Algebraic Effects, Linearity, and Quantum Programming Languages

Sam Staton

Radboud University Nijmegen

Abstract

We develop a new framework of algebraic theories with linear parameters, and use it to analyze the equational reasoning principles of quantum computing and quantum programming languages. We use the framework as follows:

- we present a new elementary algebraic theory of quantum computation, built from unitary gates and measurement;
- we provide a completeness theorem for the elementary algebraic theory by relating it with a model from operator algebra;
- we extract an equational theory for a quantum programming language from the algebraic theory;
- we compare quantum computation with other local notions of computation by investigating variations on the algebraic theory.

1. Introduction

Quantum programming languages test many of the challenges of modern programming language theory: linear use of resources, separation, locality. A good way to understand a programming language is to understand equality of programs. In this paper we develop a general algebraic framework for computational effects involving linear resources. We use it to give a complete axiomatization of equality of quantum programs.

What is quantum computing? From a programming language perspective, quantum computing involves qubits and entanglement:

- *There is a type* qubit *of qubits.* Viewed as an abstract type, we can imagine a qubit as having an internal state that is a position on the surface of a sphere (called the Bloch sphere), but the accessor functions do not actually permit us to read its position on the surface. The three accessor functions are, informally, as follows. (*Notation:* we underline them.)
  - $\underline{\mathsf{new}}$: allocate a new qubit, with initial position at the top of the Z axis (called $|0\rangle$).
  - $\underline{\mathsf{apply}}_{\mathtt{U}}$: apply a rotation to the qubit on the sphere around a given axis by a given angle, as specified by a unitary matrix `U`. For example, we can kind-of negate a qubit by rotating it by 180° around the X axis, taking the top of the sphere to the bottom; this unitary rotation is notated `X`, and so the function that applies the rotation is notated $\underline{\mathsf{apply}}_{\mathtt{X}}$.
  - $\underline{\mathsf{measure}}$: make a random boolean choice, with the probability of returning either 0 or 1 depending on the Z co-ordinate of the qubit (this is called the standard basis). For example, if the qubit was on the X axis, the result of measuring will be 0 or 1 with equal probability, like tossing a fair coin; if it was at the very top of the sphere, the result of measuring will be 0 with certainty; if it was at the very bottom of the sphere, the result of measuring will be 1 with certainty. Measuring a qubit destroys it: all that remains is the result of the measurement.
- *For types A and B, there is a type $A \otimes B$ of entangled pairs.* For instance the type $\mathsf{qubit} \otimes \mathsf{qubit}$ is a type of pairs of possibly entangled qubits. Entanglement is achieved by controlled unitary rotations. For example, the controlled-`X` unitary, `cX`, affects two qubits, and if t is an expression of type $\mathsf{qubit} \otimes \mathsf{qubit}$ then also $\underline{\mathsf{apply}}_{\mathtt{cX}}(t)$ is an expression of type $\mathsf{qubit} \otimes \mathsf{qubit}$. The computation $\underline{\mathsf{apply}}_{\mathtt{cX}}(a, b)$ is like 'if a is 1 then return $(a, \neg b)$ else return (a, b)', so that the second value returned depends on the first value input. The entanglement occurs because this controlled rotation happens without actually measuring a, and indeed it is reversible. Yet if a is subsequently measured then the controlled rotation appears to have behaved in this way.

The main contribution of this paper is the fact that the relationship between unitary rotations and measurement can be completely described by three simple axioms (Theorem 9), and allocation by two simple axioms (Theorem 11). This simple axiomatization (combined with the unitary groups and commutativity laws) completely characterizes earlier models that are built from operator algebra and functional analysis.

In the remainder of this introduction, we give an informal overview of these results. To express the equations, we need to first discuss the syntax.

Quantum programming languages and quantum programs. A quantum programming language captures the ideas of quantum computation in a linear type theory. For example, we can write a program of type $\mathsf{qubit} \to \mathsf{qubit}$:

$$\mathsf{fn}\ a : \mathsf{qubit} \Rightarrow \mathsf{if}\ \underline{\mathsf{measure}}\, a = 0\ \mathsf{then}\ \underline{\mathsf{new}}()\ \mathsf{else}\ \underline{\mathsf{apply}}_{\mathtt{X}}(\underline{\mathsf{new}}()) \tag{1}$$

which measures a qubit and returns a new qubit initialized either to $|0\rangle$ or $|1\rangle$, depending on the outcome. In other words, the input qubit is collapsed on to the Z axis.

For another example of a program, we recall the Hadamard rotation, `Had`, which maps the Z axis onto the X axis by rotating 180° around the axis that lies between the X and Z axes. We can use this to define a coin toss operation: first initialize a qubit to the

POPL '15, January 15–17 2015, Mumbai, India.

ACM 978-1-4503-3300-9/15/01…$15.00.
http://dx.doi.org/10.1145/2676726.2676999

top of the sphere, and then rotate by Hadamard, and then measure: the program

$$\underline{\mathsf{measure}}(\underline{\mathsf{apply}_{\mathtt{Had}}}(\underline{\mathsf{new}}())) \ : \ \mathsf{bit}$$

returns 0 or 1 with equal probability. We can use this to randomly rotate a qubit around the Z axis: writing $\mathtt{Z}$ for the rotation of 180° around the Z axis,

$$\mathsf{fn}\ a : \mathsf{qubit} \Rightarrow \mathsf{if}\ \underline{\mathsf{measure}}(\underline{\mathsf{apply}_{\mathtt{Had}}}(\underline{\mathsf{new}}())) = 0 \\ \mathsf{then}\ a\ \mathsf{else}\ \underline{\mathsf{apply}_{\mathtt{Z}}}(a)\ : \mathsf{qubit} \to \mathsf{qubit} \quad (2)$$

It turns out that this program (2) is actually equal to the first program (1): projecting onto the Z axis is the same as randomly flipping around the Z axis. This fact, perhaps counter-intuitive at first, is a consequence of our simple intuitive axiomatization of quantum computation.

Notions of computation and algebraic theories. In programming language theory it is often useful to analyze different features of a programming language separately. In the quantum programs above there are standard features such as sequencing, if-then-else, functions, as well as the aspects specific to quantum computing.

One way of distinguishing the specific 'notions of computation' from the other standard features of programming languages is by using algebraic effects [36]. A key contribution of this paper (§2) is a new algebraic framework for analyzing the effects of a linear-use resource.

A useful step towards the algebraic analysis is to step away from booleans, if-then-else and sums by writing $\mathsf{measure}(a, t, u)$ for the program $(\mathsf{if}\ \underline{\mathsf{measure}}(a) = 0\ \mathsf{then}\ t\ \mathsf{else}\ u)$. By working with expressions such as $\mathsf{measure}(a, t, u)$, which are in a kind-of continuation-passing normal form, we can forget about structural aspects and typing issues. Indeed to analyze the algebraic theory of quantum computation, we need only distinguish between two kinds of thing: qubits and computations.

Notice our typographic convention: we underline programming-language-style commands ($\underline{\mathsf{measure}}(a)$) but not the corresponding algebraic operations ($\mathsf{measure}(a, t, u)$). We make this correspondence precise in Section 5.

