(\mathit{uniti}^\times \ggg \mathit{swap}^\times) \ .$$

On 0, we first need to use *alloc* to allocate something of inhabited type, namely 1, before we can discard it:

$$\mathit{discard} = \mathit{lift}(\mathit{alloc} \ggg \mathit{uniti}^\times) \ .$$

Analogously to the injections in $\mathcal{UΠ}_a$, we can derive *projections* from this discarding effect as

$$\mathit{fst} = \mathit{id} \ast\!\!\ast\!\!\ast\ \mathit{discard} \ggg \mathit{unit}^\times \qquad \mathit{snd} = \mathit{swap}^\times \ggg \mathit{fst}$$

though these can also be defined more simply as $\mathit{fst} = \mathit{lift}(\mathit{id})$ and $\mathit{snd} = \mathit{lift}(\mathit{swap}^\times)$.

To allow the choice metacombinator to be used for conditional execution, we need a way to merge branches. This can be defined in $\mathcal{UΠ}_a^\chi$ as the $\mathit{merge} : b + b \rightsquigarrow b$ combinator, exploiting hiding and the fact that $a + a \cong a \times (1 + 1)$, as in

$$\mathit{merge} = (\mathit{uniti}^\times +\!+\!+ \mathit{uniti}^\times) \ggg \mathit{distribi} \ggg \mathit{fst} \ .$$

We are finally ready to explore the measurement combinator in $\mathcal{UΠ}_a^\chi$.

3.4.1. *Measurement in the computational basis.* As we have seen previously, $\mathcal{U}\Pi_a$ permits a notion of classical cloning, and $\mathcal{U}\Pi_a^\chi$ inherits it. When we combine this with the ability to discard information in $\mathcal{U}\Pi_a^\chi$ using the projections, we obtain a surprisingly robust notion of measurement in the computational basis. This measurement combinator $\mathit{measure} : b \rightsquigarrow b$ is defined simply to be

$$\mathit{measure} = \mathit{clone} \ggg \mathit{fst}$$

Classically, this is a complicated way of doing absolutely nothing – the map takes a piece of classical data, copies it, and then immediately throws away the copy. In the quantum case, however, this performs decohering measurement. We illustrate this by an example:

Consider an arbitrary qubit state $|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, and the associated density matrix

$$|\phi\rangle\langle\phi| = (\alpha\,|0\rangle + \beta\,|1\rangle)(\overline{\alpha}\,\langle 0| + \overline{\beta}\,\langle 1|)$$

Conjugating by $[\![\mathit{clone}]\!]$ (noting that this does indeed perform classical cloning in the canonical model of **CPTP**) yields the density matrix

$$\begin{aligned}[\![\mathit{clone}]\!]\,|\phi\rangle\langle\phi|\,[\![\mathit{clone}]\!]^\dagger &= (\alpha\,|00\rangle + \beta\,|11\rangle)(\overline{\alpha}\,\langle 00| + \overline{\beta}\,\langle 11|) \\ &= |\alpha|^2\,|00\rangle\langle 00| + \alpha\overline{\beta}\,|00\rangle\langle 11| + \beta\overline{\alpha}\,|11\rangle\langle 00| + |\beta|^2\,|11\rangle\langle 11| \quad .\end{aligned}$$

We remark that, in **CPTP**, $[\![\mathit{fst}]\!]$ is given by the *partial trace* (see, e.g., [**?**]) of density matrices. This means that

$$\begin{aligned}&\pi_1(|\alpha|^2\,|00\rangle\langle 00| + \alpha\overline{\beta}\,|00\rangle\langle 11| + \beta\overline{\alpha}\,|11\rangle\langle 00| + |\beta|^2\,|11\rangle\langle 11|) \\ &\quad = |\alpha|^2\,\mathrm{tr}(|0\rangle\langle 0|)\,|0\rangle\langle 0| + \alpha\overline{\beta}\,\mathrm{tr}(|0\rangle\langle 1|)\,|0\rangle\langle 1| + \beta\overline{\alpha}\,\mathrm{tr}(|1\rangle\langle 0|)\,|1\rangle\langle 0| + |\beta|^2\,\mathrm{tr}(|1\rangle\langle 1|)\,|1\rangle\langle 1| \\ &\quad = |\alpha|^2\,|0\rangle\langle 0| + |\beta|^2\,|1\rangle\langle 1|\end{aligned}$$

since for vectors $|a\rangle$ and $|b\rangle$ in the computational basis, $\mathrm{tr}(|a\rangle\langle b|) = 1$ when $|a\rangle = |b\rangle$, and $\mathrm{tr}(|a\rangle\langle b|) = 0$ otherwise. Note that measurement in an arbitrary basis can then be performed by conjugating the measurement combinator with the appropriate change-of-base combinator. For example, measurement in the Hadamard basis $\{|+\rangle, |-\rangle\}$ is performed using $\mathit{hadamard} \ggg \mathit{measure} \ggg \mathit{hadamard}$.

3.4.2. *Conditionals.* The intention behind the choice structure on arrows is that it gives rise to conditional execution of arrows. Unfortunately, the standard construction of conditionals in arrows with choice [**?**] may not be the most desirable one in the quantum case, as it relies implicitly on measuring a large part of the state involved.

In the usual construction of conditionals, given a predicate on b (a combinator $b \rightsquigarrow 1 + 1$) we must first construct its associated *test* [**?**, **?**]. This is done by

$$\mathit{test}(p) = \mathit{clone} \ggg \mathit{id} \ast\!\!\ast\!\!\ast\, p \ggg \mathit{distrib} \ggg (\mathit{unit}^\times +\!\!+\!\!+ \mathit{unit}^\times) \quad .$$

Using the *merge* combinator to join control flow paths, the conditional combinator *if* p *then* e_1 *else* e_2 is defined as

$$\mathit{if}\ p\ \mathit{then}\ e_1\ \mathit{else}\ e_2 = \mathit{test}(p) \ggg e_1 +\!\!+\!\!+ e_2 \ggg \mathit{merge} \quad .$$

From this, it may be difficult to see exactly where the measurement occurs. Consider the following, simple example, where b is some ensemble of qubits $b = (1+1) \times \cdots \times (1+1)$ and the predicate p simply extracts the first qubit from the ensemble, i.e., $p = \mathit{fst}$. (This is only slightly idealised; *any* predicate on b must be some lifted $\mathcal{U}\Pi_a$ combinator followed by a projection, which can always be assumed to be the first projection.) But in this case, the first part of $\mathit{test}(p)$ is $\mathit{clone} \ggg \mathit{fst} \ast\!\!\ast\!\!\ast \mathit{id}$, which, by definition, is measurement applied to all but the first qubit, which is merely classically cloned. In Section **??**, we will see another method of constructing conditionals (due to [**?**]) more suitable to the quantum case, in that it only requires the measurement of the single qubit necessary to unambiguously direct control flow.

3.4.3. *Expressiveness.* Finally we can extend the universality theorems for $\mathcal{U}\Pi$ and $\mathcal{U}\Pi_a$ to one that $\mathcal{U}\Pi_a^\chi$ is approximately universal for arbitrary quantum computations, that is, quantum channels. The proof can be found in the Supplementary Material.

Theorem 4. *$\mathcal{U}\Pi_a^\chi$ is approximately universal for quantum channels: For any quantum channel Λ and $\delta > 0$ there exists a $\mathcal{U}\Pi_a^\chi$ combinator c such that $\|\Lambda - [\![c]\!]\|_{\mathrm{op}} < \delta$.*

4. Categorical semantics

In this section we develop denotational semantics for the simple programming languages of the previous section in three stages. First, the base language Π can be interpreted in rig groupoids. This is then extended to $\mathcal{U}\Pi$ by providing interpretations for the phase and Hadamard combinators. Finally, we discuss two categorical constructions, R and L, that model the arrow constructions – i.e., *lift* combinator constructors of $\mathcal{U}\Pi_a$ and $\mathcal{U}\Pi_a^\chi$ respectively – in order to encapsulate information allocation and hiding and account for measurement.

4.1. **Rig groupoids.** To interpret the type system and semantics of Π in a category, it needs to have combinators $\oplus$ and $\otimes$ that distribute over each other in the correct way. This is captured in the notion of a *rig groupoid*. Recall that a groupoid is a category in which every morphism is invertible.

Definition 5. A *rig category* is category $\mathbf{C}$ with two symmetric monoidal structures $(\oplus, 0)$ and $(\otimes, I)$, as well as natural isomorphisms

$$\delta_{A,B,C} \colon A \otimes (B \oplus C) \to (A \otimes B) \oplus (A \otimes C)$$
$$\delta_0 \colon A \otimes 0 \to 0$$

satisfying several coherence laws for which we refer to [**?**, **?**].

Here $\oplus$ need not be a coproduct, and $\otimes$ need not be a product, which is the special case of a *distributive category*.

A rig groupoid suffices to interpret Π. Being a language for classical reversible computing, a canonical such model is the rig groupoid **FinBij** of finite sets and bijective functions. The base types are interpreted as $[\![0]\!] = 0$, $[\![1]\!] = I$, $[\![b + b']\!] = [\![b]\!] \oplus [\![b']\!]$, and $[\![b \times b']\!] = [\![b]\!] \otimes [\![b']\!]$. The combinator type $b \leftrightarrow b'$ becomes the (invertible) morphisms $[\![b]\!] \to [\![b']\!]$. As for the atomic combinators, the swap morphisms for $\oplus$ and $\otimes$ interpret $swap^+$ and $swap^\times$, respectively. The unitor $\rho^\oplus_A \colon A \oplus \to 0$ and its inverses coming from the monoidal structure $(\oplus, 0)$ denote $unit^+$ and $uniti^+$. The associator $\alpha^\oplus_{A,B,C} \colon (A \oplus B) \oplus C \to A \oplus (B \oplus C)$ and its inverse interpret $assoc^+$ and $associ^+$. We use the coherence isomorphisms of $(\otimes, I)$ to interpret $[\![unit^\times]\!] = \rho^\otimes$, $[\![uniti^\times]\!] = (\rho^\otimes)^{-1}$, $[\![assoc^\times]\!] = \alpha^\otimes$, and $[\![associ^\times]\!] = (\alpha^\otimes)^{-1}$. The distributors finish the interpretation: $[\![distrib]\!] = \delta$, $[\![distribi]\!] = \delta^{-1}$, $[\![distribo]\!] = \delta_0$, and $[\![distriboi]\!] = \delta_0^{-1}$. The combinators are simply composition via $\circ$, $\oplus$, and $\otimes$.

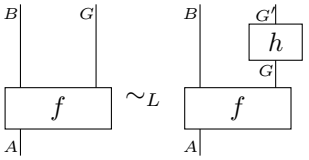
Interpreting $\mathcal{U}\Pi$ requires additionally to give semantics to the $phase_\varphi$ and *hadamard* combinators. In principle, they can be given trivial semantics (i.e., as identities) in any rig groupoid, though that would yield mere classical semantics, and thus defeat the purpose of the quantum extension to Π to begin with. A far better approach would be to interpret $\mathcal{U}\Pi$ in a category with quantum capabilities, the canonical choice being the category **Unitary** of finite dimensional Hilbert spaces and unitaries. Here, the semantics of these two combinators are given by

$$[\![phase_\varphi]\!] \quad = \quad x \mapsto \varphi \cdot x \qquad\qquad [\![hadamard]\!] \quad = \quad \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} .$$

Notice that another way of writing $[\![phase_\varphi]\!]$ would be as the 1×1 matrix (φ), cementing the idea of phases as unitaries $\mathbb{C} \to \mathbb{C}$.

4.2. **Garbage and heap.** We now extend the categorical semantics of $\mathcal{U}\Pi$ to take quantum information effects into account. Let $\mathbf{C}$ be a symmetric monoidal category. We think of its objects as types, its morphisms as programs, and its tensor product as parallel composition. Hence a morphism $f \colon A \to B \otimes G$ will denote a program that takes an input of type A and produces an output of type B together with some *garbage* of type G. As we want to disregard the garbage, we will identify this morphism with $(\mathrm{id}_B \otimes h) \circ f \colon A \to B \otimes G'$ for any morphism $h \colon G \to G'$ that postprocesses the garbage, called a *mediator*. That is, we consider the equivalence relation $\sim_L$ generated by:

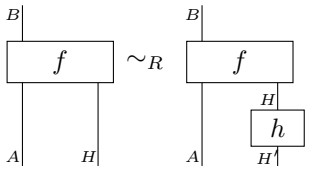
(1)



Dually, instead of garbage, we can also consider a *heap*. That is, morphisms $f \colon A \oplus H \to B$ will denote a program that takes input of type A and may use a heap H in producing an output of type B. Again, we only care about access to the heap and not the actual contents of the heap, so we will identify

f with $f \circ (h \oplus \mathrm{id}_H) \colon A \oplus H' \to B$ for any morphisms $h \colon H' \to H$ that preprocesses the heap. That is, we consider the equivalence relation $\sim_R$ generated by:

(2)



What exactly are $\sim_L$ and $\sim_R$? Equations (**??**) and (**??**) define relations that are already reflexive (with the identity as mediator) and transitive (compose mediators), but not always symmetric. The following lemma shows that they are already symmetric in special cases of interest, such as when the base category $\mathbf{C}$ is a groupoid.

