

A Curry-Howard-Lambek type isomorphism for quantum computation with classical control

Philip Atzemoglou

January 26, 2009



Elevation; reproduced under permission by original artist.

1 Introduction

Quantum mechanics was initially developed in the first third of the previous century, with quantum computation being explicitly studied since the 1980's. Despite all of these efforts throughout the years, however, quantum computing is a nascent branch of science, with most of its computational paradigms not developed yet to higher levels of abstraction. While we do have a handful of quantum algorithms providing promising results in the area of computational efficiency, much of the work involved in designing such an algorithm seems to still be largely based on guesswork. There is no clear set of rules, or unifying principle, that would easily allow us to combine computational primitives into building a new and efficient quantum algorithm. Even though it is probably the case that quantum computation provides for an inherently more powerful means of computation than classical computing, that remains an unproven conjecture to this day. Unless we develop the means of abstracting our computational primitives to higher notions, setting the foundations for a solid theory of algorithms and figuring out what gives quantum computers their extra power, we will probably never be able to prove a computational speedup or write efficient algorithms en masse.

My plan is to use category theory to examine the categorical semantics of quantum computation and classical structures. This investigation will look into all three aspects of a Curry-Howard-Lambek type isomorphism for quantum computation with classical control, starting with the formulation of a categorical model and moving on to study both the logic and lambda calculus arising it.

The second and third sections of this report respectively cover the background in terms of category theory and quantum computing that is necessary for this report. The fourth section combines material from the previous two to define the model that will be used in my research. This model includes coloured picture diagrams that represent classical operations on interacting quantum observables. The fifth section gives a background in linear logic, along with an explanation of how it relates to categorical models. The sixth section gives a background in the linear lambda calculus and it discusses how it naturally arises from \dagger -compact categories. Finally, a seventh section looks at the work ahead for my proposed thesis, including various open questions popping up from the analysis of this report. This last part will discuss about

extending the linear logic and lambda calculus presented, in order to account for classical structures, via the translation of *spider monoids*. That will allow the application of the semantic techniques of this report in measurement based quantum computation. As a result, these constructions will then be able to be used to identify that element of quantum computation which gives it its purported computational benefits.

2 Category theory

Category theory is an area of mathematics that provides us with a means of reasoning about common properties of abstract structures; it allows the use of diagrammatic reasoning while, at the same time, extending connections to mathematical logic. This report will be making use of a lot of category theory [Mac98] so, for the sake of completeness, this section will provide definitions for of the notions used.

2.1 Basic notions

We begin by looking at categories, the most fundamental of the structures studied by category theory. Following that, we will be able to define functors and natural transformations, all of which will be used extensively later on.

Definition 2.1. A *category* is a collection of objects and arrows between objects such that the following conditions hold:

- There is a composition operator \circ , that can take any two arrows of the form $f : A \rightarrow B$ and $g : B \rightarrow C$ and produce a new arrow $g \circ f : A \rightarrow C$
- Composition is associative, so $h \circ (g \circ f) = (h \circ g) \circ f$
- For every object A in our category, there is an identity arrow $id_A : A \rightarrow A$
- The identity arrows satisfy the *unit law*, whereby, for any arrow $f : A \rightarrow B$ in our category, $id_B \circ f = f = f \circ id_A$

Definition 2.2. A *functor* is a morphism between categories, mapping objects to objects and arrows to arrows, in a way that preserves identities and composition.

Example. Consider two categories \mathcal{C} and \mathcal{D} . A functor $F : \mathcal{C} \rightarrow \mathcal{D}$, assigns to each object $A \in \mathcal{C}$ an object $FA \in \mathcal{D}$ and to each arrow $f : A \rightarrow B$ of \mathcal{C} an arrow $Ff : FA \rightarrow FB$ in \mathcal{D} . Since it preserves identities and composition, it will have to be the case that $F(id_A) = id_{FA}$ and $F(g \circ f) = Fg \circ Ff$.

Definition 2.3. When F and G are functors such that $F, G : \mathcal{C} \rightarrow \mathcal{D}$, a *natural transformation* $\eta : F \Rightarrow G$ is a collection of arrows $\eta_A : FA \rightarrow GA$ in \mathcal{D} for every object A in \mathcal{C} . These arrows have to be such that for any $f : A \rightarrow B$ in \mathcal{C} , the following diagram commutes:

$$\begin{array}{ccc}
 FA & \xrightarrow{\eta_A} & GA \\
 Ff \downarrow & & \downarrow Gf \\
 FB & \xrightarrow{\eta_B} & GB
 \end{array}$$

Note. When all the η_A of a natural transformation are isomorphisms, we call η a natural isomorphism.

2.2 More advanced notions

We now wish to define a way of describing how different categories are related to each other. In order to do that, consider categories \mathcal{C} and \mathcal{D} , with functors F and G between them such that:

$$\mathcal{C} \begin{array}{c} \xrightarrow{F} \\ \xleftarrow{G} \end{array} \mathcal{D}$$

The strictest and perhaps most obvious type of relation occurs when both of these functors are identity endofunctors $F = G = Id$, which would mean that the categories are *equal*. An *isomorphism* is a weaker kind of relation where $G \circ F = Id_{\mathcal{C}}$ and $F \circ G = Id_{\mathcal{D}}$. An even weaker kind of relation is an *equivalence*, where instead of requiring the composition of functors to be

equal to the identity functor, we ask that there be a natural isomorphism between them. In other words $G \circ F \cong Id_{\mathcal{C}}$ and $F \circ G \cong Id_{\mathcal{D}}$.

Continuing down that path leads us to a very important notion called an adjunction, which is another yet weaker kind of relation between categories [Che07]. To define this more rigorously:

Definition 2.4. Let \mathcal{C} and \mathcal{D} be categories, with functors $F : \mathcal{C} \rightarrow \mathcal{D}$ and $G : \mathcal{D} \rightarrow \mathcal{C}$. We say there is an *adjunction* $\langle F, G, \eta, \varepsilon \rangle$ when there exist two natural transformations $\eta : Id_{\mathcal{C}} \Rightarrow GF$ and $\varepsilon : FG \Rightarrow Id_{\mathcal{D}}$, respectively called the *unit* and *counit* of the adjunction, such that the following diagrams commute:

$$\begin{array}{ccc} G & \xrightarrow{\eta^G} & GFG \\ & \searrow 1_G & \downarrow G\varepsilon \\ & & G \end{array} \qquad \begin{array}{ccc} F & \xrightarrow{F\eta} & FGF \\ & \searrow 1_F & \downarrow \varepsilon^F \\ & & F \end{array}$$

Note. We say that F is *left adjoint* to G , writing this as $F \dashv G$. Similarly, G is *right adjoint* to F .

Definition 2.5. A *monad* $\langle T, \eta, \mu \rangle$ in a category \mathcal{C} consists of an endofunctor $T : \mathcal{C} \rightarrow \mathcal{C}$, together with two natural transformations $\eta : Id_{\mathcal{C}} \Rightarrow T$ and $\mu : T^2 \Rightarrow T$, such that the following diagrams commute:

$$\begin{array}{ccc} T^3 & \xrightarrow{T\mu} & T^2 \\ \mu T \downarrow & & \downarrow \mu \\ T^2 & \xrightarrow{\mu} & T \end{array} \qquad \begin{array}{ccccc} T & \xrightarrow{\eta T} & T^2 & \xleftarrow{T\eta} & T \\ & \searrow 1_T & \downarrow \mu & \swarrow 1_T & \\ & & T & & \end{array}$$

Corollary 2.6. Any adjunction $\langle F, G, \eta, \varepsilon \rangle$ defines a monad. This can be done by setting $T = GF$, which would make the unit of our adjunction a natural transformation of the form $\eta : Id_{\mathcal{C}} \Rightarrow T$ and our $\mu = G\varepsilon F : GF GF \Rightarrow GF$. The resulting monad would be of the form $\langle GF, \eta, G\varepsilon F \rangle$. This can all be easily verified by checking that the following diagrams do indeed commute:

$$\begin{array}{ccc} GFGFGF & \xrightarrow{GFG\varepsilon F} & GFGF \\ G\varepsilon FGF \downarrow & & \downarrow G\varepsilon F \\ GF & \xrightarrow{G\varepsilon F} & GF \end{array} \qquad \begin{array}{ccccc} GF & \xrightarrow{\eta GF} & GFGF & \xleftarrow{GF\eta} & GF \\ & \searrow 1_T & \downarrow G\varepsilon F & \swarrow 1_T & \\ & & GF & & \end{array}$$