Algebraic structure of quantum computation. We are now in a position to postulate the algebraic structure of quantum computation. It supports four constructions:

- if t is a computation involving a qubit a then there is a computation $\mathsf{new}(a.t)$ that allocates that qubit, initialized to $|0\rangle$, and continues as t;
- if t is a computation involving qubits $b_1 \dots b_n$ and $\mathtt{U}$ is a unitary matrix over n qubits then there is a quantum computation $\mathsf{apply}_{\mathtt{U}}(a_1 \dots a_n,\ b_1 \dots b_n.t)$ that first performs the unitary $\mathtt{U}$ to the qubits $a_1 \dots a_n$ and then binds the resulting qubits $b_1 \dots b_n$ in the continuation t;
- if t and u are computations and a is a qubit then there is a computation $\mathsf{measure}(a, t, u)$ that measures a and continues as either t or u depending on the result of the measurement. This construction is linear in the quantum parameters: t and u cannot contain a.
- if x is a variable standing for a continuation that expects n qubits (notation, $x : n$) and $a_1 \dots a_n$ are qubits, then $x(a_1 \dots a_n)$ is a computation over n qubits.

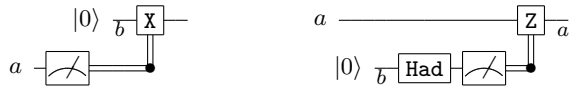
We can construct our two programs, (1) and (2), as algebraic expressions as follows:

$$\mathsf{measure}(a, \mathsf{new}(b.x(b)), \mathsf{new}(b.\mathsf{apply}_{\mathtt{X}}(b, b'.x(b'))))$$

$$\mathsf{new}(b.\mathsf{apply}_{\mathtt{Had}}(b, b'.\mathsf{measure}(b', x(a), \mathsf{apply}_{\mathtt{Z}}(a, a'.x(a')))))$$

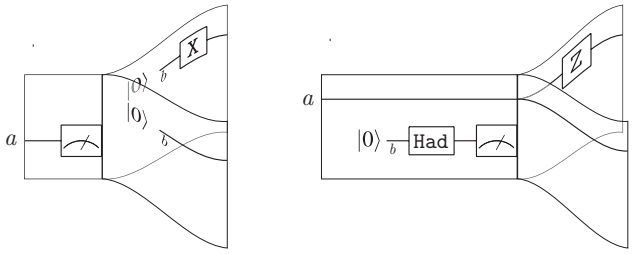
Here x is a variable standing for the continuation of the computation (as yet unspecified) which is parameterized by the result qubit.

Diagrammatic notation. Informal diagrams are often helpful. Our notion of quantum computation is very similar to the notion described by quantum circuits, for instance in the book by Nielsen and Chuang [31]. The two programs above ((1), (2)) could be written as the following circuits:



The circuits should be read from left to right. The single wires carry qubits, the double wires carry classical bits. The boxes are the unitary rotations, and the meter device is measurement, which takes a quantum wire to a classical one. The vertical lines indicate control: the outcome of the measurement determines whether the gate is applied.

Aside: Just as we wanted to isolate notions of quantum computation from other aspects of programming languages, so we ought to isolate the quantum parts of quantum circuits (qubits, unitaries, measurement) from the classical wires. This leads us to a 3-d notation like

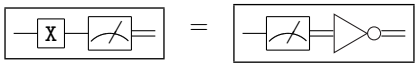


in which measurement causes a branch in the circuit instead of a classical wire. These kinds of diagram are very similar to Melliès' notation for storage of classical bits [28], but they are also common in the many-worlds interpretation of quantum mechanics. In this paper we will only use diagrams in an informal way, and so we will write 2-d circuit diagrams for simplicity. We return to the relationship with classical bits in Section 6.2.

Axioms of quantum computation. We are now in a position to summarize the algebraic laws of quantum computation. A full account is in Section 3. In brief, the laws of measurement are:

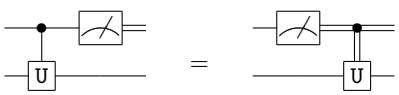
- after measurement, quantum negation is like classical negation:

$$\mathsf{apply}_{\mathtt{X}}(a, a'.\mathsf{measure}(a', x, y)) = \mathsf{measure}(a, y, x)$$



- after measurement, quantum control is like classical control:

$$\mathsf{apply}_{\mathtt{cU}}(a, b, a'b'.\mathsf{measure}(a', x(b'), y(b'))) \\ = \mathsf{measure}(a, x(b), \mathsf{apply}_{\mathtt{U}}(b, b'.y(b')))$$



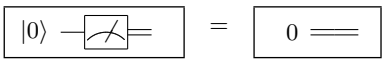
- we can 'discard' a qubit by measuring it and ignoring the result; gates on discarded qubits are redundant:

$$\mathsf{apply}_{\mathtt{U}}(a, a'.\mathsf{measure}(a', x, x)) = \mathsf{measure}(a, x, x)$$

We combine these three axioms with the equations of the unitary groups and the following two sensible axioms for qubit allocation:

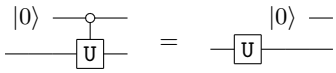
- new qubits are initialized to $|0\rangle$, according to measurement

$$\mathsf{new}(a.\mathsf{measure}(a,x,y)) = x$$



- new qubits are initialized to $|0\rangle$, according to controlled gates:

$$\mathsf{new}(a.\mathsf{apply}_{\bar{\mathtt{cU}}}(a,b,a'b'.x(a',b'))) = \mathsf{apply}_{\mathtt{U}}(b,b'.\mathsf{new}(a.x(a,b')))$$



Completeness theorem. Other authors have proposed equational theories for quantum computation and quantum programming languages. We discuss this in Section 3.5. What makes this paper special is that it is a *complete* equational theory of quantum computation.

Of course, it is very useful to have some handy equations — they can be used to optimize programs and partially evaluate them. But by giving a fully complete equational theory we can understand quantum computation from the axioms of the theory without having to turn to denotational models built from operator algebra. This is the subject of Section 4.

We have the following result.

Theorem 11. *The following data are equivalent:*

- *An algebraic expression, modulo the equality derivable from our axioms;*
- *A completely positive unit-preserving map between finite dimensional C*-algebras.*

(It is already well-known that a completely positive map can be understood in terms of allocating qubits, applying a unitary, and then measuring, and Selinger [42] phrased this in terms of programming languages; the novelty here is our complete axiomatization of equality.)

Thus our simple equational theory provides a justification for the model of quantum computation based on operator algebra. Although linear algebra plays a major role in many models of quantum computation, the non-convex addition in linear algebra often has no direct physical justification. Viewed in this way, our work suggests an alternative technique for reconstructing quantum theory from reasonable postulates, which is a subject of much recent work (e.g. [12, 20]). In our equational theory, vector spaces do *not* have an explicit role. The point is that rather than jumping from physical intuitions directly to linear algebra, one can first set up a precise equational model of the physical intuitions, and then use that to justify the models that use linear algebra.