Lemma 6. *When every morphism in* $\mathbf{C}$ *is split monic, equation* (**??**) *defines an equivalence relation. When every morphism in* $\mathbf{C}$ *is split epic, equation* (**??**) *defines an equivalence relation.*

Proof. It suffices to establish symmetry. If $(\mathrm{id} \otimes h) \circ f = g$, there is k with $k \circ h = \mathrm{id}$, so $(\mathrm{id} \otimes k) \circ g = (\mathrm{id} \otimes k) \circ (\mathrm{id} \otimes h) \circ f = f$. An analogous argument holds for $\sim_R$. □

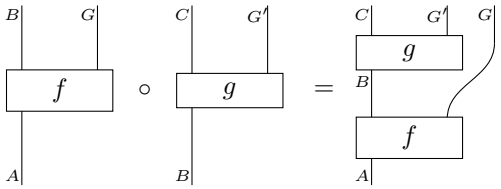
Lemma 7. *Let* $(\mathbf{C}, \oplus)$ *be a monoidal category. If* $f \sim_R f'$ *and* $g \sim_R g'$*, then:*

(i) $g \circ f \sim_R g' \circ f'$ *if* g *and* f *are composable;*
(ii) $f \oplus g \sim_R f' \oplus g'$*;*

Dually $g \circ f \sim_L g' \circ f'$ *and* $f \otimes g \sim_L f' \otimes g'$ *when* $f \sim_L f'$ *and* $g \sim_L g'$ *in a monoidal category* $(\mathbf{C}, \otimes)$*.*

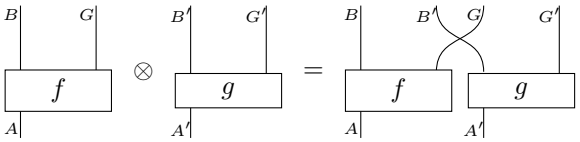
Proposition 8. *If* $(\mathbf{C}, \otimes, I)$ *is a symmetric monoidal category, there is a well-defined symmetric monoidal category* $L[\mathbf{C}]$ *whose:*

- objects *are the same as those of* $\mathbf{C}$*;*
- morphisms $A \to B$ *are equivalence classes of morphisms* $A \to B \otimes G$ *in* $\mathbf{C}$ *under* (**??**)*;*
- composition *is:*



- identities *are the inverse right unitors:*

- tensor unit I *is as in* $\mathbf{C}$*;*
- tensor product of objects *is as in* $\mathbf{C}$*;*
- tensor product of morphisms *is:*



Dually, if $(\mathbf{C}, \oplus, O)$ *is a symmetric monoidal category, there is a well-defined symmetric monoidal category* $R[\mathbf{C}] = L[\mathbf{C}^{\mathrm{op}}]^{\mathrm{op}}$*. Explicitly:*

- objects *are the same as those of* $\mathbf{C}$*;*
- morphisms $A \to B$ *are equivalence classes of morphisms* $A \oplus H \to B$ *in* $\mathbf{C}$ *under* (**??**)*.* □

Proof. Well-definedness follows from Lemma **??**. It is straightforward to verify that coherence isomorphisms in $L[\mathbf{C}]$ may be taken to be those in $\mathbf{C}$ composed with the inverse right unitor. □

So what is the point of doing these constructions? This is made clear in the following proposition. Indeed, in Section **??**, we will see that $L[\mathbf{C}]$ and $R[\mathbf{C}]$ are, in a certain sense, the smallest categories containing $\mathbf{C}$ with these properties.

Proposition 9. *The monoidal unit* I *is terminal in* $L[\mathbf{C}]$*, and the monoidal unit* O *is initial in* $R[\mathbf{C}]$*.*

As a consequence of this fact, there are canonical projections $X \otimes Y \xrightarrow{\pi_1} X$ and $Y \otimes Y \xrightarrow{\pi_2} X$ in $L[\mathbf{C}]$ given by $X \otimes Y \xrightarrow{\mathrm{id}\otimes !} X \otimes I \xrightarrow{\rho^{\otimes}} X$ (and symmetrically for π_2). Likewise, there are canonical injections $X \xrightarrow{\amalg_1} X \oplus Y$ and $Y \xrightarrow{\amalg_2} X \oplus Y$ (defined dually to the above) in $R[\mathbf{C}]$.

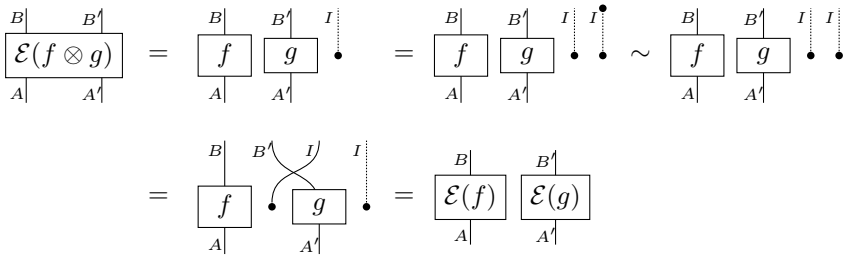
Proposition 10. *If $(\mathbf{C}, \otimes)$ is a monoidal category, there is a strict monoidal functor $\mathcal{E}\colon \mathbf{C} \to L[\mathbf{C}]$ given by $\mathcal{E}(A) = A$ on objects, and on morphisms as:*

$$\mathcal{E}(A \xrightarrow{f} B) = A \xrightarrow{f} B \overset{\rho^{\otimes -1}}{\to} B \otimes I$$

Dually, if $(\mathbf{C}, \oplus)$ is a monoidal category, there is a strict monoidal functor $\mathcal{D}\colon \mathbf{C} \to R[\mathbf{C}]$ given by $\mathcal{D}(A) = A$ on objects, and on morphisms as:

$$\mathcal{D}(A \xrightarrow{f} B) = A \oplus 0 \overset{\rho^{\oplus}}{\to} A \xrightarrow{f} B$$

Proof. By definition $\mathcal{E}(A \otimes B) = \mathcal{E}(A) \otimes \mathcal{E}(B)$ on objects. On morphisms:



Coherence isomorphisms in $L[\mathbf{C}]$ are precisely the image under $\mathcal{E}$ of those in $\mathbf{C}$. □

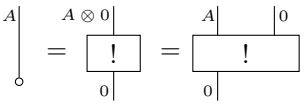
4.3. **Lifting tensor products.** We will compose the L and R constructions, applying them to the base rig groupoid consecutively. In this section we show that if $\mathbf{C}$ is a rig category then so is $R[\mathbf{C}]$, while $L[R[\mathbf{C}]]$ loses its direct sum and becomes merely a monoidal category. However, we will see later that the direct sum in $L[R[\mathbf{C}]]$ is *binoidal* and satisfies the laws associated with an arrow with choice.

The overall idea with these constructions is that if $\mathbf{C}$ interprets the base language $\mathcal{U}\Pi$, then $R[\mathbf{C}]$ interprets the arrow metalanguage $\mathcal{U}\Pi_a$ of $\mathcal{U}\Pi$ extended with allocation. More precisely, if $[\![u]\!]\colon [\![b_1]\!] \oplus [\![b_3]\!] \to [\![b_2]\!]$ is the interpretation of some $\mathcal{U}\Pi$ combinator u in a rig groupoid $\mathbf{C}$, the interpretation $[\![\mathit{lift}(u)]\!]$ in $\mathcal{U}\Pi_a$ is given by the equivalence class $[[\![u]\!]]_{\sim_R} : [\![b_1]\!] \to [\![b_2]\!]$ in $R[\mathbf{C}]$. In turn, $L[R[\mathbf{C}]]$ interprets the arrow metalanguage $\mathcal{U}\Pi_a^{\chi}$ extending $\mathcal{U}\Pi$ with allocation and hiding by interpreting $[\![\mathit{lift}(v)]\!]\colon [\![b_1]\!] \to [\![b_2]\!]$ as the equivalence class of $[\![v]\!]\colon [\![b_1]\!] \to [\![b_2]\!] \otimes [\![b_3]\!]$ in $R[\mathbf{C}]$. Later, in Section **??** we will exhibit universal properties of $R[\mathbf{C}]$ and $L[\mathbf{C}]$, and argue that they justify these constructions in the canonical semantics of $\mathcal{U}\Pi$: $R[\mathbf{Unitary}]$ is equivalent to the category **Isometry** of finite dimensional Hilbert spaces and isometries, while $L[R[\mathbf{Unitary}]]$ is equivalent to the category **CPTP** of finite dimensional Hilbert spaces and quantum channels.

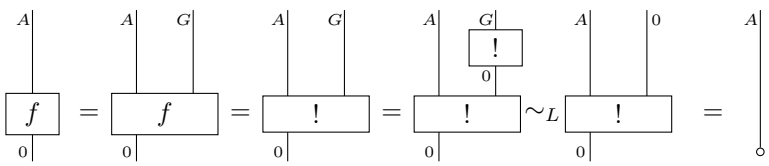
Although L and R are dual constructions, the order in which they transform $\mathbf{C}$ is important: we first add heaps and then garbage. The reason for this asymmetry is the following lemma.

Lemma 11. *Let $\mathbf{C}$ be a symmetric monoidal category. If $\mathbf{C}$ has an initial object, then so does $L[\mathbf{C}]$.*

Proof. Let 0 be an initial object in $\mathbf{C}$. For each object A there is then a morphism $0 \to A$ in $L[\mathbf{C}]$ given by the morphism



in $\mathbf{C}$. If $f\colon 0 \to A$ is a morphism in $L[\mathbf{C}]$, represented by a morphism $f\colon 0 \to A \otimes G$ in $\mathbf{C}$, then:



Thus 0 is indeed initial in $L[\mathbf{C}]$ as well. □

We start by endowing $\otimes$ and $\oplus$ with the capability to allow heaps.

Lemma 12. *If* $\mathbf{C}$ *is a rig category, then so is* $R[\mathbf{C}]$.

Proof. The monoidal structure $(\oplus, 0)$ is inherited from $\mathbf{C}$ straightforwardly. More intricately, $R[\mathbf{C}]$ is monoidal with $(\otimes, I)$ inherited from $\mathbf{C}$:

- the tensor product of objects is $A \otimes B$;
- the tensor unit is I;
- the tensor product of morphisms $f\colon A \oplus H \to B$ and $f'\colon A' \oplus H' \to B'$ is

$$(A \otimes A') \oplus H'' \xrightarrow{\delta^{-1}} (A \oplus H) \otimes (A' \oplus H') \xrightarrow{f \otimes f'} B \otimes B'$$

  where $H'' = (H \otimes A') \oplus (A \otimes H') \oplus (H \otimes H')$;

For the rest of the proof, see the Supplementary Material. □

The dual result does not hold: if $\mathbf{C}$ is a rig category, then $L[\mathbf{C}]$ need not be (though it is always monoidal by Proposition **??**). This asymmetry is caused by the fact that $\otimes$ distributes over $\oplus$, but not the other way around. Lemma **??** is the nullary case of this fact. For a special case where this does hold, regard a Boolean algebra as a posetal category $\mathbf{C}$; then $\wedge$ and $\vee$ do distribute over each other both ways, so in that case $L[\mathbf{C}]$ is again a rig category.