2.3 Specific constructions

Definition 2.7. A *monoidal category* is a category that has been equipped with an bifunctor called tensor $\otimes : \mathcal{C} \times \mathcal{C} \longrightarrow \mathcal{C}$. Up to appropriate natural isomorphisms, the tensor is associative and features a special object I that acts as a left and right identity:

$$a_{A,B,C} : A \otimes (B \otimes C) \xrightarrow{\cong} (A \otimes B) \otimes C$$

$$l_A : I \otimes A \xrightarrow{\cong} A \qquad r_A : A \otimes I \xrightarrow{\cong} A$$

The tensor product thus forms a monoid, with I acting as the unit. These isomorphisms have to further satisfy some conditions called coherence conditions. These can be summarily represented by requiring that the following diagrams commute for all A, B, C and D :

$$\begin{array}{ccc}
 & (A \otimes B) \otimes (C \otimes D) & \\
 & \swarrow a & \searrow a \\
 A \otimes (B \otimes (C \otimes D)) & & ((A \otimes B) \otimes C) \otimes D \\
 \downarrow id \otimes a & & \uparrow pi \otimes v \\
 A \otimes ((B \otimes C) \otimes D) & \xrightarrow{a} & (A \otimes (B \otimes C)) \otimes D
 \end{array}$$

$$\begin{array}{ccc}
 l_I = r_I : I \otimes I \longrightarrow I & & A \otimes (I \otimes B) \xrightarrow{a} A \otimes (I \otimes B) \\
 & & \downarrow id \otimes l \\
 & & A \otimes B \xleftarrow{r \otimes id} A \otimes (I \otimes B)
 \end{array}$$

Definition 2.8. A *symmetric monoidal category* is a monoidal category with an additional natural isomorphism called symmetry, $\sigma_{A,B} : A \otimes B \xrightarrow{\cong} B \otimes A$,

such that the following diagrams commute:

$$\begin{array}{ccccc}
 A \otimes (B \otimes C) & \xrightarrow{id \otimes s} & A \otimes (C \otimes B) & \xrightarrow{a} & (A \otimes C) \otimes B \\
 \downarrow a & & & & \downarrow s \otimes id \\
 (A \otimes B) \otimes C & \xrightarrow{s} & C \otimes (A \otimes B) & \xrightarrow{a} & (C \otimes A) \otimes B
 \end{array}$$

$$\begin{array}{ccc}
 A \otimes B & \xrightarrow{s} & B \otimes A \\
 & \searrow id & \downarrow s \\
 & & A \otimes B
 \end{array}$$

$$\begin{array}{ccc}
 A \otimes I & \xrightarrow{s} & I \otimes A \\
 \downarrow r & & \swarrow l \\
 A & &
 \end{array}$$

Definition 2.9. A symmetric monoidal *closed* category is a symmetric monoidal category where, for any two objects A and B , there is an *exponential object*² $A \multimap B$, together with an evaluation morphism $ev_{A,B} : (A \multimap B) \otimes A \rightarrow B$. In addition to that, on any arrow of the form $f : C \otimes A \rightarrow B$, a process called *currying* yields a unique morphism $\Lambda(f) : C \rightarrow (A \multimap B)$ such that:

$$ev_{A,B} \circ (\Lambda(f) \otimes id_A) = f$$

$$\begin{array}{ccc}
 C \otimes A & & \\
 \downarrow \Lambda(f) \otimes id_A & \searrow f & \\
 (A \multimap B) \otimes A & \xrightarrow{ev_{A,B}} & B
 \end{array}$$

Definition 2.10. A *compact closed category* is a symmetric monoidal category where, for every object A , there is a dual object A^* along with two morphisms $d_A : I \rightarrow A^* \otimes A$ and $e_A : A \otimes A^* \rightarrow I$ such that:

²Note that these are different to the exponential connectives of linear logic.

$$(e_A \otimes id_A) \circ (id_A \otimes d_A) = id_A \quad \text{and} \quad (id_{A^*} \otimes e_A) \circ (d_A \otimes id_{A^*}) = id_{A^*}$$

The dual object is unique up to canonical isomorphism.

Corollary 2.11. *Every compact closed category is closed.*

Proof. All of the exponential structure can be recreated by setting $A \multimap B = A^* \otimes B$. The evaluation function can be simulated by:

$$ev_{A,B} = r_A \circ (id_B \otimes e_A) \circ (id_B \otimes s) \circ a^{-1} \circ (s \otimes id_A)$$

□

Definition 2.12. A \dagger -compact category is a compact closed category that is equipped with an involutive, contravariant, identity on objects endofunctor. That functor, called dagger, reverses all arrows, leaves objects unchanged and preserves the tensor structure. For any $f : A \rightarrow B$, it will be the case that $f^\dagger : B \rightarrow A$ and $f^{\dagger\dagger} = f$. Moreover, for any object A in our category, it must be the case that $\sigma_{A,A^*} \circ e_A^\dagger = d_A$

3 Quantum computing

Quantum computing is a radically different paradigm for computation which relies on the laws of quantum mechanics, in the hopes of achieving a higher computational efficiency than its currently used classical counterpart. This section will cover some fundamental concepts of quantum computing [Mer07, NC00], reviewing all the material that is necessary for understanding this report, without however providing a rigorous overview of the laws of quantum mechanics.

Classical computers operate on regular bits, whose value is either 0 or 1. When studying quantum computers, we view $|0\rangle$ and $|1\rangle$ as orthonormal vectors, using them as a basis to span a complex Hilbert space. We are free to pick a different set of orthonormal vectors as the basis of our Hilbert space, however, the one mentioned earlier is usually referred to as the *standard basis*. The length of a vector in this Hilbert space does not really matter, just its direction, so we group all vectors up to a complex multiple into equivalence

classes called rays. Qubits, the quantum analogue of a bit, can have any of these rays as their value. This means that the value or *state* of any qubit can be written as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where α and β are complex coefficients. The fact that a quantum state can be "a little bit of" $|0\rangle$ and "a little bit of" $|1\rangle$ at the same time, is called *superposition*. Transformations of a quantum system's state are described by unitary operations acting on the system's Hilbert space. State vectors can also be viewed as linear maps, in which case we simply write

$$|\psi\rangle : \mathbb{C} \rightarrow \mathcal{H} :: 1 \mapsto \psi$$

The state of composite quantum systems is represented by the tensor product of the Hilbert spaces that describe their constituent parts. This behaves like a regular Kronecker product; for a system composed of A and B , we write $A \otimes B$. Similarly, for two linear maps f and g running parallel to each other, each acting on a different state of a composite system, we would write $f \otimes g$. It seems natural that we could use $|\psi\rangle \otimes |\phi\rangle$ to describe the state of two qubits. We tend to write $|00\rangle$ for $|0\rangle \otimes |0\rangle$ and $|11\rangle$ for $|1\rangle \otimes |1\rangle$. Because of superposition, however, there are some cases where a state can not be written as the tensor product of two or more states. Typical examples of this are the bell states: $|00\rangle + |11\rangle$, $|00\rangle - |11\rangle$, $|01\rangle + |10\rangle$ and $|01\rangle - |10\rangle$.

Hilbert spaces require fixing a basis in their underlying vector space, so as a result, they come equipped with an inner-product:

$$\langle - | - \rangle : \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{C}$$

In order to formally introduce Dirac notation in our work, we want to further refine the definition of an inner-product by breaking it down to a composition of a *bra* $\langle\phi|$ and a *ket* $|\psi\rangle$, yielding $\langle\phi|\psi\rangle = \langle\phi| \circ |\psi\rangle$. This requires that we introduce the notion of an *adjoint*:

$$(f : \mathcal{H}_1 \rightarrow \mathcal{H}_2) \mapsto (f^\dagger : \mathcal{H}_2 \rightarrow \mathcal{H}_1)$$

where

$$\langle\psi| := (|\psi\rangle)^\dagger : \mathcal{H} \rightarrow \mathbb{C}$$

Performing a measurement on a quantum system against some orthonormal basis, destroys its state, by making it collapse into one of the basis vectors. To define this more formally, a measurement against some orthonormal basis consists of a set of projectors P_i , each sending the measured state to one of the basis vectors. At the time of measurement, a probabilistic decision is made as to which projector will be applied to the state vector. If we represent the state as a linear combination of basis vectors, the square of the complex coefficient of any basis vector gives us the probability of collapsing to that outcome during a measurement.