2. Algebraic theories with linear parameters

In this section we introduce a formalism for describing computational effects over linearly-used resources. Our leading example is quantum computations over qubits. To motivate this, we consider some simple examples. If x and y are quantum computations and a is a qubit, then $\mathsf{measure}(a,x,y)$ is a quantum computation that first measures a, and, depending on the result, continues as x or as y. We can allocate new qubits, writing $\mathsf{new}(a.\mathsf{measure}(a,x,y))$ for the computation that first allocates a, then measures it, continuing as either x or y. We can also apply unitaries to qubits, writing $\mathsf{apply}_{\mathtt{cX}}(a,b,a'b'.\mathsf{measure}(a',\mathsf{measure}(b',v,x),\mathsf{measure}(b',y,z)))$ for the computation that first applies a controlled-not gate to qubits a and b, yielding qubits a' and b', and then measures a' and then b'.

We make some informal remarks about this syntax, before introducing a formal system.

1. There are two kinds of variable: in the examples above, a, b stand for qubits whereas v, x, y, z stand for computations.
2. In the example with new, the a is binding, and we could just as well write $\mathsf{new}(b.\mathsf{measure}(b,x,y))$.
3. Computations can involve qubit parameters a, b, and the variables x, y stand for computations, so we will also allow variables with qubit parameters. When we write $x(a,b)$, the computation variable x is being passed parameters a and b. For instance we can write a computation expression $\mathsf{new}(a.z(a))$ which allocates a new qubit a and passes it as a parameter to the continuation z; we can substitute $\mathsf{measure}(a,x,y)$ for $z(a)$, resulting in the term $\mathsf{new}(a.\mathsf{measure}(a,x,y))$. The notation means that we do not consider implicit variable capture.
4. Care must be taken when using qubits. Measuring a qubit destroys it, and so we adopt the convention that, in an expression $\mathsf{measure}(a,t,u)$, the qubit parameter a should not appear free in t or u. This kind of linearity also plays a crucial role in the syntax for unitaries: in $\mathsf{apply}_{\mathtt{cX}}(a,b,\dots)$ it is crucial that a and b are different qubits and not aliases for the same qubit. For simplicity we adopt the convention that apply consumes its qubit parameters, creating new ones that are passed to the continuation.

2.1 General syntactic framework

Our general syntactic framework is an algebraic framework which is not at all specific to quantum computation. Indeed, it is a new substructural version of the 'parameterized algebraic theories' already used to analyze various kinds of computation including local store, π-calculus-style communication [47] and logic programming [45], following some other similar syntactic frameworks [11, 18, 30, 35, 40]. There are two kinds of variable:

1. Computation variables (ranged over by x, y etc.). Computations can depend on parameters, and we write $x : p$ if x has p parameters. The number p is sometimes called a valence.
2. Parameter variables (ranged over by a, b). In the quantum situation, these stand for qubits.

As in classical algebra, a theory comprises a signature (Def. 1) and axioms (Def. 2).

Definition 1. *An* arity, *$(p \mid m_1 \dots m_k)$, is a natural number $(p \geq 0)$ followed by a list of natural numbers $(m_1 \dots m_k)$.*

A signature with linear parameters *comprises a set of operations, and for each operation* O *an arity $(p \mid m_1 \dots m_k)$. Informally, this specifies that* O *takes p parameter arguments and k computation arguments, and the ith computation argument has m_i parameters bound in it. We write* $\mathrm{O} : (p \mid m_1 \dots m_k)$.

We define a three-place judgement $\Gamma \mid \Delta \vdash t$ of well-formed terms in context in a given signature. Here, Γ is a list of computation variables with their valences, and Δ is a list of parameter variables. The judgement is the least one closed under the following rules.

$$\frac{-}{\Gamma, x : p, \Gamma' \mid a_1 \dots a_p \vdash x(a_1 \dots a_p)}$$

$$\frac{\Gamma \mid a_1 \dots a_p \vdash t}{\Gamma \mid a_{\sigma(1)} \dots a_{\sigma(p)} \vdash t} \quad (\sigma \text{ is a permutation of } p)$$

$$\frac{\begin{array}{c}\Gamma \mid \Delta, b_1 \dots b_{m_1} \vdash t_1 \\ \dots \quad \Gamma \mid \Delta, b_1 \dots b_{m_k} \vdash t_k \qquad \mathrm{O} : (p \mid m_1 \dots m_k)\end{array}}{\Gamma \mid \Delta, a_1 \dots a_p \vdash \mathrm{O}(a_1, \dots, a_p,\ b_1 \dots b_{m_1}.t_1,\ \dots,\ b_1 \dots b_{m_k}.t_k)}$$

In the term formation rule for operations, the b's are binding, and we work up to renaming those bound variables, just as in predicate logic ($\forall x.P(x) = \forall y.P(y)$) or λ-calculus.

We make some remarks about the rules.

- An operation O consumes its parameters, the a's, and they cannot appear free in the continuations, the t's (unless of course one happens to use the same symbol for the a's and the b's).
- Weakening and contraction are admissible in the Γ context but not in the Δ context.
- The syntax admits the following simultaneous substitution law:

$$\frac{x_1 : m_1 \dots x_k : m_k \mid \Delta \vdash t \qquad \begin{array}{c} \Gamma \mid \Xi, a_1 \dots a_{m_1} \vdash u_1 \\ \dots \\ \Gamma \mid \Xi, a_1 \dots a_{m_k} \vdash u_k \end{array}}{\Gamma \mid \Delta, \Xi \vdash t[{}^{\Xi, a_1 \dots a_{m_1} \vdash u_1}/_{x_1} \dots {}^{\Xi, a_1 \dots a_{m_k} \vdash u_k}/_{x_k}]}$$

Definition 2. *Fix an algebraic signature with linear parameters. An* axiom *is a pair of terms in the same context; it is written* $\Gamma \mid \Delta \vdash t = u$.

A presentation of an algebraic theory with linear parameters *comprises an algebraic signature with linear parameters and a set of axioms over the signature.*

In an algebraic theory we form an equivalence relation on terms in each context $(\Gamma|\Delta)$ by closing substitution instances of the axioms under reflexivity, symmetry, transitivity and congruence.

We now proceed to develop the basic model theory of algebraic theories with linear parameters. The reader could now jump to read about the algebraic theory of quantum computation (§3).

2.2 Models of algebraic theories with linear parameters

Classical algebraic theories are typically first understood as set-theoretic structures, but it is often profitable to look at models whose carrier is not merely a set and whose structure maps are not merely functions, that is, to look at models in different categories. In Section 4, we will prove our full completeness result by looking at models whose carriers are C*-algebras.

To account for the parameters, we are urged to look at categories where for each object X and for each natural number n there is a given object $n \bullet X$. We can then think of a morphism $X \to n \bullet Y$ informally as a morphism $X \to Y$ with n parameters. We will interpret operations $\mathrm{O} : (p \mid m_1 \dots m_k)$ as morphisms $(m_1 \bullet X) \times \dots \times (m_k \bullet X) \to (p \bullet X)$.

Let us make this more formal. Let $\mathbf{Bij}$ be the category whose objects are natural numbers and whose morphisms are bijections between the natural numbers (considered as sets). We will consider it as a symmetric monoidal category, where the monoidal operation is addition of numbers. Recall (e.g. [23]) that an *action* of $\mathbf{Bij}$ comprises a category $\mathcal{V}$ together with a functor $\bullet : \mathbf{Bij} \times \mathcal{V} \to \mathcal{V}$ and natural isomorphisms, $0 \bullet X \cong X$ and $(m + n) \bullet X \cong m \bullet (n \bullet X)$, satisfying coherence conditions.