Lemma 13. *If* $\mathbf{C}$ *is a rig category, then* $\mathcal{D}\colon \mathbf{C} \to R[\mathbf{C}]$ *is a strict rig functor.*

Proof. By definition $\mathcal{D}(A \otimes B) = \mathcal{D}(A) \otimes \mathcal{D}(B)$ on objects. On morphisms, $\mathcal{D}(f) \otimes \mathcal{D}(g)$ is

$$(A \otimes A') \oplus H'' \xrightarrow{\delta} (A \oplus 0) \otimes (A' \oplus 0) \xrightarrow{\rho^{\oplus} \otimes \rho^{\oplus}} A \otimes A' \xrightarrow{f \otimes g} B \otimes B'$$

with $H'' = (0 \otimes A') \oplus (A \otimes 0) \oplus (0 \otimes 0)$. Now $H'' \simeq 0$ by successive applications of the annihilators and unitors for $\oplus$, so by coherence $\mathcal{D}(f) \otimes \mathcal{D}(g)$ is the composition of $\rho^{\oplus}\colon (A \otimes A') \oplus 0 \to A \otimes A'$ and $f \otimes g$, which is precisely $\mathcal{D}(f \otimes g)$. □

4.4. **Arrows with choice.** To complete the categorical semantics, we will show that $L[R[\mathbf{C}]]$ supports *arrows with choice* [**?**]. We saw in Proposition **??** and Lemma **??** that if $\mathbf{C}$ is a rig category, then $L[R[\mathbf{C}]]$ is monoidal under $\otimes$. What happens to $\oplus$? It is no longer necessarily a monoidal structure on $L[R[\mathbf{C}]]$, but we will show that it is *binoidal* – roughly, it is monoidal except that $\oplus$ is only a functor in each variable separately rather than in both variables jointly [**?**]. This is enough to interpret conditionals [**?**].

Definition 14. A category $\mathbf{C}$ is *binoidal* when it is equipped with functors $A \oplus -\colon \mathbf{C} \to \mathbf{C}$ and $- \oplus B\colon \mathbf{C} \to \mathbf{C}$ for each choice of objects $A, B \in \mathbf{C}$ such that applying the first functor to B results in the same object $A \oplus B$ as applying the second functor to A.

An *arrow with choice* is a functor F from a monoidal category $(\mathbf{C}, \oplus, 0)$ where 0 is initial to a binoidal category $(\mathbf{D}, \oplus)$ such that for any $g\colon B \to B'$:

$$F(f \oplus B) = F(f) \oplus B \tag{3}$$

$$F(\amalg_1) \circ f = (f \oplus B) \circ F(\amalg_1) \tag{4}$$

$$(f \oplus B') \circ F(A \oplus g) = F(A \oplus g) \circ (f \oplus B) \tag{5}$$

$$\alpha_{\oplus} \circ ((f \oplus B) \oplus C) = (f \oplus (B \oplus C)) \circ \alpha_{\oplus} \tag{6}$$

To prove that $L[R[\mathbf{C}]]$ is binoidal under $\oplus$, we will require that it has *inhabited garbage*: for any equivalence class of morphisms $A \to B \otimes G$ in $R[\mathbf{C}]$, there is a morphism $I \to G$ in $R[\mathbf{C}]$. Because 0 is initial in $R[\mathbf{C}]$, the problematic case where there is an isomorphism $\vartheta\colon G \to 0$ is handily avoided as $f\colon A \to B \otimes G$ is then equivalent to $A \xrightarrow{f} B \otimes G \xrightarrow{\mathrm{id} \otimes \vartheta} B \otimes 0 \xrightarrow{\mathrm{id} \otimes !} B \otimes I$.

In the remaining cases, inhabited garbage can be constructed when $\mathbf{C}$ is *semisimple*, meaning that any object is isomorphic to one built out of copies of the tensor unit I using $\oplus$ and $\otimes$. This is the case for **Unitary**. In this case, canonical inhabitants, i.e., morphisms $I \to G$ for each inhabited object G, can be constructed as the interpretation of the $\mathcal{U}\Pi_a$ combinators given in Section **??**.

Lemma 15. *If* $\mathbf{C}$ *is a semisimple rig category, then* $L[R[\mathbf{C}]]$ *is a binoidal category under* $\oplus$.

Proposition 16. *If* $\mathbf{C}$ *is a semisimple rig category,* $\mathcal{E}\colon R[\mathbf{C}] \to L[R[\mathbf{C}]]$ *is an arrow with choice.*

Combining the previous Proposition with Lemma **??** shows that $\mathcal{E} \circ \mathcal{D}\colon \mathbf{C} \to L[R[\mathbf{C}]]$ is an arrow with choice. The categorical semantics of $\mathcal{U}\Pi$, $\mathcal{U}\Pi_a$, and $\mathcal{U}\Pi_a^{\chi}$, including certain derived combinators with structural or otherwise significant interpretations, are summarised in Figure **??**.

Base types

$$[\![0]\!] = O \quad [\![1]\!] = I \quad [\![b + b']\!] = [\![b]\!] \oplus [\![b']\!] \quad [\![b \times b']\!] = [\![b]\!] \otimes [\![b']\!]$$

Semantics of $\mathcal{U}\Pi$ $\quad [\![b_1 \leftrightarrow b_2]\!] = [\![b_1]\!] \to [\![b_2]\!]$ in $\mathbf{C}$

$$\begin{array}{lllll} [\![swap^+]\!] = \sigma^\oplus & [\![unit^+]\!] = \rho^\oplus & [\![uniti^+]\!] = (\rho^\oplus)^{-1} & [\![assoc^+]\!] = \alpha^\oplus & [\![associ^+]\!] = (\alpha^\oplus)^{-1} \\ [\![swap^\times]\!] = \sigma^\otimes & [\![unit^\times]\!] = \rho^\otimes & [\![uniti^+]\!] = (\rho^\otimes)^{-1} & [\![assoc^\times]\!] = \alpha^\otimes & [\![associ^\times]\!] = (\alpha^\otimes)^{-1} \\ [\![id]\!] = \mathrm{id} & [\![distrib]\!] = \delta & [\![distribi]\!] = \delta^{-1} & [\![distribo]\!] = \delta_0 & [\![distriboi]\!] = \delta_0^{-1} \end{array}$$

$$[\![c_1 \fatsemi c_2]\!] = [\![c_2]\!] \circ [\![c_1]\!] \quad [\![c_1 + c_2]\!] = [\![c_1]\!] \oplus [\![c_2]\!] \quad [\![c_1 \times c_2]\!] = [\![c_1]\!] \otimes [\![c_2]\!] \quad [\![inv(c)]\!] = [\![c]\!]^{-1}$$

In **Unitary**:

$$[\![phase_\varphi]\!] = (\varphi) \quad [\![hadamard]\!] = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Semantics of $\mathcal{U}\Pi_a$ $\quad [\![b_1 \rightarrowtail b_2]\!] = [\![b_1]\!] \to [\![b_2]\!]$ in $R[\mathbf{C}]$

$$\begin{array}{lll} [\![lift\ u]\!] = [[\![u]\!]]_{\sim_R} & [\![arr\ v]\!] = \mathcal{D}([\![v]\!]) & [\![c_1 \ggg c_2]\!] = [\![c_2]\!] \circ [\![c_1]\!] \\ [\![c_1 \mathbin{***} c_2]\!] = [\![c_1]\!] \otimes [\![c_2]\!] & [\![c_1 \mathbin{+++} c_2]\!] = [\![c_1]\!] \oplus [\![c_2]\!] & [\![first\ c]\!] = [\![c]\!] \otimes \mathrm{id} \\ [\![second\ c]\!] = \mathrm{id} \otimes [\![c]\!] & [\![left\ c]\!] = [\![c]\!] \oplus \mathrm{id} & [\![right\ c]\!] = \mathrm{id} \oplus [\![c]\!] \\ [\![alloc]\!] = O \xrightarrow{!} [\![b_2]\!] & [\![inl]\!] = \Pi_1 & [\![inr]\!] = \Pi_2 \end{array}$$

Semantics of $\mathcal{U}\Pi_a^\chi$ $\quad [\![b_1 \rightsquigarrow b_2]\!] = [\![b_1]\!] \to [\![b_2]\!]$ in $L[R[\mathbf{C}]]$

$$\begin{array}{lll} [\![lift\ v]\!] = [[\![v]\!]]_{\sim_L} & [\![arr\ v]\!] = \mathcal{E}([\![v]\!]) & [\![c_1 \ggg c_2]\!] = [\![c_2]\!] \circ [\![c_1]\!] \\ [\![c_1 \mathbin{***} c_2]\!] = [\![c_1]\!] \otimes [\![c_2]\!] & [\![c_1 \mathbin{+++} c_2]\!] = [\![right\ c_2]\!] \circ [\![left\ c_1]\!] & [\![first\ c]\!] = [\![c]\!] \otimes \mathrm{id} \\ [\![second\ c]\!] = \mathrm{id} \otimes [\![c]\!] & [\![left\ c]\!] = [\![c]\!] \oplus [\![b_1']\!] & [\![right\ c]\!] = [\![b_1']\!] \oplus [\![c]\!] \\ [\![discard]\!] = [\![b_1]\!] \xrightarrow{!} I & [\![fst]\!] = \pi_1 & [\![snd]\!] = \pi_2 \end{array}$$

FIGURE 7. The categorical semantics of $\mathcal{U}\Pi$, $\mathcal{U}\Pi_a$, and $\mathcal{U}\Pi_a^\chi$ in summary, including the semantics of various derived combinators (marked yellow). The semantics of $phase_\varphi$ and *hadamard* in $\mathcal{U}\Pi$ refer to their canonical semantics in **Unitary**. In $\mathcal{U}\Pi_a^\chi$, b_1' in the semantics of *left* c and *right* c refer to the type of the alternate choice; e.g., when c has type $b_1 \rightsquigarrow b_2$, *left* c has type $b_1 + b_1' \rightsquigarrow b_2 + b_1'$.

5. Properties of the categorical constructions

The previous section developed enough properties of our categorical semantics to interpret $\mathcal{U}\Pi_a$. But the L- and R-constructions have more properties, that give them the status of a very useful generic construction. From these universal properties it follows that two fundamental embeddings in reversible computing and quantum computing are both captured by our categorical semantics.

5.1. **Universal properties.** The first insight is the following factorisation lemma, that brings morphisms in $L[\mathbf{C}]$ and $R[\mathbf{C}]$ in a normal form in terms of pure morphisms from $\mathbf{C}$.

Lemma 17. *If $\mathbf{C}$ is a rig category, then:*

(i) any map $A \to B$ in $R[\mathbf{C}]$ is represented by $A \xrightarrow{\Pi_1} A \oplus H \xrightarrow{\mathcal{D}(f)} B$ for some $f\colon A \oplus H \to B$ in $\mathbf{C}$;

(ii) any map $A \to B$ in $L[\mathbf{C}]$ is represented by $A \xrightarrow{\mathcal{E}(f)} B \otimes G \xrightarrow{\pi_1} B$ for some $f\colon A \to B \otimes G$ in $\mathbf{C}$.

These factorisations are unique.

Proof. Point (ii) follows from [**?**, Lemma 8], see [**?**, Lemma 15]. Point (i) then follows from (ii) by Proposition **??**. □

For any object A there is a unique morphism $A \to I$ in $L[\mathbf{C}]$, represented by the unitor $A \to I \otimes A$ in $\mathbf{C}$, that simply considers the input as garbage and does nothing else. This makes $L[\mathbf{C}]$ affine, and in fact it is the universal affine category including $\mathbf{C}$.

Theorem 18. *$L[\mathbf{C}]$ is the affine completion of a monoidal category $\mathbf{C}$: for an affine monoidal category $\mathbf{D}$ and a monoidal functor $F\colon \mathbf{C} \to \mathbf{D}$ there is a unique monoidal functor $\hat{F}\colon L[\mathbf{C}] \to \mathbf{D}$ with $F = \hat{F} \circ \mathcal{E}$.*

Proof. This is an easy generalisation of [**?**, Theorem 16]. □

Theorem 19. *$R[\mathbf{C}]$ is the coaffine completion of a rig category $\mathbf{C}$: for any coaffine rig category $\mathbf{D}$ and rig functor $F\colon \mathbf{C} \to \mathbf{D}$ there is a unique rig functor $\hat{F}\colon R[\mathbf{C}] \to \mathbf{D}$ such that $F = \hat{F} \circ \mathcal{D}$.*

Proof. By Proposition **??** and Theorem **??**, there is a unique functor $\hat{F}$ making the triangle commute that is monoidal with respect to $\oplus$. Lemmas **??** and **??** show that it is also monoidal with respect to $\otimes$, that is, a rig functor. □

It follows immediately that the L- and R-constructions are functorial: a monoidal functor $\mathbf{C} \to \mathbf{D}$ induces a monoidal functor $L[\mathbf{C}] \to L[\mathbf{D}]$, and a rig functor $\mathbf{C} \to \mathbf{D}$ induces a rig functor $R[\mathbf{C}] \to R[\mathbf{D}]$. The combination $L[R[\mathbf{C}]]$ is more than the sum of Theorems **??** and **??**. The following lemma shows that it is also universal for arrows with choice as in Section **??**.