4 Categorical model

This section goes over the categories used to model quantum computation and classical operations in the rest of the report. It starts by explaining how the Hilbert space formalism can be recast into the language of \dagger -compact categories [AC04]. One of the biggest practical advantages of monoidal categories is that, on many occasions, they "formally justify their absence" [Coe06] in that they can be represented using a graphical calculus that greatly simplifies categorical reasoning. This section also demonstrates how every element of the initial quantum structure is represented graphically in what resembles a two dimensional Dirac notation. It should be noted at this point that all the diagrams should be read from bottom to top, as that will be the direction of the compositional flow of time. The next part of this section deals with classical operations, which are modelled in terms of internal spider monoids, as well as with the computational interplay inherent in introducing complementarity.

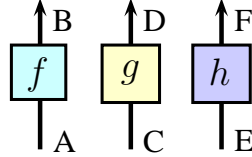
4.1 Categorical quantum computation

The category we will be using to model quantum computation is called *FD-Hilb* and is the category of finite dimensional complex Hilbert spaces. Its objects are finite dimensional Hilbert spaces and its arrows are linear maps. Monoidal multiplication is represented by the Kronecker tensor product, while the monoidal unit object corresponds to the set of complex numbers

$I = \mathbb{C}$. Associativity of the tensor and tensor identities are up to equality, so $a_{A,B,C}$, l_A and r_A are reduced to identity arrows.

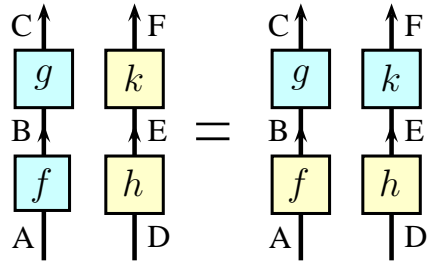
The adjoint is modelled using the dagger functor. In terms of the picture calculus, the dagger denotes flipping a picture upside down, while the arrows continue pointing the same way they were before (i.e. upwards).

For any three arrows $f : A \rightarrow B$, $g : C \rightarrow D$ and $h : E \rightarrow F$, associativity allows us to write their tensor product as:



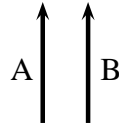
By bifactoriality of the tensor, we know that it preserves composition. For any arrows of the form $f : A \rightarrow B$, $g : B \rightarrow C$, $h : D \rightarrow E$ and $k : E \rightarrow F$, once we add composition to our diagrams, the following property should become more evident:

$$(g \circ f) \otimes (k \circ h) = (g \otimes k) \circ (f \otimes h)$$



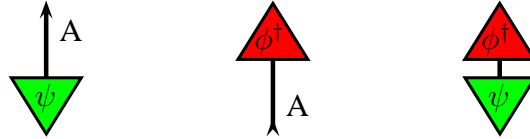
as well as that it preserves identities:

$$id_{A \otimes B} = id_A \otimes id_B$$



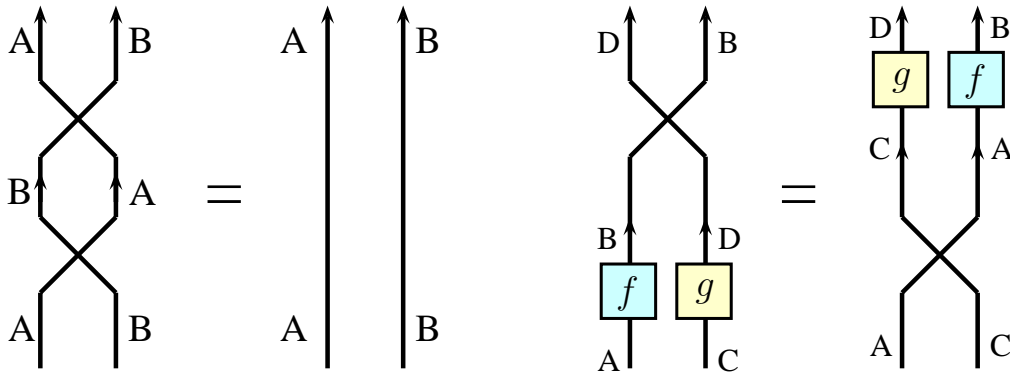
The proper graphical representation for the unit I is "no line", while arrows containing I as their domain or codomain are represented as follows:

$$\psi : I \rightarrow A \qquad \phi^\dagger : A \rightarrow I \qquad \phi^\dagger \circ \psi : I \rightarrow I$$



A special case of arrows called scalars consists of all arrows of the form $c : I \rightarrow I$. In these specific cases, our categorical structure collapses to the point where tensor is equal to composition. In other words $c_1 \otimes c_2 = c_1 \circ c_2 = c_2 \otimes c_1$. Scalars can be moved freely around in the category's graphical representation.

Symmetry corresponds to a well known quantum operation called *swap*; it is graphically represented by a pair of crossing lines. The following properties are a lot easier to understand graphically:



$$\sigma_{A,B} \circ \sigma_{A,B} = id_A \otimes id_B$$

$$\sigma_{A,B} \circ (f \otimes g) = (g \otimes f) \circ \sigma_{A,B}$$

Compact closure is used to model entangled states. These are the only cases we see arrows pointing downwards, as the $*$ in A^* reverses the arrow's

direction. The graphical representation looks like this:

$$d_A : I \longrightarrow A^* \otimes A \qquad e_A : A \otimes A^* \longrightarrow I$$

These have to adhere to a property, fundamental in proving teleportation, whose graphical representation is reminiscent of *yanking* a wire:

4.2 Representing classical structures

One of the fundamental known distinctions between quantum and classical computation is derived from no-go theorems. Classical computers routinely copy and delete data; it is such a commonplace thing to do that we hardly ever notice how entwined it is to the classical computational paradigm itself. Quantum computers, on the other hand, can not perform either of these operations. In this part of this section, we will see how to turn this problem into a very important feature, which will in turn enable us to account for classical operations within the, already defined, quantum categorical framework. The following result [WZ82] is referred to as the *no-cloning* theorem:

Theorem 4.1. *There is no quantum operation D , such that*

$$D : |\psi\rangle \mapsto |\psi\rangle \otimes |\psi\rangle$$

$$D : |\phi\rangle \mapsto |\phi\rangle \otimes |\phi\rangle$$

unless $|\psi\rangle$ and $|\phi\rangle$ are orthogonal.

Similarly, there is another result [PB00], commonly referred to as the *no-deleting* theorem:

Theorem 4.2. *There is no quantum operation E , such that*

$$E : |\psi\rangle \mapsto 1$$

$$E : |\phi\rangle \mapsto 1$$

unless $|\psi\rangle$ and $|\phi\rangle$ are orthogonal.

From these two theorems, we can deduce that the only cases of quantum states that we could treat as classical are those pertaining to orthogonal vectors. This matches our initial intuition that quantum computation, as a complex Hilbert space, is spanned by a classical basis.