Let $\mathcal{V}$ be an action of $\mathbf{Bij}$ such that the category $\mathcal{V}$ has products and each functor $m \bullet - : \mathcal{V} \to \mathcal{V}$ preserves products.

Definition 3. *A* structure *for a signature in $\mathcal{V}$ is an object X together with, for each operation* $\mathrm{O} : (p \mid m_1 \dots m_k)$ *a morphism* $(m_1 \bullet X) \times \dots \times (m_k \bullet X) \to p \bullet X$.

Given a structure for a signature, one can interpret each term in context $x_1 : m_1 \dots x_k : m_k \mid a_1 \dots a_p \vdash t$ as a morphism in the category, $[\![t]\!] : (m_1 \bullet X) \times \dots \times (m_k \bullet X) \to p \bullet X$. This interpretation is defined by induction on the structure of terms in a standard way. Informally, the interpretation assigns a value in X for each valuation of its variables. Note that exchange of parameters amounts to the functoriality of the action, and the admissibility of substitution amounts to the composition of the morphisms that interpret the terms.

Definition 4. *A* model *of an algebraic theory with linear parameters is a structure for its signature such that for each axiom $x_1 : m_1 \dots x_k : m_k \mid a_1 \dots a_p \vdash t = u$ the interpretation morphisms $[\![t]\!], [\![u]\!] : (m_1 \bullet X) \times \dots \times (m_k \bullet X) \to p \bullet X$ are equal.*

Proposition 5 (Soundness)**.** *If an equality is derivable in a theory then it is is true in all models.*

This is proved by induction on the derivations of equality.

Proposition 6. *Consider terms $x_1{:}m_1 \dots x_k{:}m_k \mid a_1 ... a_p \vdash t, u$ of an algebraic theory that are equal in all models. Their equality is derivable from the axioms of the theory.*

Proof. Fix an algebraic theory with linear parameters. Consider the category of functors $[\mathbf{Bij} \to \mathbf{Set}]$ and natural transformations between them. This has finite products: $(F \times G)(p) = F(p) \times G(p)$. There is an action of $\mathbf{Bij}$ on $[\mathbf{Bij} \to \mathbf{Set}]$ given by $(n \bullet F)(p) = F(p + n)$. For each computation context $\Gamma = (x_1 : m_1 \dots x_k : m_k)$ we define a functor $T_\Gamma : \mathbf{Bij} \to \mathbf{Set}$ with $T_\Gamma(q)$ the set of terms in context $(\Gamma | b_1 \dots b_q)$ modulo the derivable equations. This can be given the structure of a model, in the functor category $[\mathbf{Bij} \to \mathbf{Set}]$: notice that $(p \bullet T_\Gamma)(\Delta)$ is the set of terms in context $(\Gamma \mid \Delta, a_1 \dots a_p)$, and we define the structure maps using the term formation for the operations. In this model, for any term $\Gamma \mid a_1 \dots a_p \vdash t$ we have in particular a function $[\![t]\!]_0 : T_\Gamma(m_1) \times \dots \times T_\Gamma(m_k) \to T_\Gamma(p)$, and by definition, $t = [\![t]\!]_0(x_1(b_1 \dots b_{m_1}), \dots, x_k(b_1 \dots b_{m_k}))$ in $T_\Gamma(p)$. So if two terms $\Gamma \mid a_1 \dots a_p \vdash t, u$ are such that $[\![t]\!] = [\![u]\!]$ in this model, then $t = u$ must be derivable. □

Functors $(\mathbf{Bij} \to \mathbf{Set})$ are called 'species of structure', and have been used to analyze aspects of quantum computation including variations on the Fock space construction (e.g. [7]).

In this paper we are particularly interested in models that are fully complete in the following sense.

Definition 7. *A model X in a category is* fully complete *if for all contexts $(x_1 : m_1 \dots x_k : m_k | a_1 \dots a_p)$,*

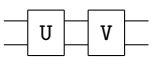
- *for every morphism $f : (m_1 \bullet X) \times \dots (m_k \bullet X) \to (p \bullet X)$ there is a term $x_1 : m_1 \dots x_k : m_k \mid a_1 \dots a_p \vdash t$ such that $[\![t]\!] = f$;*
- *for all terms $x_1 : m_1 \dots x_k : m_k \mid a_1 \dots a_p \vdash t, u$, if $[\![t]\!] = [\![u]\!]$ then $t = u$ is derivable.*

3. An algebraic theory of quantum computation

3.1 Rudiments of unitaries.

The basic idea of quantum computation is that the $2^n \times 2^n$ unitary matrices describe purely quantum circuits over n qubits. We recall some key parts of the theory of unitaries.

Recall that a square matrix $\mathtt{U}$ of complex numbers is *unitary* if its conjugate transpose $\mathtt{U}^*$ is its inverse ($\mathtt{U}.\mathtt{U}^* = \mathtt{I} = \mathtt{U}^*.\mathtt{U}$). Unitaries of the same dimension form a group under matrix multiplication: multiplication of two unitaries is again unitary. If $\mathtt{U}$ and $\mathtt{V}$ are $2^p \times 2^p$ unitaries, then we can use quantum circuit diagrams to illustrate their multiplication by horizontal juxtaposition.



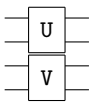
For the simplest example, a 1-dimensional unitary is just a complex number with modulus 1. This is sometimes called a global phase shift. Global phase shifts can ultimately be neglected in quantum computation, indeed that is a consequence of our Axiom (C), but they still play a role in the unitary groups.

Every rotation by some angle about an axis of the sphere can be understood as a 2×2 unitary. In fact, every 2×2 unitary

arises by combining a global phase shift with such a rotation. We are particularly interested in the rotation by π about the X axis, $\mathtt{X} = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, rotation by θ about the Z axis, $\mathtt{Z}^\theta = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{smallmatrix}\right)$, and the Hadamard rotation, $\mathtt{Had} = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$.

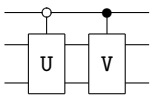
To explain the different ways of combining unitaries, we note that the collection of all finite unitaries of different dimensions forms a groupoid, that is, a category where all morphisms are isomorphisms. The objects are natural numbers, and a morphism $\mathtt{U} : n \to n$ is an $n \times n$ unitary. Recall that a monoidal structure on a category is a way of combining objects and morphisms. There are two symmetric monoidal structures on this groupoid of unitaries, coming from the multiplication and addition of dimensions.

One monoidal structure is multiplication: if $\mathtt{U}$ and $\mathtt{V}$ are unitaries, $m \times m$ and $n \times n$ respectively, the Kronecker product $\mathtt{U} \otimes \mathtt{V}$ is a unitary $(m \times n) \times (m \times n)$ matrix: $(\mathtt{U} \otimes \mathtt{V})_{i+mk,j+ml} = \mathtt{U}_{i,j}\mathtt{U}_{k+1,l+1}$. When $m = 2^p$ and $n = 2^q$ are powers of two, so $m \times n = 2^{p+q}$, we can illustrate this structure on quantum circuit diagrams as vertical juxtaposition.