Lemma 20. *Let $\mathbf{C}$ be a category with an initial object.*

(i) The category $L[\mathbf{C}]$ has an initial object, and $\mathcal{E}\colon \mathbf{C} \to L[\mathbf{C}]$ preserves it.
(ii) If $F\colon \mathbf{C} \to \mathbf{D}$ preserves the initial object then so does $\hat{F}\colon L[\mathbf{C}] \to \mathbf{D}$.
(iii) If $F\colon \mathbf{C} \to \mathbf{D}$ preserves injections then so does $\hat{F}\colon L[\mathbf{C}] \to \mathbf{D}$.

Proof. For (i), observe that any morphism $0 \to A$ in $L[\mathbf{C}]$ must be represented by the unique morphism $0 \to A \otimes G$ in $\mathbf{C}$ for some G, and these are all equivalent under $\sim_L$. By construction $\mathcal{E}(0) = 0$, see Proposition **??**. Points (ii) and (iii) now follow immediately. □

5.2. **Toffoli and Stinespring.** A central question of foundational importance for reversible modes of computing concerns that of *reversible expressivity*. Any irreversible computing machine is trivially able to simulate a reversible one – after all, reversible operations are just ordinary operations which happen to be invertible – but what is lost by considering *only* the reversible ones? Fortunately for the viability of reversible modes of computing, the answer turns out to be "nothing at all," so long as one is willing to accept some *ancillary inputs* and *garbage outputs* to occur.

5.2.1. *The fundamental theorem of classical reversible computing.* In the case of classical reversible circuit logic, Toffoli found this question to be of so supreme importance to his theory that he dubbed the expressivity theorem the *fundamental theorem* [**?**] of classical reversible computing. Toffoli showed this by demonstrating that any finite function ϕ can be simulated by a network consisting of an encoder, a finite bijective function f, and a decoder. This encoder and decoder should be *"essentially independent of ϕ and contain as little "computing power" as possible,"* [**?**] and though this characterisation is (perhaps intentionally) vague, Toffoli goes on to show that one can always choose a "trivial encoder" (an injection) and a "trivial decoder" (a projection).

It is perhaps surprising that this highly specific statement (and its proof) on the nature of finite functions, down to decomposition of a finite function as an injection, a bijective function, and a projection, can be formulated as the following purely categorical statement:

Theorem 21. *The monoidal functor $L[R[\mathbf{FinBij}]] \to \mathbf{FinSet}$ induced by the (monoidal) inclusion functor $\mathbf{FinBij} \to \mathbf{FinSet}$ is full.*

To clarify, the inclusion functor $\mathbf{FinBij} \xrightarrow{I} \mathbf{FinSet}$ acts as the identity on both objects and morphisms, noting that any finite bijection is trivially also a finite function. The monoidal functor $L[R[\mathbf{FinBij}]] \to \mathbf{FinSet}$ then arises by successively applying Theorems **??** and **??**, as in

$$\begin{array}{ccccc} \mathbf{FinBij} & \xrightarrow{\mathcal{D}} & R[\mathbf{FinBij}] & \xrightarrow{\mathcal{E}} & L[R[\mathbf{FinBij}]] \\ & \searrow^{I} & \downarrow{\scriptstyle \check{I}} & & \downarrow{\scriptstyle \hat{I}} \\ & & \mathbf{FinSet} & \xrightarrow[\mathrm{id}]{} & \mathbf{FinSet} \end{array}$$

Notice that it follows from Lemma **??** that any morphism of $L[R[\mathbf{FinBij}]]$ factors (essentially uniquely) as $A \xrightarrow{\mathcal{E}(\amalg_1)} A \oplus H \xrightarrow{\mathcal{E}(\mathcal{D}(f))} B \otimes G \xrightarrow{\pi_1} B$ for some bijection $A \oplus H \xrightarrow{f} B \otimes G$. Recall that all morphisms in **FinBij** are isomorphisms, and that all functors preserve isomorphisms. Using Lemma **??** and noting that the inclusion $\mathbf{FinBij} \xrightarrow{I} \mathbf{FinSet}$ trivially preserves injections (and projections), it follows that any function in the image of $\hat{\check{I}}$ factors as

$$A \xrightarrow[\textit{injection}]{\amalg_1} A \oplus H \xrightarrow[\textit{bijection}]{\hat{\check{I}}(\mathcal{E}(\mathcal{D}(f)))} B \otimes G \xrightarrow[\textit{projection}]{\pi_1} B$$

What the fundamental theorem of reversible computing then states is that *all* functions are, in fact, in the image of $\hat{\hat{I}}$, and so permit such a factorisation. That is exactly to say that $\hat{\hat{I}}$ is full.

5.2.2. *Stinespring's dilation theorem.* Similarly, in the setting of quantum theory, the L- and R-constructions capture Stinespring's dilation theorem. As discussed in Section **??**, this theorem shows that any mixed quantum operation can be modeled by a pure quantum operation, so long as one is willing to enlarge the situation with an auxiliary system. Write **Isometry** for the category of Hilbert spaces and isometric linear functions, and **Unitary** for the subcategory of unitary linear functions.

Theorem 22 (Huot & Staton)**.** *There is an equivalence* $R[\mathbf{Unitary}] \simeq \mathbf{Isometry}$ *of rig categories, and an equivalence* $L[R[\mathbf{Unitary}]] \simeq \mathbf{CPTP}$ *of monoidal categories.*

Proof. See [**?**, **?**] and [**?**]. □

Notice that the previous theorem holds for Hilbert space that are not necessarily finite-dimensional.

6. Applications

We will now consider some areas where our results can be put to work. First, we argue that our categorical model can be used to prove useful, nontrivial properties about measurement entirely algebraically, without ever needing to consider the gritty details of quantum channels. Second, we illustrate the use of $\mathcal{U}\Pi_a^\chi$ as a metalanguage by providing a translation from Selinger's quantum flowhcharts [**?**] to $\mathcal{U}\Pi_a^\chi$.

6.1. **Properties of measurement.** In this section, we will use type subscripts on combinators whenever it is necessary to disambiguate between definitions, or to make the presentation clearer. For example, we will write $measure_{b+b'}$ to mean the measurement combinator on the type $b + b' \rightsquigarrow b + b'$.

Our first property is bit technical, concerning the behaviour of injections with measurement. We will see shortly how it can be used to prove far more interesting things. It's proof is in the Supplementary Material.

Proposition 23. *Measurement commutes with injections:* $[\![measure_b \ggg inl]\!] = [\![inl \ggg measure_{b+b'}]\!]$*, and likewise for* inr*.*

Any complex, finite dimensional Hilbert space is isomorphic to one of the form $\mathbb{C}^n$, where n is its dimension. From this it follows that each canonical injection $\Pi_i : \mathbb{C} \to \mathbb{C}^n$ is associated with a distinct vector $|i\rangle$ linearly independent from all other $|j\rangle$ for $1 \leq j \leq n$, $j \neq i$. Together, these are precisely the classical states forming the computational basis of $\mathbb{C}^n$.

Abstracting, we say that a classical state is nothing but an injection, i.e., composition of inl and inr. This intuition is correct, in that we can show that measurement does nothing to classical states:

Proposition 24. *If* s *is a classical state then* $[\![s \ggg measure]\!] = [\![s]\!]$*.*

Proof. We first see that measurement on 1 is the identity: $measure_1 = clone_1 \ggg fst = uniti^\times \ggg (id \times discard) \ggg unit^\times$. Since $[\![discard]\!]$ when applied to $[\![1]\!] = I$ is nothing but the identity by I terminal, it follows that

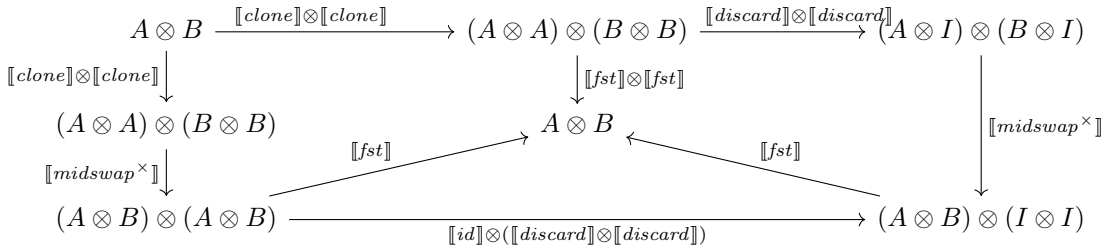
$$\begin{aligned}[\![measure_1]\!] &= [\![uniti^\times \ggg (id \times discard) \ggg unit^\times]\!] = [\![unit^\times]\!] \circ [\![id \times discard]\!] \circ [\![uniti^\times]\!] \\ &= [\![unit^\times]\!] \circ [\![uniti^\times]\!] = [\![id]\!] \ .\end{aligned}$$

Now, since a classical state $s : 1 \rightsquigarrow b$ is precisely an injection, it follows by Proposition **??** that $[\![s \ggg measure_b]\!] = [\![measure_1 \ggg s]\!] = [\![s]\!] \circ [\![measure_1]\!] = [\![s]\!]$. □

A very useful property of measurement is that the result of measuring a joint system is nothing but the product of measurements on each constituent system individually. This is shown as follows:

Proposition 25. *Measurement of products is the product of measurements:* $[\![measure_{b\times b'}]\!] = [\![measure_b \times measure_{b'}]\!]$*.*

Proof. Using the fact that $measure = clone \ggg fst$ and $clone_{b\times b'} = clone_b \times clone_{b'} \ggg midswap^\times$, the property follows by naturality of $[\![midswap^\times]\!]$:



Again, A and B range over interpretations of arbitrary $\mathcal{U}\Pi_a^\chi$ types b and b'. □

An immediate consequence of this property is that measurements also commute with projections:

Proposition 26. *Measurement on a product type commutes with projections:* $[\![measure_{b\times b'} \ggg fst]\!] = [\![fst \ggg measure_b]\!]$ *and likewise for snd.*

Proof. Since $[\![measure_{b\times b'}]\!] = [\![measure_b \times measure_{b'}]\!] = [\![measure_b]\!] \otimes [\![measure_{b'}]\!]$ by Proposition **??**, it follows by naturality of $\pi_1 = [\![fst]\!]$ that

$$[\![measure_{b\times b'} \ggg fst]\!] = \pi_1 \circ [\![measure_b]\!] \otimes [\![measure_{b'}]\!] = [\![measure_b]\!] \circ \pi_1 = [\![fst \ggg measure_b]\!] \ .$$

which was what we wanted. □

The final property we want to show is that measurement is idempotent. Conceptually, this can be seen as an extension to the property that measurement does nothing to classical states. This is because the result of measuring a quantum state will always be a mixed classical state, so further measuring this has no effect. To do this, we will need to remark that it can be shown that cloning is coassociative.

Proposition 27. *Cloning is associative:* $[\![clone \ggg (clone \times id) \ggg assoc^\times]\!] = [\![clone \ggg (id \times clone)]\!]$.

We can then show idempotence:

Proposition 28. *Measurement is idempotent:* $[\![measure \ggg measure]\!] = [\![measure]\!]$.