Definition 4.3. A *classical structure*³ [CP07, CD08, CPP08b] is defined in terms of special \dagger -Frobenius cocommutative comonoids, henceforth referred to as spider monoids. They are represented by a triplet (A, δ, ϵ) , where A is an object and δ and ϵ are two morphisms of the form:

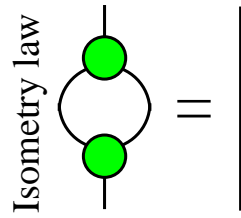
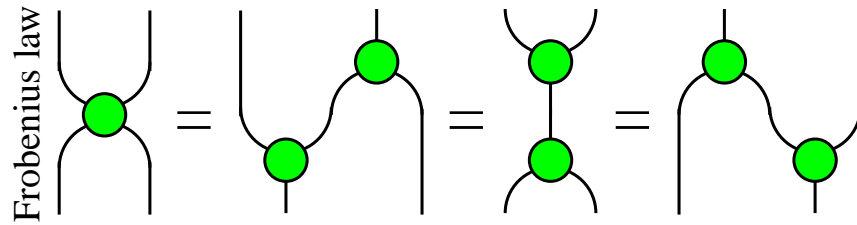
$$\delta : A \longrightarrow A \otimes A :: a_i \mapsto a_i \otimes a_i \qquad \epsilon : A \longrightarrow I :: a_i \mapsto 1$$



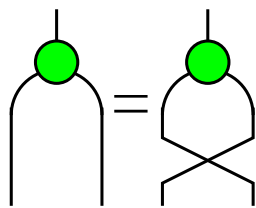
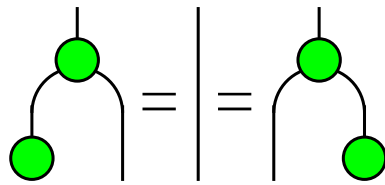
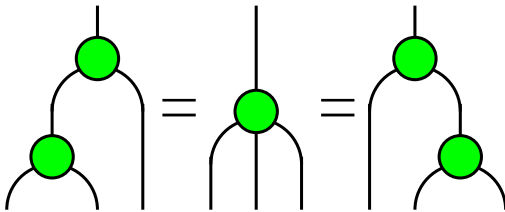
where the a_i are the orthogonal vectors to which copying and deleting take place. There are two ways, that can be used interchangeably, of defining the properties that these morphisms will need to satisfy. The most common definition is that they need to satisfy monoid, comonoid, isometry and Frobenius

³N.B. Thanks to [CPV08], we now know that classical structures are in bijective correspondence to bases.

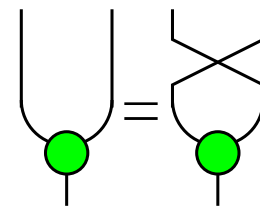
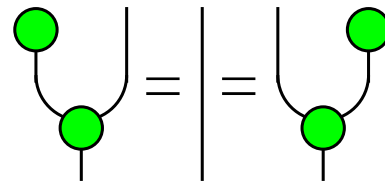
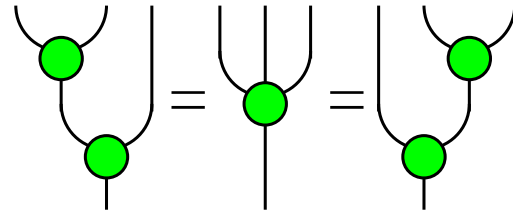
laws or graphically:



Monoid laws

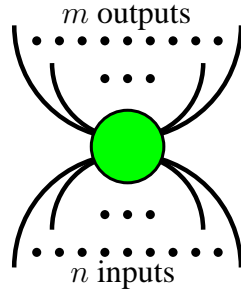


Comonoid laws

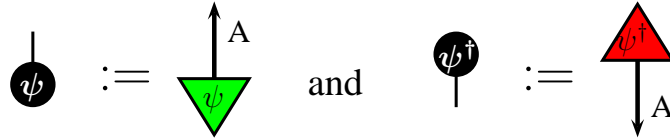


The graphical representation for all of these laws corresponds to connected graphs. Furthermore, the inputs and outputs on all of the required equations match, so, as [CP06, CPP08b, CD08] proved, we can equivalently define

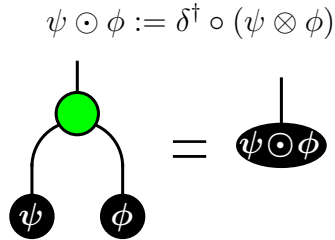
classical structures using the *spider theorem*. This theorem states that if a graph generated by δ and ϵ is connected, then it is completely characterized by its domain and codomain. If the domain is $\underbrace{A \otimes \dots \otimes A}_n$ and the codomain is $\underbrace{A \otimes \dots \otimes A}_m$, then it can be reduced to a "spider" with n input and m output wires.



Arbitrary states that are points of A (i.e. of the form $|\psi\rangle : I \longrightarrow A$) are denoted by "black dots":

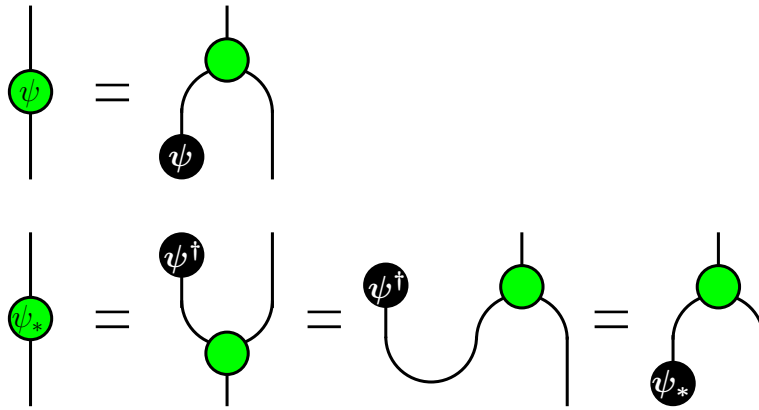


The monoid operation $\delta^\dagger : A \otimes A \longrightarrow A$, also known as *fusion*, can be used to combine arbitrary states. For two states $|\psi\rangle$ and $|\phi\rangle$, their fusion is written as:



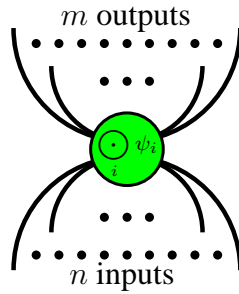
On any classical structure (A, δ, ϵ) , we define a map Λ that lifts any state $|\psi\rangle$ of A to the endomorphism $\Lambda(\psi) := \delta^\dagger \circ (\psi \otimes 1_A) : A \longrightarrow A$. Similarly,

we can lift any bra $\langle \psi |$ by setting $\Lambda(\psi)^\dagger = \Lambda(\psi_*) = (\psi^\dagger \otimes 1_A) \circ \delta : A \longrightarrow A$. We denote these graphically as:



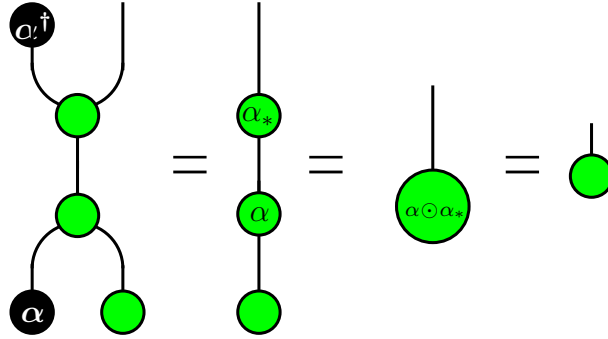
After combining all of these definitions, we can prove [CD08] that classical structures follow what is known as the generalized spider theorem:

Theorem 4.4. *Any connected graph generated by the operations of the classical structure (A, δ, ϵ) , states $|\psi_i\rangle : I \longrightarrow A$ and the \dagger -compact structure, is completely characterized by its domain, codomain and $\Lambda(\bigodot_i \psi_i)$. The graphical representation is that of a "decorated spider"*

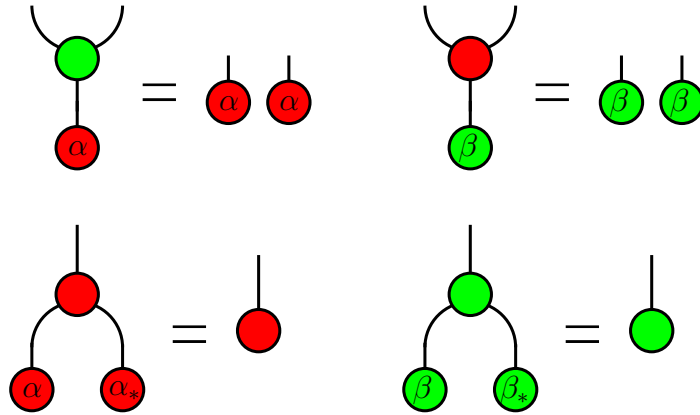


Definition 4.5. A point $\alpha : I \longrightarrow A$ is *unbiased* relative to (A, δ, ϵ) iff $\Lambda(\alpha)$ is unitary. In other words, there needs to be a scalar $s : I \longrightarrow I$ such that

$s \cdot \alpha \odot \alpha^\dagger = \epsilon^\dagger$, or graphically:



Definition 4.6. Two classical structures $(A, \delta_G, \epsilon_G)$ and $(A, \delta_R, \epsilon_R)$ in a \dagger -compact category are called *complementary* if the points that are classical for one are unbiased for the other and vice versa. δ_G and ϵ_G are depicted using green dots; the points they copy and delete are drawn in red and are unbiased for the second classical structure. δ_R and ϵ_R are depicted using red dots; they copy and delete green points, which are unbiased for the green classical structure. Graphically this thing becomes:



5 Linear logic

This section provides an incremental overview of the structures found in some flavours of linear logic, leading up to the corresponding logic for \dagger -compact

categories, while at the same time illustrating how each of these concepts relates to category theoretic notions and properties.