The other monoidal structure is addition of dimensions: if $\mathtt{U}$ and $\mathtt{V}$ are unitaries, $m \times m$ and $n \times n$ respectively, the block diagonal $(m+n)\times(m+n)$ matrix $\left(\begin{smallmatrix} \mathtt{U} & 0 \\ 0 & \mathtt{V} \end{smallmatrix}\right)$ is also unitary. In general we write $D(\mathtt{U}_1, \ldots, \mathtt{U}_k)$ for a block diagonal.

The block diagonal unitaries are controlled unitaries. The smallest example is $\mathtt{Z}^\theta$, which can be thought of as a controlled global phase shift. More generally if $\mathtt{U}$ and $\mathtt{V}$ are $2^p \times 2^p$ unitaries, then $D(\mathtt{U}, \mathtt{V})$ is a $2^{p+1} \times 2^{p+1}$ unitary, illustrated as follows:



(3)

In particular the block diagonal $\mathtt{cX} \stackrel{\text{def}}{=} D(\mathtt{I}_2, \mathtt{X})$ is sometimes called 'controlled not'.

We have already seen the symmetry structure for $+$ at dimension 1: it is the $\mathtt{X}$ gate. The symmetry for the multiplication monoidal structure is $\mathtt{swap} \stackrel{\text{def}}{=} D(1, \mathtt{X}, 1)$.

In fact, every $2^p \times 2^p$ unitary can be built from the two monoidal structures and the single qubit unitaries (e.g. [31, §4.5.2]).

Remark: Our aim in this paper is to study the interaction between the unitaries and qubit allocation and measurement. There is a body of work on axiomatizing unitaries (e.g. [43]), which is complementary to our aims. Moreover the groupoid of unitaries has topological structure. We are ignoring this continuity in this paper, but it is important from a foundational point of view (e.g. [20, Axiom P4']), and it is central to the study of approximation of quantum gates (e.g. [31, Ch. 4.5]) and the notions of estimation inherent to quantum algorithms (e.g. [31, Ch. 5]).

3.2 The signature of quantum computation

The signature for quantum computation comprises the following operations.

$$\mathsf{new} : (0 \mid 1) \qquad \mathsf{measure} : (1 \mid 0, 0)$$
$$\mathsf{apply}_{\mathtt{U}} : (n \mid n) \quad \text{for every } 2^n \times 2^n \text{ unitary } \mathtt{U}$$

Explicitly, these operations induce the following term formation rules:

$$\frac{\Gamma \mid \Delta, a \vdash t}{\Gamma \mid \Delta \vdash \mathsf{new}(a.t)} \qquad \frac{\Gamma \mid \Delta \vdash t \quad \Gamma \mid \Delta \vdash u}{\Gamma \mid \Delta, a \vdash \mathsf{measure}(a, t, u)}$$

$$\frac{\Gamma \mid \Delta, b_1 \ldots b_n \vdash t}{\Gamma \mid \Delta, a_1, \ldots a_n \vdash \mathsf{apply}_{\mathtt{U}}(\vec{a},\ \vec{b}.t)}$$

together with the variables and exchange law (§2.1). (There are no closed terms in this theory, in common with continuation passing style.)

Examples

1. We can make a new qubit initialized to $|1\rangle$ and continue as x.

   $$x : 1 \mid - \vdash \mathsf{new}(a.\mathsf{apply}_{\mathtt{X}}(a, b.x(b)))$$

   This is justified by the term formation rules:

   $$\dfrac{\dfrac{\overline{x : 1 \mid b \vdash x(b)}}{x : 1 \mid a \vdash \mathsf{apply}_{\mathtt{X}}(a, b.x(b))}}{x : 1 \mid - \vdash \mathsf{new}(a.\mathsf{apply}_{\mathtt{X}}(a, b.x(b)))}$$

   Here, the a and b are binding: this expression is equal to $\mathsf{new}(c.\mathsf{apply}_{\mathtt{X}}(c, d.x(d)))$ and also equal to the expression $\mathsf{new}(a.\mathsf{apply}_{\mathtt{X}}(a, a.x(a)))$. The notation $x(b)$ indicates that the continuation x takes a parameter b.

   A convenient shorthand:

   $$\mathsf{new}_0(a.t) \stackrel{\text{def}}{=} \mathsf{new}(a.t), \quad \mathsf{new}_1(a.t) \stackrel{\text{def}}{=} \mathsf{new}(a.\mathsf{apply}_{\mathtt{X}}(a, a.t)).$$

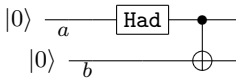
2. We can use quantum operations to make a random choice between continuations x and y.

   $$x : 0, y : 0 \mid - \vdash \mathsf{new}(a.\mathsf{apply}_{\mathtt{Had}}(a, b.\mathsf{measure}(b, x, y)))$$

3. A Bell state comprises two entangled qubits. We can create a Bell state and pass it to x:

   $$x : 2 \mid - \vdash \mathsf{new}(a.\mathsf{new}(b.\mathsf{apply}_{\mathtt{Had}}(a, a.\ \mathsf{apply}_{D(\mathtt{I}_2,\mathtt{X})}(a, b,\ ab.x(a, b)))))$$

   This would be written as the following circuit diagram:

4. The linearity constraints mean that we cannot implicitly discard nor duplicate qubits. However we can explicitly discard a qubit a, by measuring it and ignoring the result.

   $$x : 0 \mid a \vdash \mathsf{measure}(a, x, x)$$

5. In our formalism, measurement consumes the qubit parameter. Another convention is that measurement retains the qubit but it is now collapsed into one of the basis states. This can be simulated by immediately creating a new qubit with the result of the measurement:

   $$x : 1 \mid a \vdash \mathsf{measure}(a, \mathsf{new}_0(a.x(a)), \mathsf{new}_1(a.x(a)))$$

6. In fact, we will later show that (5) is the same as randomly rotating the phase of a by π, using the rotation $\mathtt{Z} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$.

   $$x : 1 \mid a \vdash \mathsf{new}_0(b.\mathsf{apply}_{\mathtt{Had}}(b, b.\mathsf{measure}(b, x(a), \mathsf{apply}_{\mathtt{Z}}(a, a.x(a)))))$$

3.3 Axioms of quantum computation

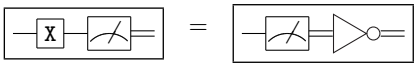
The terms built from new, apply, measure and variables are subject to the following laws. The axioms are of four main kinds. There are two classes of interesting axioms, relating unitaries and measurement (A–C), and relating qubit allocation with unitaries and measurement (D–E). The remaining axioms are more administrative,

relating the properties of unitaries with composition in the syntax (F–I), and commutativity of the theory (J–L).

Interaction between unitary gates and measurement. Our first set of axioms describe the interaction between the unitaries and the standard basis measurement operations. Axiom (A) states that the quantum not gate (`X`) simply negates a measurement.