Proof. Since $[\![fst]\!] = [\![lift\ id]\!]$, it can be shown that $[\![measure]\!] = [\![lift\ clone]\!]$, so by the definition of $\ggg$ in $\mathcal{U}\Pi_a^\chi$ we have that

$$[\![measure \ggg measure]\!] = [\![lift(clone \ggg (clone \times id) \ggg assoc^\times)]\!] = [\![lift(clone \ggg (id \times clone))]\!],$$

the final equality following from coassociativity of cloning (Proposition **??**). But then $[\![id \times clone]\!] = \mathrm{id} \otimes [\![clone]\!]$ mediates between $[\![clone]\!]$ and $[\![clone \ggg (id \times clone)]\!]$ in $R[\mathbf{C}]$, so $[\![measure]\!] = [\![lift\ clone]\!] = [\![lift\ (clone \ggg (id \times clone))]\!] = [\![measure \ggg measure]\!]$ in $L[R[\mathbf{C}]]$. □

6.2. **Quantum flow charts.** In this section, we demonstrate the translation of (noniterative) quantum flow charts into $\mathcal{U}\Pi_a^\chi$. As the name suggests, quantum flow charts are the quantum extension of the classical imperative flow chart languages, used extensively in areas such as program compilation and partial evaluation [**?**, **?**]. They were first considered by Selinger in several variations [**?**]: here, we consider the purely quantum variant, which has only quantum data, in the form of qubit ensembles. The only type of data supported is the type *Qbit* of qubits, so typing contexts Γ are simply given by lists of active variables, which are all of *Qbit* type. In its textual form, a quantum flowchart is simply a list of commands. The supported commands are as follows (with q ranging over variables):

$$c ::= new\ qbit\ q := 0 \mid discard\ q \mid q \mathrel{*}= U \mid permute\ \varphi \mid initial \mid measure\ q \mid merge \mid c; c \mid c \oplus c$$

Commands take sums of typing contexts to sums of typing contexts, each summand denoting a program branch. Briefly, *new qbit* and *discard* allocate and discard new variables respectively, $q \mathrel{*}= U$ applies some unitary U to the variable q, *permute* φ changes the variable order by applying an arbitrary permutation φ, *initial* initialises an empty typing context, and *merge* merges two program branches (with the same typing context) into one. The most novel command is arguably *measure* q, which measures the qubit q and branches on the (classical) measurement result: this style of measurement-based flow control goes by the motto of "quantum data, classical control." Flow charts are composed in sequence and in parallel using ; and $\oplus$ respectively.

We begin with the translation of types and contexts, given simply by $\mathcal{T}[\![q : Qbit]\!] = 1 + 1$ and $\mathcal{T}[\![\Gamma, \Gamma']\!] = \mathcal{T}[\![\Gamma]\!] \times \mathcal{T}[\![\Gamma']\!]$. Before we proceed with the translation of commands, we will use the abuse of

notation $|0\rangle$ and $|1\rangle$ to refer to the injections $\mathrm{inl} : 1 \rightsquigarrow 1+1$ and $\mathrm{inr} : 1 \rightsquigarrow 1+1$ respectively, to indicate that these serve as allocation of the constant classical values of $|0\rangle$ and $|1\rangle$. Commands are translated as follows:

$$\begin{array}{rclrcl}
\mathcal{T}[\![new\ qbit\ q := 0]\!] & = & uniti^{\times} \ggg id \times |0\rangle & \mathcal{T}[\![discard\ q]\!] & = & fst \\
\mathcal{T}[\![q \mathrel{*}= U]\!] & = & id \times arr(arr(\hat{U})) & \mathcal{T}[\![permute\ \varphi]\!] & = & arr(arr(\hat{\varphi})) \\
\mathcal{T}[\![initial]\!] & = & alloc & \mathcal{T}[\![merge]\!] & = & merge \\
\mathcal{T}[\![c; c']\!] & = & \mathcal{T}[\![c]\!] \ggg \mathcal{T}[\![c']\!] & \mathcal{T}[\![c \oplus c']\!] & = & \mathcal{T}[\![c]\!] +\!\!+\!\!+ \mathcal{T}[\![c']\!]
\end{array}$$

$$\mathcal{T}[\![measure\ q]\!] = id \times measure \ggg distrib \ggg ((id \times |0\rangle) +\!\!+\!\!+ (id \times |1\rangle))$$

In the above, the terms $\hat{U}$ and $\hat{\varphi}$ denote the approximations of the unitary U and permutation φ (which is a particularly simple kind of unitary) respectively as $\mathcal{U}\Pi$ terms, as given by the universality theorem for $\mathcal{U}\Pi$ (Theorem **??**).

7. Conclusion and future work

We have shown how quantum measurement, often presented in somewhat mysterious terms, arises through two suprisingly simple arrow constructions, associated with the information effects of *allocation* and *hiding*, on a reversible quantum combinator language. We have provided categorical semantics for all of these languages through elementary universal constructions on *rig categories*. These let us prove useful nontrivial properties of measurement as semantic equivalences, recast the fundamental theorem of reversible computation as an elementary categorical statement, and interpret (noniterative) quantum flow charts. There are several avenues for further research:

Classical and quantum data. In the current formulation, $\mathcal{U}\Pi$ only allows forming quantum data types. Hence qubit measurement, for example, can only be given the type $Qbit \to Qbit$, rather than $Qbit \to Bit$. Extending $\mathcal{U}\Pi_a^\chi$ with classical data would address this shortcoming. Semantically, this would need a (sufficiently nice) construction of the category of C*-algebras and quantum channels from the category of Hilbert spaces and quantum channels.

Categorical semantics of SILQ. The modern quantum programming language SILQ [**?**] has several original features: annotation of *measurement-free* and *quantum-free* functions, a linear type system, and the automatic uncomputation of garbage. $\mathcal{U}\Pi_a^\chi$ enables a type-level interpretation of *measurement-free* functions (the *pure* combinators, *arr* v), and access to a canonical reversibilisation of combinators. It would be interesting to extend the $\mathcal{U}\Pi$ family with the remaining features to provide combinator semantics for SILQ, in the same way that Π^0 does in the classical case for Theseus [**?**].

Is there a quantum effect? We have shown that there are elementary arrow constructions connecting reversible and irreversible quantum computations. Is there a similar arrow construction connecting *classical* reversible computation and *quantum* reversible computation? Such a construction would likely involve several steps, such as adjoining the circle group to introduce phases, and considering a variation of convex combinations of morphisms respecting direct sums, to give morphisms that introduce and eliminate superpositions.

Recursion and subnormalised channels. The notion of quantum channel considered in this paper is too rigid to enable recursion or iteration, which is why we were only able to give semantics to the noniterative fragment of quantum flow charts. To enable recursion would require a more relaxed notion of quantum channel, from (completely positive) trace-preserving maps to trace-nonincreasing ones. Can recursion be added as an effect, in the form of an arrow construction from **CPTP** to **CPTN**?

Measurement-based quantum computation. Measurement-based quantum computation is in a sense opposite to the quantum circuit model. In the latter all operations are reversible except the very last one. In the former all operations are irreversible up to reversible corrections being fed forward. What is the relationship to the model given in this article? Is there a translation from the language of measurement patterns used in measurement-based quantum computation to $\mathcal{U}\Pi_a^\chi$?

Acknowledgements: We thank Pablo Andrés-Martínez for his comments and suggestions on this paper. Some early ideas relating to this paper were explored by the second author in conjunction with Martti Karvonen, during a short-term scientific mission to the University of Edinburgh sponsored by COST Action IC1405: *Reversible computation - extending horizons of computing.*

Appendix A. Supplementary material

Universality proofs

In classical computing, the notion of universality is binary; either a computational system is universal, and so able to express *all* classical computations, or it is not. This situation is more subtle and fine-grained in quantum computing. Since the state space of qubits alone is uncountable, it can be shown that an uncountable number of nontrivial gates is required to achieve *exact* universality. In other words, exact universality seems too strong a requirement for quantum computing.

If we're willing to accept a small approximation error, however, we can get away with not only a finite gate set, but also a rather small one, such as the Clifford+T gate set (composed of the S, T, $CNOT$, and Hadamard gate H). Precisely, given some unitary U and acceptable approximation error $\delta > 0$, we can produce a unitary $\hat{U}$ computed by a circuit composed of only Clifford+T gates, such that $\left\| U - \varphi\hat{U} \right\|_{\text{op}} < \delta$ for some phase φ. Error is measured using the *operator norm*

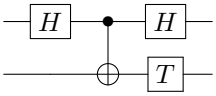
$$\|A\|_{\text{op}} = \sup\{\|A\psi\| \mid \psi \text{ with } \|\psi\| = 1\} \ .$$

where the unqualified norm $\|\cdot\|$ refers to the usual inner product norm $\|\psi\| = \sqrt{\langle\psi|\psi\rangle}$. This can be used to quantify similarity of both quantum states and operations (i.e., isometries), in that it can be shown that similar isometries produce similar evolutions, and similar states have similar measurement statistics.

Theorem 0. *$\mathcal{U}\Pi$ is approximately universal for $2^n \times 2^n$ unitaries: For any unitary U and $\delta > 0$ there exists a $\mathcal{U}\Pi$ combinator u such that $\|U - [\![u]\!]\|_{\text{op}} < \delta$.*

Proof. It is well established that, *e.g.*, the Clifford+T gate set, composed of the S, T, $CNOT$ and Hadamard gate H is approximately universal (see, e.g., [**?**]). Specifically, for any $2^n \times 2^n$ unitary U and permitted error $\delta > 0$, there exists a unitary $\hat{U}$ computed by a circuit composed entirely of Clifford+T gates, as well as a phase φ, such that $\left\| U - \varphi\hat{U} \right\|_{\text{op}} < \delta$.

Let U be some $2^n \times 2^n$ unitary and $\delta > 0$, and let $\hat{U}$ be its Clifford+T approximant with phase φ. Using the circuit representation of $\hat{U}$, we construct the $\mathcal{U}\Pi$ combinator $\hat{u}$ by replacing Clifford+T gates with their implementation in Figure **??**, using $\cdot \times \cdot$ for parallel composition and $\cdot \fatsemi \cdot$ for sequential composition. For example, the quantum circuit



is translated to the $\mathcal{U}\Pi$ term $\mathit{hadamard} \times \mathit{id} \fatsemi \mathit{cnot} \fatsemi \mathit{hadamard} \times t$.

Since this is only accurate up to a phase φ, the final combinator is $u = \varphi \bullet \hat{u}$, and we have $\|U - [\![u]\!]\|_{\text{op}} = \|U - [\![\varphi \bullet \hat{u}]\!]\|_{\text{op}} = \|U - \varphi[\![\hat{u}]\!]\|_{\text{op}} = \left\| U - \varphi\hat{U} \right\|_{\text{op}} < \delta$. □

Theorem 0. *$\mathcal{U}\Pi_a$ is approximately universal for isometries: For any $2^n \times 2^m$ isometry V and $\delta > 0$ there exists a $\mathcal{U}\Pi_a$ combinator v such that $\|V - [\![v]\!]\|_{\text{op}} < \delta$.*

Proof. Let V be some isometry and $\delta > 0$. By Theorem **??** and Lemma **??**, it follows that $V = U \circ \Pi_1$ for some unitary U. By Theorem **??** there exists for any given $\delta' > 0$ a $\mathcal{U}\Pi$ combinator u approximating U. Choose u approximating U with δ, and construct $v = \mathit{inl} \fatsemi \mathit{arr}(u)$, using the fact that any $\mathcal{U}\Pi$ combinator c can be lifted to a $\mathcal{U}\Pi_a$ combinator $\mathit{arr}(u)$ with the exact same semantics. Then

$$\begin{aligned}
\|(U \circ \Pi_1) - [\![\mathit{inl} \ggg \mathit{arr}(u)]\!]\|_{\text{op}} &= \|(U \circ \Pi_1) - ([\![\mathit{arr}(u)]\!] \circ [\![\mathit{inl}]\!])\|_{\text{op}} \\
&= \|(U \circ \Pi_1) - ([\![\mathit{arr}(u)]\!] \circ \Pi_1)\|_{\text{op}} \\
&= \|(U - [\![\mathit{arr}(u)]\!]) \circ \Pi_1\|_{\text{op}} \\
&\leq \|U - [\![\mathit{arr}(u)]\!]\|_{\text{op}} \|\Pi_1\|_{\text{op}} \\
&= \|U - [\![\mathit{arr}(u)]\!]\|_{\text{op}} = \|U - [\![u]\!]\|_{\text{op}} < \delta
\end{aligned}$$

which was what we wanted. □

Theorem 0. *$\mathcal{U}\Pi_a^\chi$ is approximately universal for quantum channels: For any quantum channel Λ and $\delta > 0$ there exists a $\mathcal{U}\Pi_a^\chi$ combinator c such that $\|\Lambda - [\![c]\!]\|_{\text{op}} < \delta$.*