Linear logic is a resource sensitive logic, first introduced in [Gir87]. Whereas other logics provide the structural rules of weakening and contraction in order to facilitate predicate re-use or non-use in proving truths, linear logic drops the indiscriminate use of these rules and treats predicates as resources that need to be expended in order to produce proofs. In terms of the Gentzen sequent calculus, the rules of weakening and contraction would be represented as:

$$\frac{\Gamma \vdash B}{\Gamma, A \vdash B} \text{ (Weakening)} \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B} \text{ (Contraction)}$$

Due to its resource sensitivity, linear logic finds many applications in computer science such as in type theory, the semantics of programming languages and the study of concurrency. A computational interpretation was given in [Abr93], which was later extended in [AD05] to better express the unique features of quantum computation. The definitions and presentation of this section are largely based on the approach taken by these last two papers; note, however, that some extra rules have been added to this logic. A categorical interpretation will be given for each of the rules presented, in order to facilitate the Curry-Howard parallelisms of this report. The proof rules for our logic are as follows:

	Logic	Categories
Id	$\overline{A \vdash A}$	$\overline{id_A : A \longrightarrow A}$
$\otimes R$	$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}$	$\frac{f : \Gamma \longrightarrow A \quad g : \Delta \longrightarrow B}{f \otimes g : \Gamma \otimes \Delta \longrightarrow A \otimes B}$
$\otimes L$	$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}$	$\frac{f : (\Gamma \otimes A) \otimes B \longrightarrow C}{f \circ a_{A,B,\Gamma} : \Gamma \otimes (A \otimes B) \longrightarrow C}$
Cut	$\frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}$	$\frac{f : \Gamma \longrightarrow A \quad g : A \otimes \Delta \longrightarrow B}{g \circ (f \otimes id_\Delta) : \Gamma \otimes \Delta \longrightarrow B}$
$\multimap E$	$\frac{\Gamma \vdash A \multimap B \quad \Delta \vdash A}{\Gamma, \Delta \vdash B}$	$\frac{f : \Gamma \longrightarrow (A \multimap B) \quad g : \Delta \longrightarrow A}{ev_{A,B} \circ (f \otimes g) : \Gamma \otimes \Delta \longrightarrow B}$
$\multimap R$	$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}$	$\frac{f : \Gamma \otimes A \longrightarrow B}{\Lambda(f) : \Gamma \longrightarrow (A \multimap B)}$

At this point, it is interesting to compare the two worlds and see how some notions translate from one to the other. The identity rule corresponds to identity arrows in our categories. The cut rule defines function composition. The right tensor rule ($\otimes R$) defines tensoring, while the left tensor rule ($\otimes L$) defines the associativity of the tensor. Linear implication (\multimap) is a notion equivalent to a category's exponential objects, so naturally, implication elimination ($\multimap E$) and the right implication rule ($\multimap R$) respectively define the category's evaluation and currying functions. Now we'll add a structural rule known as *exchange*:

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C} \quad \frac{f : \Gamma \otimes A \otimes B \otimes \Delta \longrightarrow C}{f \circ (id_\Gamma \otimes s_{A,B} \otimes id_\Delta) : \Gamma \otimes B \otimes A \otimes \Delta \longrightarrow C}$$

This rule corresponds to the symmetry isomorphism for the tensor. All we need in order to have a fully fledged representation of symmetric monoidal

closed categories is a monoidal unit. That is denoted by 1 and comes with the simple rule of $\overline{\vdash 1}$. The natural question to ask at this point is whether we can represent compact closure, which as it turns out has already been done in [AD05]. The way to do this is by explicitly including linear negation in our notation, which for an object A would give us A^\perp . Linear negation in our case is characterized by the following laws:

$$\begin{aligned} A^{\perp\perp} &= A \\ 1^\perp &= 1 \\ (A \otimes B)^\perp &= A^\perp \otimes B^\perp \\ A \multimap B &= A^\perp \otimes B \end{aligned}$$

In "conventional" flavours of linear logic, the De Morgan dual of the tensor is defined as a multiplicative operation called par. In our case, however, we have just set the tensor to be equal to par; so conjunction is equal to disjunction. Surprisingly, this is the most natural way to include compactness in our logic. The dual object A^* of every object A in our category, is represented by A^\perp in our logic. We can then use the right implication rule to transform our identity rule to $\vdash A \multimap A$ which, when translated via the linear negation laws, becomes $\vdash A^\perp \otimes A$, or the equivalent of $d_A : I \longrightarrow A^* \otimes A$ in our category. Thus, we can now represent compact structure in this flavour of linear logic.

We will turn our focus to \dagger -compact categories. The additional element needed for faithfully representing a quantum structure is the dagger functor. All that we need in order to represent the dagger is an extra rule called the *dagger flip*:

$$\frac{A_1, A_2, \dots, A_k \vdash B_1 \otimes B_2 \otimes \dots \otimes B_\ell}{B_1, B_2, \dots, B_\ell \vdash A_1 \otimes A_2 \otimes \dots \otimes A_k} (\dagger\text{-flip}) \qquad \frac{f : A \longrightarrow B}{f^\dagger : B \longrightarrow A}$$

6 Lambda calculus

The lambda calculus is a formal system studying function definition, application and recursion. It started as a tool for studying the theory of functions, but quickly found application in areas such as logic, proof theory, computability and even formed the basis for what we now know as functional programming. The typed lambda calculus was introduced by placing more structures on the formal system's construction rules, as a way of prohibiting cases with an infinite reduction path. The lambda calculus corresponding to \dagger -compact categories and linear quantum logics, henceforth referred to as a \dagger -*lambda calculus*, is a radical departure from conventional lambda calculi. Its roots lie in the *typed linear lambda calculus* but, instead of restricting lambda abstraction to only binding variables, the \dagger -*lambda calculus* also allows the binding of composite terms. As a consequence of this, composite terms can now also appear to the left of a turnstile; a fact which calls for an expanded set of β -reduction rules. This section will go over the basic definitions and typing rules necessary for such a \dagger -*lambda calculus* while, once again, explaining how these correspond to category theoretic notions. Please note that the definitions presented in this section, especially those of the reduction and reconstruction rules, are work currently in progress and are presented in this report merely for the sake of giving the reader an idea of where this research is heading.