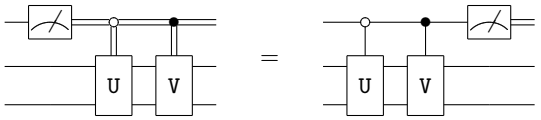
$$\mathsf{apply}_{\mathtt{X}}(a, a.\mathsf{measure}(a, x, y)) = \mathsf{measure}(a, y, x) \qquad \text{(A)}$$

Informally,



Axiom scheme (B) states that after measurement, quantum control appears the same as classical control. Recall that $D(\mathtt{U}, \mathtt{V})$ is the block diagonal matrix, $\left(\begin{smallmatrix} \mathtt{U} & 0 \\ 0 & \mathtt{V} \end{smallmatrix}\right)$.

$$\mathsf{measure}(a, \mathsf{apply}_{\mathtt{U}}(\vec{b}, \vec{b}.x(\vec{b})), \mathsf{apply}_{\mathtt{V}}(\vec{b}, \vec{b}.y(\vec{b})) = \mathsf{apply}_{D(\mathtt{U},\mathtt{V})}(a, \vec{b},\ (a, \vec{b}).\mathsf{measure}(a, x(\vec{b}), y(\vec{b}))) \qquad \text{(B)}$$



(B)

For Axiom (C) we need some shorthand. For a list of p distinct parameter variables $a_1 \dots a_p$ and a term t, define a term $\mathsf{discard}_p(a_1 \dots a_p, t)$, informally 'measure $a_1 \dots a_p$ and continue as t regardless'; formally:

$$\mathsf{discard}_0(-, t) = t;$$

$$\mathsf{discard}_{n+1}(a, \vec{b}, t) = \mathsf{measure}(a, \mathsf{discard}_n(\vec{b}, t), \mathsf{discard}_n(\vec{b}, t)).$$

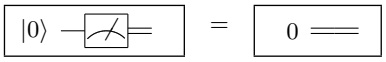
We can now phrase Axiom scheme (C), which asserts that if all the qubits involved in a unitary are to be discarded, the unitary is redundant.

$$\mathsf{apply}_{\mathtt{U}}(\vec{a},\ \vec{a}.\mathsf{discard}_n(\vec{a}, x)) = \mathsf{discard}_n(\vec{a}, x) \quad \text{where } \mathtt{U} \text{ is a } 2^n \times 2^n \text{ unitary},\ n \geq 0 \qquad \text{(C)}$$

In particular, Axiom (C) for $n = 0$ says when global phase can be ignored.

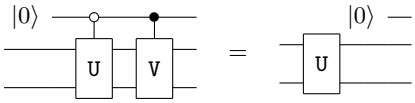
Axioms for qubit allocation: Our second class of axioms describe the interaction between new and measurement and controlled unitaries. They simply impose the idea that new qubits are initialized to $|0\rangle$. Firstly measuring a new qubit always yields 0:

$$\mathsf{new}(a.\mathsf{measure}(a, x, y)) = x \qquad \text{(D)}$$



(This axiom is similar to Selinger's axiom for quantum flow charts [42, §6.6].) Secondly a unitary controlled by a new qubit will always be controlled by 0.

$$\mathsf{new}(a.\mathsf{apply}_{D(\mathtt{U},\mathtt{V})}(a, \vec{b},\ a, \vec{b}.x(a, \vec{b}))) = \mathsf{apply}_{\mathtt{U}}(\vec{b}, \vec{b}.\mathsf{new}(a.x(a, \vec{b}))) \qquad \text{(E)}$$



This concludes the interesting axioms. The remaining axioms are more administrative.

Respecting the symmetric monoidal groupoid of unitaries. Our third class of axioms imposes the relationships between the structure of the unitaries and the compositional structure of terms built from $\mathsf{apply}_{\mathtt{U}}$. We can understand axioms (A) and (B) as relating the $+$ monoidal structure of the groupoid of unitaries with the branching structure after measurement. The remaining axiom schemes (F)-(I) relate the other structure of the groupoid (composition and the $\times$ monoidal structure) with the syntax of the algebraic theory.

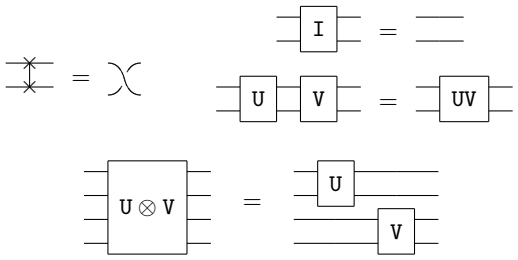
$$\mathsf{apply}_{\mathtt{swap}}(a, b, (a, b).x(a, b)) = x(b, a) \qquad \text{(F)}$$

$$\mathsf{apply}_{\mathtt{I}}(\vec{a}, \vec{a}.x(\vec{a})) = x(\vec{a}) \qquad \text{(G)}$$

$$\mathsf{apply}_{\mathtt{UV}}(\vec{a}, \vec{a}.x(\vec{a})) = \mathsf{apply}_{\mathtt{U}}(\vec{a}, \vec{a}.\mathsf{apply}_{\mathtt{V}}(\vec{a}, \vec{a}.x(\vec{a}))) \qquad \text{(H)}$$

$$\mathsf{apply}_{\mathtt{U}\otimes\mathtt{V}}(\vec{a}, \vec{b}, (\vec{a}, \vec{b}).x(\vec{a}, \vec{b})) = \mathsf{apply}_{\mathtt{U}}(\vec{a}, \vec{a}.\mathsf{apply}_{\mathtt{V}}(\vec{b}, \vec{b}.x(\vec{a}, \vec{b}))) \qquad \text{(I)}$$

Here are informal illustrations of these equations.

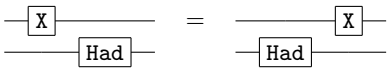


Commutativity. Our final class of axioms ensure that our equational theory is commutative in the sense of [26]. In this parameterized setting, this means that all operations commute as far as is allowed by the variable binding. For instance, the following commutativity equation scheme is already derivable from (I):

$$\mathsf{apply}_{\mathtt{U}}(\vec{b}, \vec{b}.\mathsf{apply}_{\mathtt{V}}(\vec{c}, \vec{c}.x(\vec{b}, \vec{c})) = \mathsf{apply}_{\mathtt{V}}(\vec{c}, \vec{c}.\mathsf{apply}_{\mathtt{U}}(\vec{b}, \vec{b}.x(\vec{b}, \vec{c}))$$

This is in spite of the fact that multiplication of unitaries is not commutative, e.g. `X.Had` $\neq$ `Had.X`. For instance, while

$$\mathsf{apply}_{\mathtt{X}}(a, a.\mathsf{apply}_{\mathtt{Had}}(b, b.x(a, b))) = \mathsf{apply}_{\mathtt{Had}}(b, b.\mathsf{apply}_{\mathtt{X}}(a, a.x(a, b)))$$



this does not imply that $\mathsf{apply}_{\mathtt{X}}(a, b.\mathsf{apply}_{\mathtt{Had}}(b, c.x(c)))$ is equal to $\mathsf{apply}_{\mathtt{Had}}(a, b.\mathsf{apply}_{\mathtt{X}}(b, c.x(c)))$: it is not a substitution instance, because one must respect the variable binding. (This is an example of a general technique for converting a non-commutative algebraic structure to a commutative one by passing linear-use parameters; see also [30].)