Proof. By Stinespring's dilation theorem there exists a Hilbert space E (which can always be chosen to have non-zero dimension) and isometry V such that $\Lambda = \mathrm{tr}_E \circ (V(-)V^\dagger)$. By Theorem **??** we can approximate V by some $\mathcal{U}\Pi_a$ combinator v with any choice of error $\delta' > 0$. Let $\delta > 0$, and construct $c = arr(v) \ggg fst$. Then

$$[\![c]\!] = [\![arr(v) \ggg fst]\!] = [\![fst]\!] \circ [\![arr(v)]\!] = \mathrm{tr}_E \circ ([\![v]\!](-)[\![v]\!]^\dagger)$$

so $\Lambda - [\![c]\!] = \mathrm{tr}_E \circ ((V - [\![v]\!])(-)(V^\dagger - [\![v]\!]^\dagger))$. Since if $\|V - [\![v]\!]\|_{\mathrm{op}} < \delta'$ then for any ρ with $\|\rho\|_{\mathrm{op}} = 1$ we have

$$\big\|(V - [\![v]\!])\rho(V^\dagger - [\![v]\!]^\dagger)\big\|_{\mathrm{op}} \leq \|V - [\![v]\!]\|_{\mathrm{op}} \|\rho\|_{\mathrm{op}} \big\|V^\dagger - [\![v]\!]^\dagger\big\|_{\mathrm{op}} < \delta'^2$$

it follows that $\big\|(V - [\![v]\!])(-)(V^\dagger - [\![v]\!]^\dagger)\big\|_{\mathrm{op}} < \delta'^2$. Thus, if we approximate V to an error $\|V - [\![v]\!]\|_{\mathrm{op}} < \sqrt{\frac{\delta}{\|\mathrm{tr}_E\|_{\mathrm{op}}}}$, we get

$$\begin{aligned} \|\Lambda - [\![c]\!]\|_{\mathrm{op}} &= \big\|\mathrm{tr}_E \circ ((V - [\![v]\!])(-)(V^\dagger - [\![v]\!]^\dagger))\big\|_{\mathrm{op}} \\ &\leq \|\mathrm{tr}_E\|_{\mathrm{op}} \big\|(V - [\![v]\!])(-)(V^\dagger - [\![v]\!]^\dagger)\big\|_{\mathrm{op}} \\ &< \|\mathrm{tr}_E\|_{\mathrm{op}} \sqrt{\frac{\delta}{\|\mathrm{tr}_E\|_{\mathrm{op}}}}^2 = \delta \end{aligned}$$

which was what we wanted. □

Notice the use of the operator norm in the above. It can be argued that the operator norm is not suitable for completely positive maps, as it is not robust with respect to tensor products. That is, even for a completely positive map Λ with small operator norm, $\|\Lambda \otimes \mathrm{id}\|_{\mathrm{op}}$ may be very large. For this reason, it is preferable to use the *cb-norm* instead, $\|\Lambda\|_{\mathrm{cb}} = \sup \|\Lambda \otimes \mathrm{id}_n\|_{\mathrm{op}}$. However, this is not an issue issue in this case, for if Λ is approximated by $[\![c]\!]$ up to $\delta > 0$ then $[\![c \times \mathrm{id}]\!] = [\![c]\!] \otimes \mathrm{id}$ also approximates $\Lambda \otimes \mathrm{id}$ with the same error δ. Even further, since all norms are equivalent in the finite dimensional case, there exists a constant N such that $\|\Lambda\|_{\mathrm{cb}} \leq N\|\Lambda\|_{\mathrm{op}}$ for any choice of Λ. As a consequence, any Λ can in fact be approximated up to an arbitrary δ with the cb-norm by approximating it up to $\frac{\delta}{N}$ with the operator norm.

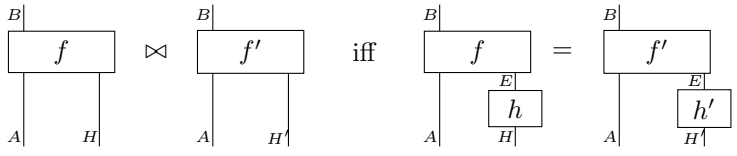
Categorical proofs

Lemma 0. *Let $(\mathbf{C}, \oplus)$ be a monoidal category. If $f \sim_R f'$ and $g \sim_R g'$, then:*

(i) $g \circ f \sim_R g' \circ f'$ if g and f are composable;
(ii) $f \oplus g \sim_R f' \oplus g'$;

Dually $g \circ f \sim_L g' \circ f'$ and $f \otimes g \sim_L f' \otimes g'$ when $f \sim_L f'$ and $g \sim_L g'$ in a monoidal category $(\mathbf{C}, \otimes)$.

To describe the equivalence relation generated by equation (**??**), define an auxiliary relation



for some morphisms h and h'. This relation $\bowtie$ is reflexive and symmetric, but may not be transitive (as we now cannot simply compose mediators). Nevertheless, $f \sim_R g$ if and only if there exists a finite sequence $f = f_1 \bowtie \cdots \bowtie f_n = g$.

Proof. Let $f = f_1 \bowtie \cdots \bowtie f_m = f'$ and $g = g_1 \bowtie \cdots \bowtie g_n = g'$. We may assume without loss of generality that $n = m$ by using reflexivity of $\bowtie$ to pad the shorter sequence with copies of the final morphisms if need be. By induction, it suffices to prove the case $n = 2$. Assume that $f_1 \bowtie f_2$ is mediated by h_1 and h_2, and $g_1 \bowtie g_2$ is mediated by h_1' and h_2'.

Now (i) follows from naturality of α:

$$\begin{array}{ccccc}
A \oplus (H_1 \oplus H_1') & \xrightarrow{\alpha} & (A \oplus H_1) \oplus H_1' & \xrightarrow{f_1 \oplus \mathrm{id}} & B \oplus H_1' \\
{\scriptstyle \mathrm{id} \oplus (h_1 \oplus h_1')}\uparrow & & \uparrow{\scriptstyle (\mathrm{id} \oplus h_1) \oplus h_1'} & & \downarrow{\scriptstyle g_1} \\
A \oplus (E \oplus E') & \xrightarrow{\alpha} & (A \oplus E) \oplus E' & & C \\
{\scriptstyle \mathrm{id} \oplus (h_2 \oplus h_2')}\downarrow & & \downarrow{\scriptstyle (\mathrm{id} \oplus h_2) \oplus h_2'} & & \uparrow{\scriptstyle g_2} \\
A \oplus (H_2 \oplus H_2') & \xrightarrow[\alpha]{} & (A \oplus H_2) \oplus H_2' & \xrightarrow[f_2 \oplus \mathrm{id}]{} & B \oplus H_2'
\end{array}$$

Similarly (ii) follows from naturality of the coherence isomorphism s below:

$$\begin{array}{ccccc}
(A \oplus A') \oplus (H_1 \oplus H_1') & \xrightarrow{s} & (A \oplus H_1) \oplus (A' \oplus H_1') & & \\
{\scriptstyle \mathrm{id} \oplus (h_1 \oplus h_1')}\uparrow & & {\scriptstyle (\mathrm{id} \oplus h_1) \oplus (\mathrm{id} \oplus h_1')}\uparrow & \searrow^{f_1 \oplus g_1} & \\
(A \oplus A') \oplus (E \oplus E') & \xrightarrow{s} & (A \oplus E) \oplus (A' \oplus E') & & B \oplus B' \\
{\scriptstyle \mathrm{id} \oplus (h_2 \oplus h_2')}\downarrow & & {\scriptstyle (\mathrm{id} \oplus h_2) \oplus (\mathrm{id} \oplus h_2')}\downarrow & \nearrow_{f_1' \oplus g_1'} & \\
(A \oplus A') \oplus (H_2 \oplus H_2') & \xrightarrow[s]{} & (A \oplus H_2) \oplus (A' \oplus H_2') & &
\end{array}$$

This concludes the proof. □

Lemma 0. *If* $\mathbf{C}$ *is a rig category, then so is* $R[\mathbf{C}]$.

Proof. The monoidal structure $(\oplus, 0)$ is inherited from $\mathbf{C}$ straightforwardly. More intricately, $R[\mathbf{C}]$ is monoidal with $(\otimes, I)$ inherited from $\mathbf{C}$:

- the tensor product of objects is $A \otimes B$;
- the tensor unit is I;
- the tensor product of morphisms $f\colon A \oplus H \to B$ and $f'\colon A' \oplus H' \to B'$ is

$$(A \otimes A') \oplus H'' \xrightarrow{\delta^{-1}} (A \oplus H) \otimes (A' \oplus H') \xrightarrow{f \otimes f'} B \otimes B'$$

  where $H'' = (H \otimes A') \oplus (A \otimes H') \oplus (H \otimes H')$;

To see that the tensor product is well-defined, follow the same induction as in the proof of Lemma **??**. Assume that $f_1 \bowtie f_2$ mediated by h_1 and h_2, and $g_1 \bowtie g_2$ mediated by h_1' and h_2'. Define:

$$\begin{aligned}
E'' &= (E \otimes A') \oplus (A \otimes E') \oplus (E \otimes E') \\
H_1'' &= (H_1 \otimes A') \oplus (A \otimes H_1') \oplus (H_1 \otimes H_1') \\
H_2'' &= (H_2 \otimes A') \oplus (A \otimes H_2') \oplus (H_2 \otimes H_2') \\
h_1'' &= (h_1 \otimes \mathrm{id}) \oplus (\mathrm{id} \otimes h_1') \oplus (h_1 \otimes h_1')\colon E'' \to H_1'' \\
h_2'' &= (h_2 \otimes \mathrm{id}) \oplus (\mathrm{id} \otimes h_2') \oplus (h_2 \otimes h_2')\colon E'' \to H_2''
\end{aligned}$$

and observe that

$$\begin{array}{ccccc}
(A \otimes A') \oplus H_1'' & \xrightarrow{\delta} & (A \oplus H_1) \otimes (A' \oplus H_1') & & \\
{\scriptstyle h_1''}\uparrow & & {\scriptstyle (\mathrm{id} \oplus h_1) \otimes (\mathrm{id} \oplus h_1')}\uparrow & \searrow^{f_1 \otimes g_1} & \\
(A \otimes A') \oplus E'' & \xrightarrow{\delta} & (A \oplus E) \otimes (A' \oplus E') & & B \otimes B' \\
{\scriptstyle h_2''}\downarrow & & {\scriptstyle (\mathrm{id} \oplus h_2) \otimes (\mathrm{id} \oplus h_2')}\downarrow & \nearrow_{f_2 \otimes g_2} & \\
(A \otimes A') \oplus H_2'' & \xrightarrow[\delta]{} & (A \oplus H_2) \otimes (A' \oplus H_2') & &
\end{array}$$

commutes by naturality of δ. Hence $f_1 \otimes g_1 \bowtie f_2 \bowtie g_2$.

Associators and unitors for $\otimes$ in $R[\mathbf{C}]$ are inherited from $\mathbf{C}$. For example, the right unitor is the composition of $\rho^{\oplus}\colon (A \otimes I) \oplus 0 \to A \otimes I$ and $\rho^{\otimes}\colon A \otimes I \to A$. These are natural by naturality of the distributors δ.