Variables are denoted by single letters and are traditionally represented using the later letters the alphabet. They are of the form:

$$x, y, z$$

Terms are denoted by different combinations of the following forms:

$$a, b ::= x \mid a \otimes b \mid \text{let } \psi \text{ be } a \otimes b \text{ in } c \mid \lambda a. b \mid ab$$

Types can appear as any combination of the following forms:

$$A, B ::= A \mid A \otimes B \mid A \multimap B$$

A few things need to be noted here. First of all, we extend the notion of a

free variable to:

$$\begin{aligned}FV(x) &= \{x\} \\FV(ab) &= FV(a) \cup FV(b) \\FV(a \otimes b) &= FV(a) \cup FV(b) \\FV(\lambda a.b) &= FV(b) - FV(a) - \{a\}\end{aligned}$$

Finally, the typing judgements, or statements, we will be making are composed of terms and types. They are always of the form:

$$a_1 : A_1, a_2 : A_2, \dots, a_k : A_k \vdash b : B$$

A set of typing rules is used to produce typing judgements; these correspond to the rules that were presented in the section on linear logic, as well as to structural notions found in \dagger -compact categories. The rules and their respective correspondences are as follows:

	Lambda Calculus	Categories
Id	$\frac{}{a : A \vdash a : A}$	$\frac{}{id_A : A \longrightarrow A}$
$\otimes R$	$\frac{\Gamma \vdash a : A \quad \Delta \vdash b : B}{\Gamma, \Delta \vdash a \otimes b : A \otimes B}$	$\frac{f : \Gamma \longrightarrow A \quad g : \Delta \longrightarrow B}{f \otimes g : \Gamma \otimes \Delta \longrightarrow A \otimes B}$
$\otimes L$	$\frac{\Gamma, a : A, b : B \vdash c : C}{\Gamma, \psi : A \otimes B \vdash \text{let } \psi \text{ be } a \otimes b \text{ in } c : C}$	$\frac{f : (\Gamma \otimes A) \otimes B \longrightarrow C}{f \circ a_{A,B,\Gamma} : \Gamma \otimes (A \otimes B) \longrightarrow C}$
Cut	$\frac{\Gamma \vdash c : A \quad a : A, \Delta \vdash b : B}{\Gamma, \Delta \vdash b[c/a] : B}$	$\frac{f : \Gamma \longrightarrow A \quad g : A \otimes \Delta \longrightarrow B}{g \circ (f \otimes id_\Delta) : \Gamma \otimes \Delta \longrightarrow B}$
$\multimap E$	$\frac{\Gamma \vdash g : A \multimap B \quad \Delta \vdash a : A}{\Gamma, \Delta \vdash ga : B}$	$\frac{f : \Gamma \longrightarrow (A \multimap B) \quad g : \Delta \longrightarrow A}{ev_{A,B} \circ (f \otimes g) : \Gamma \otimes \Delta \longrightarrow B}$
$\multimap R$	$\frac{\Gamma, a : A \vdash b : B}{\Gamma \vdash \lambda a. b : A \multimap B}$	$\frac{f : \Gamma \otimes A \longrightarrow B}{\Lambda(f) : \Gamma \longrightarrow (A \multimap B)}$

The Γ 's and Δ 's that appear in some of the rules are shorthand for sets of typed terms, each element coming with its type as usual. These sets are otherwise known as contexts. The free terms of a context Γ , denoted by $FV(\Gamma)$, are given by the union of the free terms in each of the context's terms.

Applying these rules a number of times can lead us to composite but reducible forms. These are reduced using a process called β -reduction as follows:

$$\begin{aligned}
(\lambda x. b)a &\xrightarrow{\beta} b[a/x] \\
\text{let } a \otimes b \text{ be } x \otimes y \text{ in } c &\xrightarrow{\beta} c[a/x, b/y]
\end{aligned}$$

The notation $b[c/a]$ is referred to as substitution and means "take b and replace all free occurrences of a in it with c ". In defining an operational

semantics for substitution, care has to be taken to prevent us from violating boundedness of terms. A more rigorous way of defining an substitution is given by a double induction⁴ on the structures of a and b as follows:

Case " $a \equiv x$ (is a variable)":

$$\begin{aligned} y [c/x] &:= \begin{cases} c & \text{for } x = y, \\ y & \text{for } x \neq y \end{cases} \\ (pq)[c/x] &:= (p[c/x])(q[c/x]) \\ (\lambda p.q)[c/x] &:= \lambda p.(q[c/x]) \end{aligned}$$

Case " $a \equiv mn$ (is an application)":

$$\begin{aligned} y [c/mn] &:= \begin{cases} y & \text{for } b = y \notin FV(mn), \\ (\lambda(mn).y)c & \text{for } b = y \in FV(mn) \end{cases} \\ (pq)[c/mn] &:= \begin{cases} c & \text{for } b = pq = mn, \\ (p[c/mn])(q[c/mn]) & \text{for } b = pq \neq mn \end{cases} \\ (\lambda p.q)[c/mn] &:= \lambda p.(q[c/mn]) \end{aligned}$$

Case " $a \equiv \lambda m.n$ (is an abstraction)":

$$\begin{aligned} y [c/\lambda m.n] &:= \begin{cases} y & \text{for } b = y \notin FV(\lambda m.n), \\ (\lambda(\lambda m.n).y)c & \text{for } b = y \in FV(\lambda m.n) \end{cases} \\ (pq)[c/\lambda m.n] &:= (p[c/\lambda m.n])(q[c/\lambda m.n]) \\ (\lambda p.q)[c/\lambda m.n] &:= \begin{cases} c & \text{for } \lambda m.n =_{\alpha} \lambda p.q, \\ \lambda p.(q[c/\lambda m.n]) & \text{for } \lambda m.n \neq_{\alpha} \lambda p.q. \end{cases} \end{aligned}$$

In order to preserve boundedness, the new rules sometimes⁵ do not β -

⁴In the cases where b is a λ -abstraction, certain provisions have to be in place to prevent us from violating boundedness. These provisions are: $FV(a) \cap FV(p) = \emptyset$, $FV(c) \cap FV(p) = \emptyset$, and $a \neq_{\alpha} p \neq_{\alpha} c$.

⁵When $b = y$ is a variable, $a = mn$ is an application and $y \in FV(mn)$, the term isn't β -reduced at all. Similarly, the term isn't β -reduced when $b = y$ is a variable, $a = \lambda m.n$ is an abstraction and $y \in FV(\lambda m.n)$.

reduce a term when acting on variables, instead maintaining these terms in a dormant state. These dormant states are reactivated through the use of a reconstruction rule called β^\dagger -reconstruction. The rule is defined as follows:

$$\begin{aligned}
((\lambda(mn).p)c)((\lambda(mn).q)c) &\xrightarrow{\beta^\dagger} (\lambda(mn).pq)c && \text{when } p, q \in FV(mn) \\
\lambda p.(\lambda(\lambda m.n).q)c &\xrightarrow{\beta^\dagger} (\lambda(\lambda m.n).(\lambda p.q))c && \text{when } q \in FV(\lambda m.n) \\
&&& FV(\lambda m.n) \cap FV(p) = \emptyset \\
&&& FV(c) \cap FV(p) = \emptyset \\
&&& \lambda m.n \neq_\alpha p \neq_\alpha c
\end{aligned}$$

We now need to go over the rationale behind the definitions for β -reduction. The case where a is a variable makes everything collapse to the usual way of defining β -reduction, so there isn't anything special about it to comment. In the case where $a = mn$ and b is the variable y , $y[c/mn]$ can be thought of as "take y and replace all occurrences of mn in it with c ". Since y is a variable, it will not have any occurrences of mn in it to replace. An obvious, yet wrong, intuition in this case is to set $y[c/mn] = y$. If $y \in FV(mn)$, then there's a chance we'll get to use that substitution in the future:

$$\begin{aligned}
((\lambda xy.yx)nm)[c/mn] &= ((\lambda xy.yx)[c/mn])((nm)[c/mn]) = \\
(\lambda xy.yx)((\lambda(mn).n)c)((\lambda(mn).m)c) &= ((\lambda(mn).m)c)((\lambda(mn).n)c)
\end{aligned}$$

This last form is a typical example of where we can use β^\dagger -reconstruction:

$$((\lambda(mn).m)c)((\lambda(mn).n)c) \xrightarrow{\beta^\dagger} (\lambda(mn).mn)c \xrightarrow{\beta} (mn)[c/mn] = c$$

Similarly, in the case where $a = \lambda m.n$ and b is the variable y , it might be tempting to set $y[c/\lambda m.n] = y$ since y can not possibly contain any occurrences of $\lambda m.n$. However, if $y \in FV(\lambda m.n)$, then perhaps we'll use the substitution in the future:

$$\begin{aligned}
((\lambda t.(\lambda u.t))n)[c/\lambda m.n] &= (\lambda t.(\lambda u.t))[c/\lambda m.n] \quad n[c/\lambda m.n] = \\
(\lambda t.(\lambda u.t)[c/\lambda m.n]) \quad (\lambda(\lambda m.n).n)c &= (\lambda t.(\lambda u.t)) \quad (\lambda(\lambda m.n).n)c = \\
&= \lambda u.(\lambda(\lambda m.n).n)c
\end{aligned}$$

Where, again, β^\dagger -reconstruction gives us:

$$\lambda u.(\lambda(\lambda m.n).n)c \xrightarrow{\beta^\dagger} (\lambda(\lambda m.n).(\lambda u.n))c \xrightarrow{\beta} (\lambda u.n)[c/\lambda m.n] = c$$

The rest of the cases in the definition of substitution come naturally from the usual way of defining β -reduction, only differentiating in cases where $(mn)[c/mn] = c$ and $(\lambda p.q)[c/\lambda m.n] = c$ (when $\lambda p.q =_\alpha \lambda m.n$) which are obvious cases where a substitution is called for. In cases where the bound term is a lambda abstraction, the special conditions of $FV(a) \cap FV(p) = \emptyset$, $FV(c) \cap FV(p) = \emptyset$, and $a \neq_\alpha p \neq_\alpha c$ are specially crafted to give us $FV((\lambda a.b)c) = FV(b[c/a])$; a capture avoiding substitution.