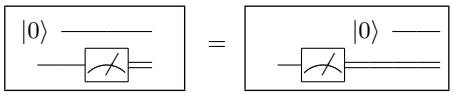
Several other commutativity equations also follow from our other axioms, but we need to explicitly include the following commutativity axioms:

$$\mathsf{measure}(a, \mathsf{measure}(b, u, v), \mathsf{measure}(b, x, y)) = \mathsf{measure}(b, \mathsf{measure}(a, u, x), \mathsf{measure}(a, v, y)) \qquad \text{(J)}$$

$$\mathsf{new}(a.\mathsf{new}(b.x(a, b))) = \mathsf{new}(b.\mathsf{new}(a.x(a, b))) \qquad \text{(K)}$$

$$\mathsf{new}(a.\mathsf{measure}(b, x(a), y(a))) = \mathsf{measure}(b, \mathsf{new}(a.x(a)), \mathsf{new}(a.y(a))) \qquad \text{(L)}$$

Commutativity laws are essentially built in to the quantum circuits notation. For instance Axiom (L) could be informally written



This concludes our axiomatization of quantum computation.

3.4 Examples of derivations

Rotation about Z doesn't affect standard basis measurement. Let $\mathtt{Z}^\theta = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{smallmatrix}\right) = D(1, e^{i\theta})$. Then

$$\begin{aligned}
&\mathsf{apply}_{\mathtt{Z}^\theta}(a, a.\mathsf{measure}(a, x, y)) \\
&\quad = \mathsf{apply}_{D(1,e^{i\theta})}(a, a.\mathsf{measure}(a, x, y)) \\
&\quad = \mathsf{measure}(a, \mathsf{apply}_1(x), \mathsf{apply}_{e^{i\theta}}(y)) && \textit{by (B)} \\
&\quad = \mathsf{measure}(a, x, y) && \textit{by (C).}
\end{aligned}$$

Similarly rotation of a new qubit about Z doesn't affect it:

$$\begin{aligned}
&\mathsf{new}(a.\mathsf{apply}_{\mathtt{Z}^\theta}(a, a.x(a))) \\
&\quad = \mathsf{new}(a.\mathsf{apply}_1(x(a))) && \textit{by (E)} \\
&\quad = \mathsf{new}(a.x(a)) && \textit{by (C).} \qquad (4)
\end{aligned}$$

Notation. To save space we introduce shorthand:

$$\nu a.\, t \stackrel{\text{def}}{=} \mathsf{new}(a.t) \qquad t \mathbin{?}_a u \stackrel{\text{def}}{=} \mathsf{measure}(a, t, u)$$

$$\mathtt{U}_{\vec{a}}(t) \stackrel{\text{def}}{=} \mathsf{apply}_{\mathtt{U}}(\vec{a}, \vec{a}.t) \quad \mathsf{disc}_{\vec{a}}(t) \stackrel{\text{def}}{=} \mathsf{d}_{\vec{a}}(t) \stackrel{\text{def}}{=} \mathsf{discard}(\vec{a}, t)$$

(Note that we use the same variable names for the input and output of the unitary gates.) For example, equation (4) would be written $\nu a.\, \mathtt{Z}^\theta_a(x(a)) = \nu a.\, x(a)$, and the key axioms would be written

$$\begin{aligned}
\mathtt{X}_a(x \mathbin{?}_a y) &= y \mathbin{?}_a x && \textit{(A)} \\
(\mathtt{U}_{\vec{b}}(x)(\vec{b})) \mathbin{?}_a (\mathtt{V}_{\vec{b}}(y)(\vec{b})) &= D(\mathtt{U}, \mathtt{V})_{a,\vec{b}}(x(\vec{b}) \mathbin{?}_a y(\vec{b})) && \textit{(B)} \\
\mathtt{U}_{\vec{a}}(\mathsf{discard}(\vec{a}, x)) &= \mathsf{discard}(\vec{a}, x) && \textit{(C)} \\
\nu a.\, (x \mathbin{?}_a y) &= x && \textit{(D)} \\
\nu a.\, D(\mathtt{U}, \mathtt{V})_{a,\vec{b}}(x(a, \vec{b})) &= \mathtt{U}_{\vec{b}}(\nu a.\, x(a, \vec{b})) && \textit{(E)}
\end{aligned}$$

Random choice is symmetric. Recall that $\nu a.\, \mathtt{Had}_a(x \mathbin{?}_a y)$ provides a random choice between x and y. We deduce that it is unbiased:

$$\begin{aligned}
\nu a.\, \mathtt{Had}_a(x \mathbin{?}_a y) &= \nu a.\, \mathtt{Z}_a(\mathtt{Had}_a(x \mathbin{?}_a y)) && \textit{using (4)} \\
&= \nu a.\, \mathtt{Had}_a(\mathtt{X}_a(x \mathbin{?}_a y)) && \textit{since } \mathtt{Z.Had} = \mathtt{Had.X} \\
&= \nu a.\, \mathtt{Had}_a(y \mathbin{?}_a x). && \textit{using (A)} \qquad (5)
\end{aligned}$$

NB. Random choice is also idempotent and medial, so it is a mean value algebra (e.g. [2]).

Random phase flip = measurement. Consider the following equation:

$$\nu a.\, \mathtt{Had}_a(x(b) \mathbin{?}_a \mathtt{Z}_b(x(b))) = (\nu a.\, x(a)) \mathbin{?}_b (\nu a.\, \mathtt{X}_a(x(a)))$$

In full:

$$\begin{aligned}
&\mathsf{new}_0(a.\mathsf{apply}_{\mathtt{Had}}(a, a.\mathsf{measure}(a, x(b), \mathsf{apply}_{\mathtt{Z}}(b, b.x(b))))) \\
&\qquad = \mathsf{measure}(b, \mathsf{new}_0(a.x(a)), \mathsf{new}_1(a.x(a)))
\end{aligned}$$

It says that randomly flipping the phase of a qubit is the same as measuring it. Nielsen and Chuang discuss the equation to demonstrate the freedom in the operator-sum representation [31, eqns 8.66–8.70]. We can prove it directly in our equational theory.

First we show that measuring a qubit is the same as making an entangled copy using controlled not, $\mathtt{cX} = D(\mathtt{I}_2, \mathtt{X})$, and then discarding the original:

$$\begin{aligned}
&(\nu a.\, x(a)) \mathbin{?}_b (\nu a.\, \mathtt{X}_a(x(a))) \\
&\quad = \nu a.\, (x(a) \mathbin{?}_b \mathtt{X}_a(x(a))) && \textit{using (L)} \\
&\quad = \nu a.\, \mathtt{cX}_{b,a}(x(a) \mathbin{?}_b x(a)) && \textit{using (B)} \\
&\quad = \nu a.\, \mathtt{cX}_{b,a}(\mathsf{disc}_b(x(a))) \\
&\quad = \nu a.\, \mathtt{cX}_{a,b}(\mathtt{cX}_{b,a}(\mathsf{disc}_a(x(b)))) && (\dagger) \\
&\quad = \nu a.\, \mathtt{cX}_{b,a}(\mathsf{disc}_a(x(b))) && \textit{using (E)}
\end{aligned}$$

†: since $\mathtt{cX.swap.cX} = \mathtt{swap.cX.swap}$ as matrices.