To see that the required coherence diagrams in $R[\mathbf{C}]$ commute, observe that if $f = \mathcal{D}(g)$ and $f' = \mathcal{D}(g')$, then $H'' = (0 \otimes A') \oplus (A \otimes 0) \oplus (0 \oplus 0) \cong 0$. By Proposition **??**, $f \otimes f'$ in $R[\mathbf{C}]$ is therefore equivalent to

$$(A \otimes A') \oplus 0 \xrightarrow{\rho^{\oplus}} A \otimes A' \xrightarrow{{\rho^{\oplus}}^{-1} \otimes {\rho^{\oplus}}^{-1}} (A \oplus 0) \otimes (A' \oplus 0) \xrightarrow{\rho^{\oplus} \otimes \rho^{\oplus}} A \oplus A' \xrightarrow{g \otimes g'} B \otimes B'$$

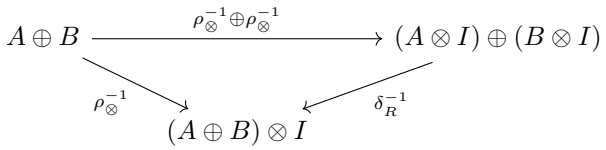
in $\mathbf{C}$, which equals $\mathcal{D}(g \otimes g')$. Likewise, it follows by Proposition **??** that if $f = \mathcal{D}(g)$ and $f' = \mathcal{D}(g')$ then $f \oplus f' = \mathcal{D}(g) \oplus \mathcal{D}(g') = \mathcal{D}(g \oplus g')$. Since coherence diagrams $R[\mathbf{C}]$ thus only ever involve morphisms in the image of $\mathcal{D}$, rig coherence follows from that of $\mathbf{C}$. □

Lemma 0. *If $\mathbf{C}$ is a semisimple rig category, then $L[R[\mathbf{C}]]$ is a binoidal category under $\oplus$.*

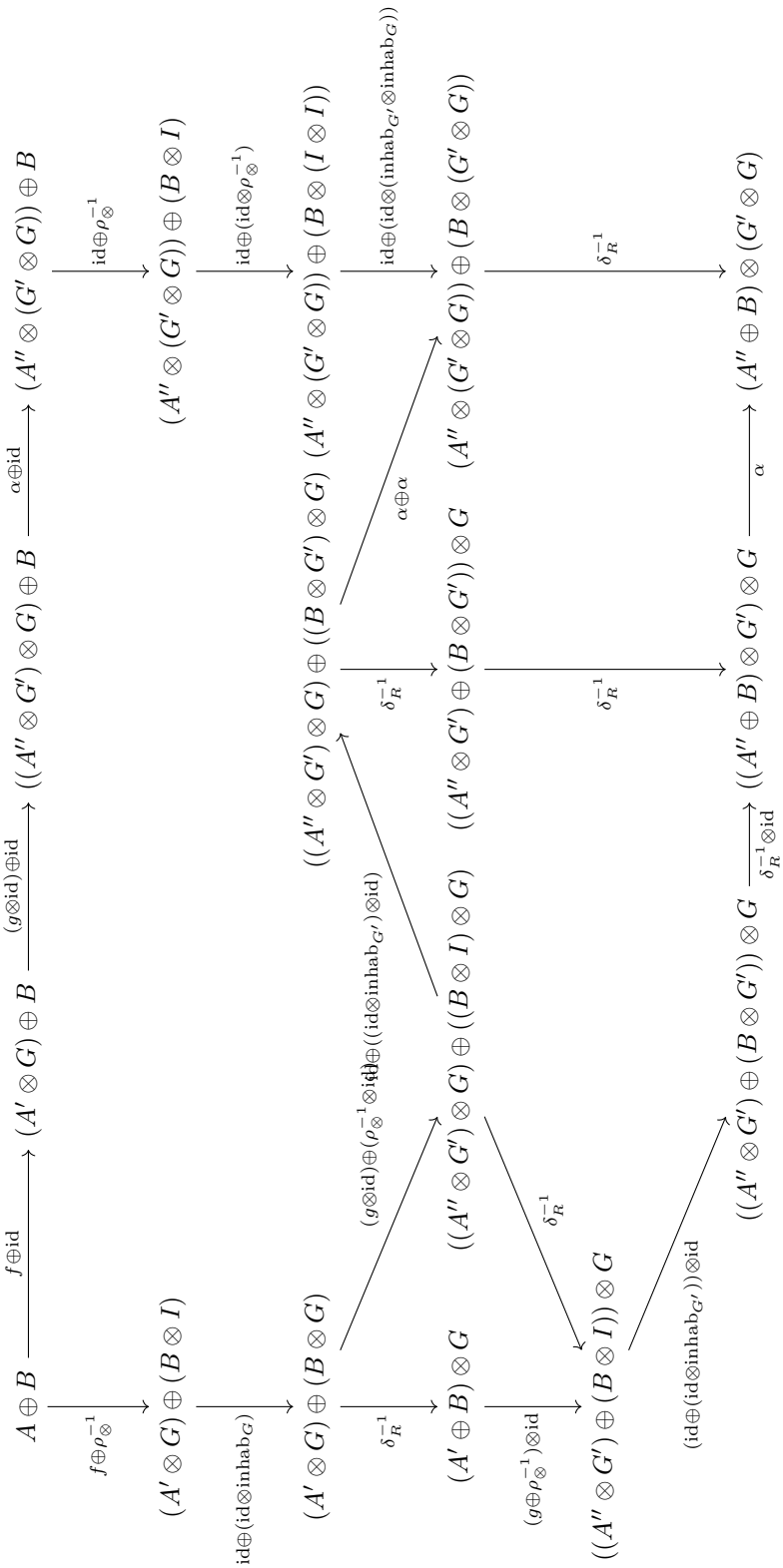
Proof. Define the functor $- \oplus B \colon L[R[\mathbf{C}]] \to L[R[\mathbf{C}]]$ on objects by $A \mapsto A \oplus B$, and on a morphism $A \to A'$ in $L[R[\mathbf{C}]]$ given by $f \colon A \to A' \otimes G$ in $R[\mathbf{C}]$ as:

$$A \oplus B \xrightarrow{f \oplus \rho_\times^{-1}} (A' \otimes G) \oplus (B \otimes I) \xrightarrow{\mathrm{id} \oplus (\mathrm{id} \otimes \mathrm{inhab}_G)} (A' \otimes G) \oplus (B \otimes G) \xrightarrow{\delta_R^{-1}} (A' \oplus B) \otimes G$$

That $- \oplus B$ preserves identities in $L[R[\mathbf{C}]]$ comes down to the following diagram in $\mathbf{C}$:



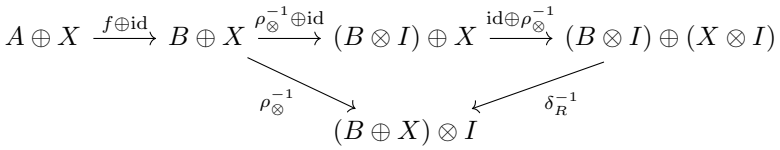
This holds by coherence. That $(g \oplus B) \circ (f \oplus B) = (g \circ f) \oplus B$ in $L[R[\mathbf{C}]]$ unfolds to the following diagram in $\mathbf{C}$:



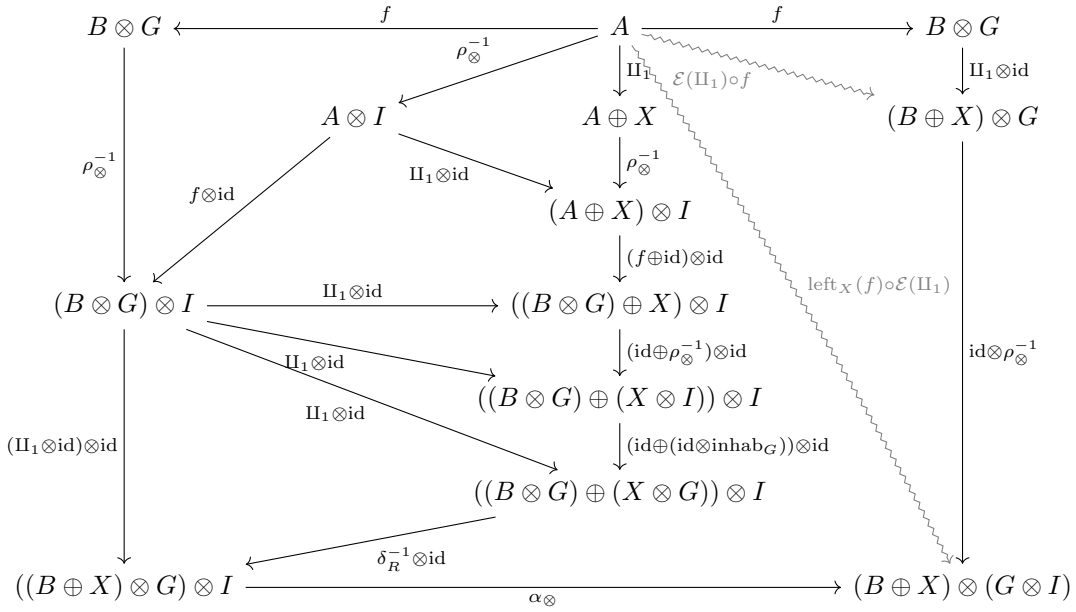
The functors $A \oplus -\colon L[R[\mathbf{C}]] \to L[R[\mathbf{C}]]$ are defined symmetrically. □

Proposition 0. *If* $\mathbf{C}$ *is a semisimple rig category,* $\mathcal{E}\colon R[\mathbf{C}] \to L[R[\mathbf{C}]]$ *is an arrow with choice.*

Proof. To see (**??**):

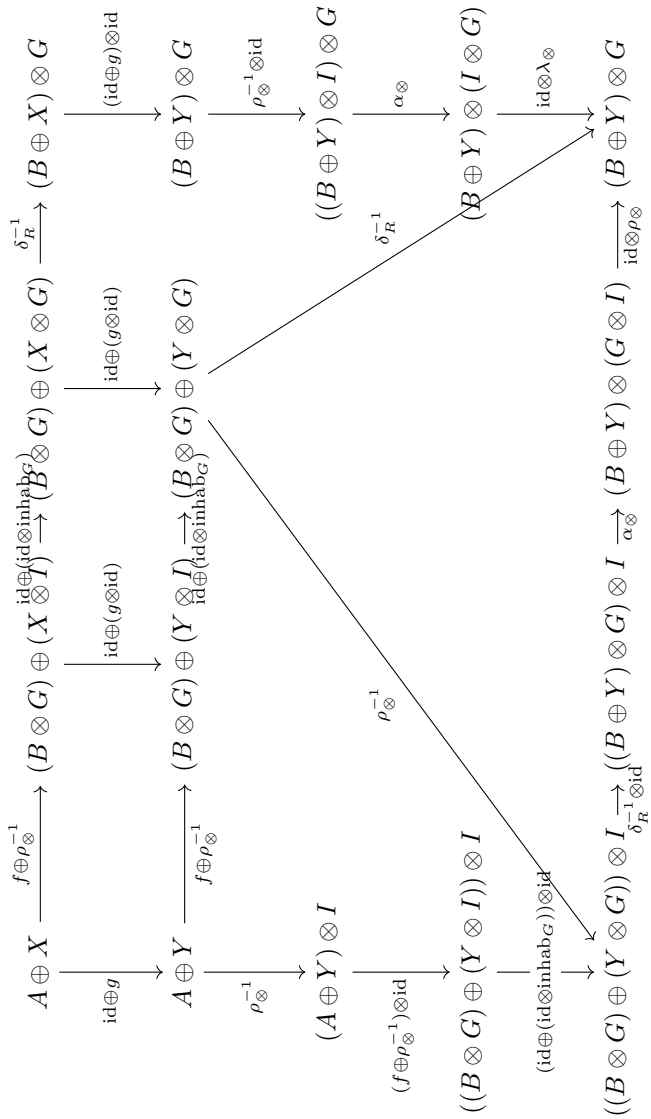


For (**??**):

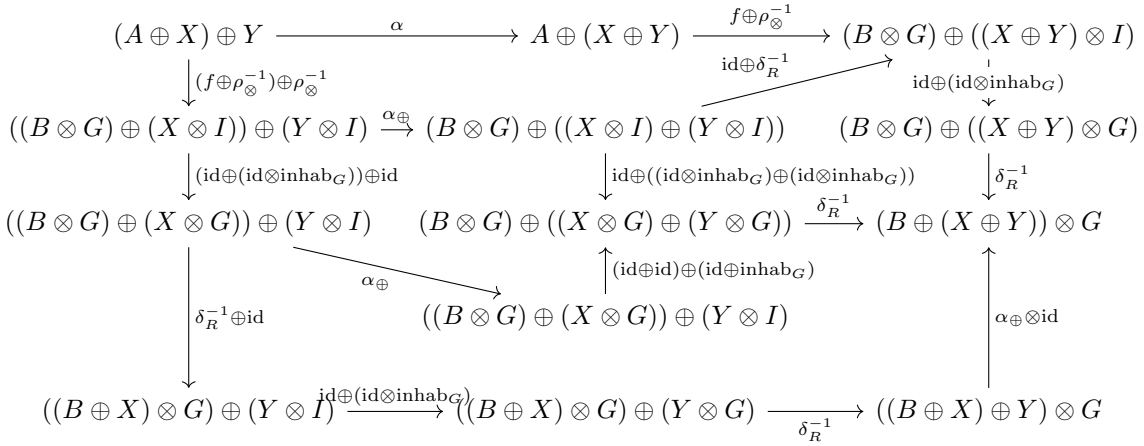


Here, the squiggly arrows are labelled with mediators that make the subdiagram commute up to equivalence.