Now, if $t_0 \xrightarrow{\beta} t_1 \xrightarrow{\beta^\dagger} t_2 \xrightarrow{\beta} \dots$, we say $t_0 \xrightarrow{\beta^\dagger \circ \beta} t_2$ and $t \xrightarrow{\beta^\dagger \circ \beta} t'$. Though it would be interesting to look at the properties of these reductions, since the contents of this section are still work in progress, the part on reductions would be way beyond the scope of this report.

A number of interesting translations take place between category theoretic and lambda calculus notions. One of them occurs in $\otimes L$, the rule that defines tensor associativity in our category, whereby we gain a rule on how to type a term when, instead of using two other terms for its derivation, we plug in a composite term in their place. Another interesting case is cut, the rule that defines composition of functions, which in the lambda calculus corresponds to a type rule for substitution. Implication elimination ($\multimap E$), the one responsible for the evaluation function of our category, turns out to be function application. Consequently, the related notion of currying that comes with the right implication rule ($\multimap R$), corresponds to lambda abstraction.

Similarly to the way we defined rules in the section on Logic, we need to add the structural rule of exchange in order to account for the symmetry isomorphism of the tensor:

$$\frac{\Gamma, a : A, b : B, \Delta \vdash c : C}{\Gamma, b : B, a : A, \Delta \vdash c : C} \quad \frac{f : \Gamma \otimes A \otimes B \otimes \Delta \longrightarrow C}{f \circ (id_\Gamma \otimes s_{A,B} \otimes id_\Delta) : \Gamma \otimes B \otimes A \otimes \Delta \longrightarrow C}$$

Compact closure is just as simple to demonstrate as it was in the case of logic. All we did in the previous section was start with the identity law and then apply the right implication law to curry it. Similarly, in the context of

the lambda calculus, we can start with the identity typing rule $a : A \vdash a : A$ and apply lambda abstraction to it to get the identity function: $\vdash \lambda a.a : A \multimap A$. The lambda calculus allows us to see the compact structure for what it is, almost as well as the picture diagrams do in category theory; it is a facilitator for the flow of information, a function that inputs $a : A$ and outputs $a : A$.

In order to model the dagger functor in the lambda calculus, the \dagger -flip rule in itself is not sufficient. The two additional rules that we will need, \dagger -R and \dagger -L, are in fact one bidirectional rule. They are defined in the table below:

	Lambda Calculus	Categories
\dagger -flip	$\frac{a_1 : A_1, \dots, a_k : A_k \vdash b_1 : B_1 \otimes \dots \otimes b_\ell : B_\ell}{b_1 : B_1, \dots, b_\ell : B_\ell \vdash a_1 : A_1 \otimes \dots \otimes a_k : A_k}$	$\frac{f : \otimes A_i \longrightarrow \otimes B_j}{f^\dagger : \otimes B_j \longrightarrow \otimes A_i}$
\dagger -R and \dagger -L	$\frac{a[p/q] : A \vdash b : B}{a : A \vdash b[q/p] : B}$	$\frac{(g \circ f)^\dagger : A \longrightarrow B}{f^\dagger \circ g^\dagger : A \longrightarrow B}$

7 Further work

This final section will include a complete plan of my proposed work, the methodology that I am going to use, as well as some preliminary milestones. It will begin by discussing some open questions that arise from the analysis in this report. I will then outline the immediate next steps in the research plan I have prepared for my dissertation, the approach that I will take, as well as the way in which these steps will help me address those aforementioned open problems. I will then continue by presenting the main motivation behind this direction of research and by explaining why these structures and semantics are ideally suited as a model for *measurement based quantum computation*. My plan is to use *spider monoids* to study these measurement based systems, in particular the resulting concurrency of operations, and then use that analysis in order to draw some results on computational complexity.

Some attempts at defining a programming language for quantum computation have already been made [Sel04b, vT04]. As these attempts progressed [vTD07], it became clear that categorical semantics can provide a sound denotational semantics for high-order quantum programming languages. The methodological approach used in this report, as well as that of my proposed thesis, is one where the language’s description arises naturally from the associated categorical semantics. This allows for a language whose formulation is not syntactically arbitrary, provides insight into the structures that govern quantum computation and, finally, extends to higher-order functions in a formalised way. Selinger’s work on quantum programming languages [Sel04a] emphasises the importance of incorporating classical control structures in a quantum programming language. This was later done in [SV06], in the form of a quantum lambda calculus with classical control, featuring an operational semantics but no denotational semantics. The first milestone in my research plan, is a complete translation of the categorical formulation of classical structures into linear logic and the typed linear lambda calculus, using ideas from [Has95, HJ95, Lam74]. Before doing so, in order to retain a notion of directionality, some slight adjustments will need to be made to classical structures, as per the way defined in [CPP08a]. The resulting logic and lambda calculus will then need to be checked, in order to make sure the theory is still consistent under β -reduction, and accompanied with proofs of soundness and consistency. Selinger and Valiron worked in the same direction [SV08], providing a complete categorical semantics for a quantum lambda calculus with classical control but modelling classical duplicability using the exponential operator $!$. That approach, however, involves using a category with biproducts, where copyability is defined without axiomatising bases. On the other hand, the approach of my proposed research, similarly to that of [CD08, CP06, CP07, CPP08b, CPV08], defines orthonormal bases in terms of *spider monoids*, replacing the exponential duplicability operator with one which is fundamentally linked to the encoding of the basis. Moreover, the interaction between complementary observables can be used to encode actual quantum operations.

Measurement based quantum computation [RB01, RB02, RBB03, Joz05, BB06] is a computational paradigm which uses a classical computer with control over a quantum resource. More specifically, the classical computer accesses a large entangled state and performs measurements on multiple qubits in specified bases. The classical computer then processes the results

and decides upon what further measurements to make. This computational paradigm matches perfectly with Selinger’s slogan: ”quantum data, classical control” [Sel04a]. Once all the formalism for representing quantum computation with classical control is in place, I plan on applying it to model measurement based quantum computation and the measurement calculus of [DKP07]. This is the second big milestone in my plan for my dissertation. The formalism of \dagger -compact categories and classical structures will faithfully represent both the classical control machine and the underlying entangled graph state, enabling us to reason about measurement based quantum computers in the convenient graphical calculus of [CD08].

In the final part of my proposed research, I will use the insight gained from the aforementioned representations in order to draw some complexity-theoretic results. Ongoing developments in the field of measurement based quantum computation [BK07, AB08] suggest that a number of quantum operations can be performed simultaneously in that paradigm. This parallelisation takes place due to the fact that multiple measurements are performed at the same time and is, to some extent, responsible for the computational speedup we realise in quantum computation [AB08]. The categorical formalism proposed for my research, along with its corresponding graphical calculus, will make it easier to visualise these operations. Moreover, the linearity of logic that permeates the categorical model makes it ideal for the study of concurrency. The ultimate goal of this entire research is to gain a better understanding of the element of quantum computation which provides it with its additional computational power. Different approaches to this question have, so far, produced widely disparate answers. I hope that my research will be able to contribute in the further refinement of some of those answers.