Similarly, (**??**) means:



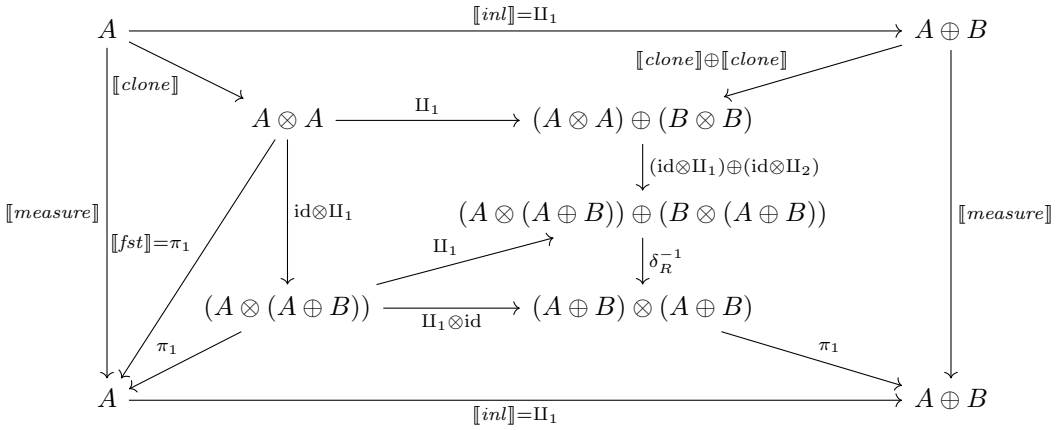
Finally (**??**):



The proof for $A \oplus -$ is analogous. □

Proposition 0. *Measurement commutes with injections:* $[\![measure_b \ggg inl]\!] = [\![inl \ggg measure_{b+b'}]\!]$, *and likewise for inr.*

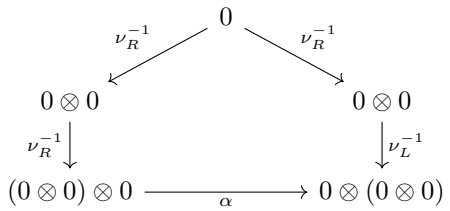
Proof. We verify commutativity of the diagram below.

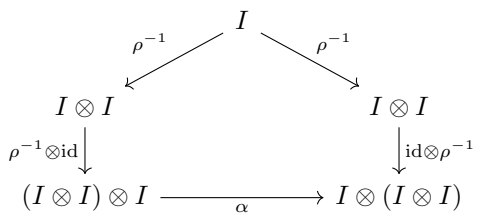


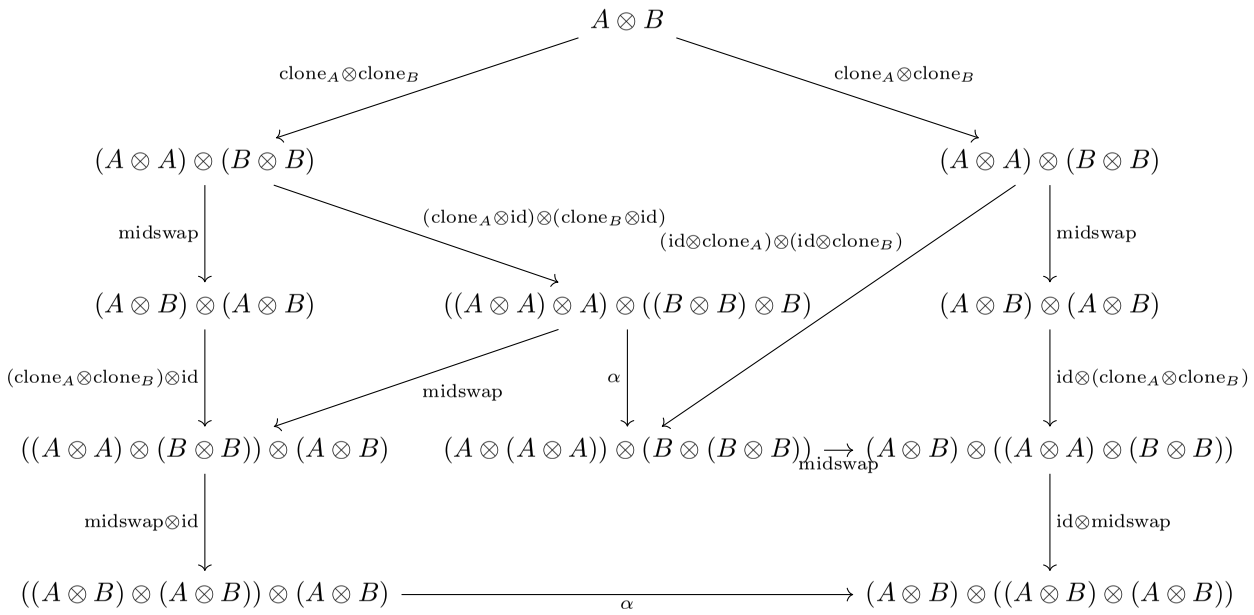
Here, the objects A and B range over interpretations of arbitrary $\mathcal{U}\Pi_a^\chi$ types b_1 and b_2, noting that $[\![b \times b']\!] = [\![b]\!] \otimes [\![b']\!]$ and $[\![b + b']\!] = [\![b]\!] \oplus [\![b']\!]$. □

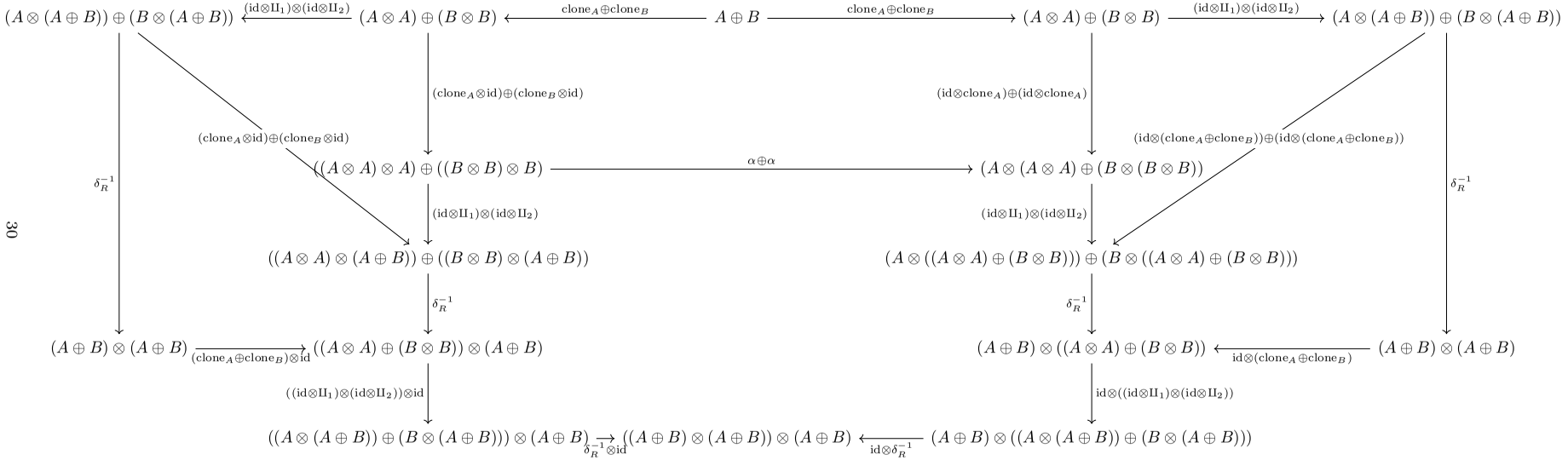
Proposition 1. *Cloning is associative:* $[\![clone \ggg (clone \times id) \ggg assoc^\times]\!] = [\![clone \ggg (id \times clone)]\!]$.

Proof. We verify commutativity of the following four diagrams, corresponding to the four instances of clone, depending on the type.






$A \otimes B$
$\mathrm{clone}_A \otimes \mathrm{clone}_B$
$\mathrm{clone}_A \otimes \mathrm{clone}_B$
$(A \otimes A) \otimes (B \otimes B)$
$(A \otimes A) \otimes (B \otimes B)$
midswap
$(\mathrm{clone}_A \otimes \mathrm{id}) \otimes (\mathrm{clone}_B \otimes \mathrm{id})$
$(\mathrm{id} \otimes \mathrm{clone}_A) \otimes (\mathrm{id} \otimes \mathrm{clone}_B)$
midswap
$(A \otimes B) \otimes (A \otimes B)$
$((A \otimes A) \otimes A) \otimes ((B \otimes B) \otimes B)$
$(A \otimes B) \otimes (A \otimes B)$
$(\mathrm{clone}_A \otimes \mathrm{clone}_B) \otimes \mathrm{id}$
midswap
α
$\mathrm{id} \otimes (\mathrm{clone}_A \otimes \mathrm{clone}_B)$
$((A \otimes A) \otimes (B \otimes B)) \otimes (A \otimes B)$
$(A \otimes (A \otimes A)) \otimes (B \otimes (B \otimes B))$
midswap
$(A \otimes B) \otimes ((A \otimes A) \otimes (B \otimes B))$
midswap⊗id
id⊗midswap
$((A \otimes B) \otimes (A \otimes B)) \otimes (A \otimes B)$
α
$(A \otimes B) \otimes ((A \otimes B) \otimes (A \otimes B))$


$(A \otimes (A \oplus B)) \oplus (B \otimes (A \oplus B))$
$(\mathrm{id}\otimes\Pi_1)\otimes(\mathrm{id}\otimes\Pi_2)$
$(A \otimes A) \oplus (B \otimes B)$
$\mathrm{clone}_A \oplus \mathrm{clone}_B$
$A \oplus B$
$\mathrm{clone}_A \oplus \mathrm{clone}_B$
$(A \otimes A) \oplus (B \otimes B)$
$(\mathrm{id}\otimes\Pi_1)\otimes(\mathrm{id}\otimes\Pi_2)$
$(A \otimes (A \oplus B)) \oplus (B \otimes (A \oplus B))$
$(\mathrm{clone}_A \otimes \mathrm{id}) \oplus (\mathrm{clone}_B \otimes \mathrm{id})$
$(\mathrm{id} \otimes \mathrm{clone}_A) \oplus (\mathrm{id} \otimes \mathrm{clone}_A)$
$(\mathrm{clone}_A \otimes \mathrm{id}) \oplus (\mathrm{clone}_B \otimes \mathrm{id})$
$(\mathrm{id}\otimes(\mathrm{clone}_A \oplus \mathrm{clone}_B)) \oplus (\mathrm{id}\otimes(\mathrm{clone}_A \oplus \mathrm{clone}_B))$
δ_R^{-1}
$((A \otimes A) \otimes A) \oplus ((B \otimes B) \otimes B)$
$\alpha \oplus \alpha$
$(A \otimes (A \otimes A) \oplus (B \otimes (B \otimes B))$
δ_R^{-1}
$(\mathrm{id}\otimes\Pi_1)\otimes(\mathrm{id}\otimes\Pi_2)$
$(\mathrm{id}\otimes\Pi_1)\otimes(\mathrm{id}\otimes\Pi_2)$
$((A \otimes A) \otimes (A \oplus B)) \oplus ((B \otimes B) \otimes (A \oplus B))$
$(A \otimes ((A \otimes A) \oplus (B \otimes B))) \oplus (B \otimes ((A \otimes A) \oplus (B \otimes B)))$
δ_R^{-1}
δ_R^{-1}
$(A \oplus B) \otimes (A \oplus B)$
$(\mathrm{clone}_A \oplus \mathrm{clone}_B) \otimes \mathrm{id}$
$((A \otimes A) \oplus (B \otimes B)) \otimes (A \oplus B)$
$(A \oplus B) \otimes ((A \otimes A) \oplus (B \otimes B))$
$\mathrm{id} \otimes (\mathrm{clone}_A \oplus \mathrm{clone}_B)$
$(A \oplus B) \otimes (A \oplus B)$
$((\mathrm{id}\otimes\Pi_1)\otimes(\mathrm{id}\otimes\Pi_2))\otimes\mathrm{id}$
$\mathrm{id}\otimes((\mathrm{id}\otimes\Pi_1)\otimes(\mathrm{id}\otimes\Pi_2))$
$((A \otimes (A \oplus B)) \oplus (B \otimes (A \oplus B))) \otimes (A \oplus B)$
$\delta_R^{-1} \otimes \mathrm{id}$
$((A \oplus B) \otimes (A \oplus B)) \otimes (A \oplus B)$
$\mathrm{id} \otimes \delta_R^{-1}$
$(A \oplus B) \otimes ((A \otimes (A \oplus B)) \oplus (B \otimes (A \oplus B)))$

□

University of Edinburgh
Email address: chris.heunen@ed.ac.uk

University of Edinburgh
Email address: robin.kaarsgaard@ed.ac.uk