References

- [AB08] Janet Anders and Dan E. Browne. *Computation from Correlation*. (arXiv:0805.1002v2 [quant-ph])
- [Abr93] Samson Abramsky. *Computational interpretations of linear logic*. In *Theoretical Computer Science*, vol. 111, pages 3-57, 1993. (<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.2984>)

- [AC04] Samson Abramsky and Bob Coecke. *A categorical semantics of quantum protocols*. In Proceedings of the 19th IEEE conference on Logic in Computer Science (LiCS'04), IEEE Computer Science Press 2004. (arXiv:quant-ph/0402130v5)
- [AD05] Samson Abramsky and Ross Duncan. *A Categorical Quantum Logic*. In Mathematical Structures in Computer Science, vol. 16, pages 469-489, 2006. (arXiv:quant-ph/0512114v1)
- [BB06] Dan E. Browne and Hans J. Briegel. *One-way Quantum Computation - a tutorial introduction*. In Book chapter, to appear. (arXiv:quant-ph/0603226v2)
- [BK07] Anne Broadbent, Elham Kashefi. *Parallelizing Quantum Circuits*. (arXiv:0704.1736v1 [quant-ph])
- [Blu93] Richard Blute. *Linear logic, coherence and dinaturality*. In Theoretical Computer Science, volume 115, issue 1, pages 3-41, 1993.
- [CD08] Bob Coecke and Ross Duncan. *Interacting quantum observables*. In Proceedings of the 35th International Colloquium on Automata, Languages and Programming, pages 298310, Lecture Notes in Computer Science 5126, Springer-Verlag, 2008.
- [Che07] Eugenia Cheng. *Adjunctions 1*. In TheCatsters Channel, YouTube, September 13, 2007. (<http://www.youtube.com/watch?v=loOJxIOmShE>)
- [Coe06] Bob Coecke. *Introducing categories to the practicing physicist*. In Advanced Studies in Mathematics and Logic, volume 30, pages 45-74, Polimetrica Publishing, 2006. (arXiv:0808.1032v1 [quant-ph])
- [CP06] Bob Coecke and Éric Oliver Paquette. *POVMs and Naimarks theorem without sums*. In Electronic Notes in Theoretical Computer Science, 2006. to appear. (arXiv:quant-ph/0608072)
- [CP07] Bob Coecke and Duško Pavlović. *Quantum measurements without sums*. In Mathematics of Quantum Computing and Technology, G. Chen, L. Kauffman and S. Lamonaco (eds), pages 567604, Taylor and Francis, 2007. (arXiv:quant-ph/0608035)

- [CPP08a] Bob Coecke, Éric Oliver Paquette and Simon Perdrix. *Bases in diagrammatic quantum protocols*. In Electronic Notes in Theoretical Computer Science (ENTCS), volume 218, pages 131-152, 2008. (arXiv:0808.1029v1 [quant-ph])
- [CPP08b] Bob Coecke, Éric Oliver Paquette and Duško Pavlović. *Classical and quantum structuralism*. In Semantic Techniques in Quantum Computation (book), S. Gay and I. Mackie, Eds., Cambridge University Press, 2008. to appear.
- [CPV08] Bob Coecke, Duško Pavlović and Jamie Vicary. *A new description of orthogonal bases*. In Mathematical Structures in Computer Science, pp 13, 2008. to appear. (arXiv:0810.0812v1 [quant-ph])
- [DKP07] Vincent Danos, Elham Kashefi and Prakash Panangaden. *The Measurement Calculus*. In Journal of the ACM (JACM), volume 54, issue 2, article no. 8, 2007. (arXiv:0704.1263v1 [quant-ph])
- [Pav97] Duško Pavlović. *Categorical logic of names and abstraction in action calculi*. In Mathematical Structures in Computer Science, volume 7, issue 6, pages 619-637, Cambridge University Press, 1997.
- [Gir87] Jean-Yves Girard. *Linear Logic*. In Theoretical Computer Science, 50(1), pages 1-102, 1987.
- [Has95] Masahito Hasegawa. *Decomposing typed lambda calculus into a couple of categorical programming languages*. In Proceedings of the 6th International Conference on Category Theory and Computer Science, Lecture Notes In Computer Science, volume 953, pages: 200-219, 1995. (<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.715>)
- [HJ95] Claudio Hermida, Bart Jacobs. *Fibrations with indeterminates: Contextual and functional completeness for polymorphic lambda calculi*. In Mathematical Structures in Computer Science, volume 5, pages: 501-531, Cambridge University Press, 1995. (http://www.narcis.info/dare/RecordID/1375/repository_id/nwodare)
- [JM08] Richard Jozsa, Akimasa Miyake. *Matchgates and classical simulation of quantum circuits*. In Proceedings of the Royal Society A; volume 464, number 2100, pages 3089-3106, DOI: 10.1098/rspa.2008.0189, 2008. (arXiv:0804.4050v2 [quant-ph])

- [Joz05] Richard Jozsa. *An introduction to measurement based quantum computation*. (arXiv:quant-ph/0508124v2)
- [Lam74] Joachim Lambek. *Functional completeness of cartesian categories*. In *Annals of Mathematical Logic*, vol. 6, pages 259-292, 1974.
- [Mac98] Saunders Mac Lane. *Categories for the Working Mathematician*. (book) Second edition, Springer, 1998. (ISBN 0-387-98403-8)
- [Mer07] N. David Mermin. *Quantum Computer Science: An Introduction*. (book) First edition, Cambridge University Press, 2007. (ISBN 978-0521876582)
- [NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. (book), Cambridge University Press, 2000. (ISBN 978-0521635035)
- [PB00] Pati, A. K. and Braunstein, S. L. *Impossibility of deleting an unknown quantum state*. In *Nature* 404, pages 164165, 2000.
- [RB01] Robert Raussendorf and Hans J. Briegel *A one-way quantum computer*. In *Physical Review Letters*, 86, 5188, 2001.
- [RB02] Robert Raussendorf and Hans J. Briegel *Computational model underlying the one-way quantum computer*. *Quantum Information and Computation*, 2, 2002. (arXiv:quant-ph/0108067v2)
- [RBB03] Robert Raussendorf, Dan E. Browne and Hans J. Briegel *Measurement-based quantum computation on cluster states*. In *Physical Review A*, 68, 2003.
- [Sel04a] Peter Selinger. *A brief survey of quantum programming languages*. In *Proceedings of the 7th International Symposium on Functional and Logic Programming*, Nara, Japan. Springer LNCS 2998, pp. 1-6, 2004. (<http://www.mscs.dal.ca/~selinger/papers.html#flops04>)
- [Sel04b] Peter Selinger. *Towards a quantum programming language*. In *Mathematical Structures in Computer Science*, 14(4):527-586, 2004. (<http://www.mscs.dal.ca/~selinger/papers.html#qpl>)

- [Sel05] Peter Selinger. *Dagger compact closed categories and completely positive maps*. In Proceedings of the 3rd International Workshop on Quantum Programming Languages (QPL 2005), Chicago. ENTCS 170:139-163, 2007. (<http://www.mscs.dal.ca/~selinger/papers.html#dagger>)
- [SV06] Peter Selinger, Benoit Valiron. *A lambda calculus for quantum computation with classical control*. In Mathematical Structures in Computer Science, volume 16, issue 3, pages 527-552, 2006. (arXiv:cs/0404056v2 [cs.LO])
- [SV08] Peter Selinger, Benoit Valiron. *A linear-non-linear model for a computational call-by-value lambda calculus (extended abstract)*. In Proceedings of the Eleventh International Conference on Foundations of Software Science and Computation Structures (FOSSACS 2008), Budapest, Springer LNCS 4962, pp. 81-96, 2008. (arXiv:0801.0813v1 [cs.LO])
- [Val02] Leslie G. Valiant. *Quantum Circuits That Can Be Simulated Classically in Polynomial Time*. In SIAM Journal on Computing, volume 31, issue 4, pages 1229-1254, 2002.
- [vT04] André van Tonder. *A Lambda Calculus for Quantum Computation*. In SIAM Journal on Computing, volume 33, issue 5, pages 1109-1135, 2004. (arXiv:quant-ph/0307150v5)
- [vTD07] André van Tonder, Miquel Dorca. *Quantum Computation, Categorical Semantics and Linear Logic*. (arXiv:quant-ph/0312174v4)
- [WZ82] Michael Wootters, W. and Zurek, W. *A single quantum cannot be cloned*. Nature 299, pages 802803, 1982.