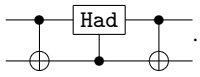
We conclude by showing that the last line is the same as randomly flipping the phase.

$$\begin{aligned}
&\nu a.\, \mathtt{cX}_{b,a}(\mathsf{disc}_a(x(b))) \\
&= \nu a.\, \mathtt{Had}_a(\mathtt{cZ}_{a,b}(\mathtt{Had}_a(\mathsf{disc}_a(x(b))))) && (\ddagger) \\
&= \nu a.\, \mathtt{Had}_a(\mathtt{cZ}_{a,b}(\mathsf{disc}_a(x(b)))) && \textit{using (C)} \\
&= \nu a.\, \mathtt{Had}_a(x(b) \mathbin{?}_a \mathtt{Z}_b(x(b))) && \textit{using (B).}
\end{aligned}$$

‡: since $\mathtt{swap.cX.swap} = (\mathtt{Had} \otimes I).\mathtt{cZ}.(\mathtt{Had} \otimes I)$, and where $\mathtt{cZ} = D(I_2, \mathtt{Z})$.

Reasoning without qutrits. The following example illustrates a key point in the proof of our completeness theorem (Thm. 11). When reasoning in terms of C*-algebras, one has access to various structures that are not definable in our syntax as it stands, such as base 3 quantum digits, 'qutrits'. We could extend our syntax to have parameter variables of different sorts, for different bases. In fact, we do not need these structures to deduce the relevant equations between computations over qubits. Let

$$\mathtt{U} = D(1, \mathtt{Had}, 1) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} =$$



.

Consider the following equation:

$$\begin{aligned}
&\nu b.\, \mathtt{U}_{a,b}(x(a) \mathbin{?}_b (y \mathbin{?}_a u)) \\
&= \nu b.\, \mathtt{U}_{a,b}(x(a) \mathbin{?}_b (y \mathbin{?}_a v))
\end{aligned} \qquad (6)$$

By considering this equation for all x, y, u, v, we have an equational way of expressing the idea that, after the unitary, the pair (a, b) of qubits will never be in the state $|1, 1\rangle$. In effect we have a qutrit, although qutrits are not expressible in this formalism. We derive equation 6 by introducing an intermediate qubit c, as follows. Let $\mathtt{T} = D(\mathtt{I}_6, \mathtt{X})$, the 8×8 Toffoli gate, and note that if $\mathtt{V}$ is a block 3-1 matrix then $\mathtt{T}.(\mathtt{V} \otimes \mathtt{I}_2) = (\mathtt{V} \otimes \mathtt{I}_2).\mathtt{T}$.

$$\begin{aligned}
&\nu b.\, \mathtt{U}_{a,b}(x(a) \mathbin{?}_b (y \mathbin{?}_a u)) \\
&= \nu b.\, \mathtt{U}_{ab}((\nu c.\, \mathsf{d}_c(x(a))) \mathbin{?}_b ((\nu c.\, \mathsf{d}_c(y)) \mathbin{?}_a (\nu c.\, u ?_c v))) && \textit{by (D)} \\
&= \nu c.\, \nu b.\, \mathtt{U}_{a,b}(\mathsf{d}_c(x(a)) \mathbin{?}_b (\mathsf{d}_c(y) \mathbin{?}_a (u \mathbin{?}_c v))) && \textit{(commutativity)} \\
&= \nu c.\, \nu b.\, (\mathtt{U} \otimes \mathtt{I})_{abc}(\mathsf{d}_c(x(a)) \mathbin{?}_b (\mathsf{d}_c(y) \mathbin{?}_a (u \mathbin{?}_c v))) && \textit{(G),(I)} \\
&= \nu c.\, \nu b.\, \mathtt{T}.(\mathtt{U} \otimes \mathtt{I})_{abc}(\mathsf{d}_c(x(a)) \mathbin{?}_b (\mathsf{d}_c(y) \mathbin{?}_a (u ?_c v))) && \textit{(E)} \\
&= \nu c.\, \nu b.\, (\mathtt{U} \otimes \mathtt{I}).\mathtt{T}_{abc}(\mathsf{d}_c(x(a)) \mathbin{?}_b (\mathsf{d}_c(y) \mathbin{?}_a (u ?_c v))) \\
&= \nu c.\, \nu b.\, (\mathtt{U} \otimes \mathtt{I})_{abc}(\mathsf{d}_c(x(a)) \mathbin{?}_b (\mathsf{d}_c(y) \mathbin{?}_a \mathtt{X}_c(u ?_c v))) && \textit{(B)} \\
&= \nu c.\, \nu b.\, (\mathtt{U} \otimes \mathtt{I})_{a,b,c}(\mathsf{d}_c(x(a)) \mathbin{?}_b (\mathsf{d}_c(y) \mathbin{?}_a (v \mathbin{?}_c u))) && \textit{(A)} \\
&= \nu c.\, \nu b.\, \mathtt{U}_{a,b}(\mathsf{d}_c(x(a)) \mathbin{?}_b (\mathsf{d}_c(y) \mathbin{?}_a (v \mathbin{?}_c u))) && \textit{(G),(I)} \\
&= \nu b.\, \mathtt{U}_{a,b}(x(a) \mathbin{?}_b (y \mathbin{?}_a v)) && \textit{(D).}
\end{aligned}$$

3.5 Other axiomatizations of quantum computation

Equational reasoning about programs is widely regarded as important, and several authors have already proposed useful equations for reasoning about quantum programs (e.g. [42, §4.9], [55, §6], and [53]), and other aspects of quantum computation [51, 54].

There are two particular equational formalisms that we would like to mention especially: the measurement calculus [15] and the ZX calculus [14] (which comes out of the categorical quantum mechanics programme, see also [1, 52]). Unlike our theory, neither of these axiomatizations of equality are complete for quantum computation. But they are both in a different spirit: they have strong results regarding term rewriting. It seems unlikely that there is a straightforward rewriting system to fully decide equality of quantum computations, firstly because of problems with rewriting expressions over the unitary groups, and secondly because our derivation of (6) was not at all automatic: it required some ingenuity.

4. Full completeness

In the previous section we provided a new algebraic description of quantum computation by giving axioms that express the relationship between the unitary gates, measurement and allocation. We now show that this algebraic theory completely characterizes quantum computation.

We show this by reference to a well-established model of quantum mechanics, over 60 years old, based on operator algebra and C*-algebras. We show that terms in our theory correspond bijectively with completely positive maps. Thus, via our axiomatization, one can fully understand quantum computation without first knowing all about operator algebra. Conversely, linear algebra gives an array of techniques for reasoning about the quantum computation that we have axiomatized: one can decide equality of computations by comparing matrix representations of the linear maps.

4.1 Rudiments of C*-algebras

The idea of matrix mechanics is that the observables of a quantum system should be elements of a C*-algebra. Recall that a C*-algebra is a vector space over the field of complex numbers that also has multiplication, a unit and an involution, satisfying associativity laws for multiplication, involution laws (e.g. $x^{**} = x$, $(xy)^* = y^*x^*$, $(\alpha x)^* = \bar{\alpha}(x^*)$) and such that the spectral radius provides a norm making it a Banach space. A *-homomorphism between C*-algebras is a linear map that preserves the multiplication, involution and unit. We write $\mathbf{Cstar}$ for the category of C*-algebras and *-homomorphisms.

A key source of examples of C*-algebras are the algebras M_k of $k \times k$ complex matrices, with matrix addition and multiplication, and where involution is conjugate transpose. In particular the set $M_1 = \mathbb{C}$ of complex numbers has a C*-algebra structure, and the 2×2 matrices, M_2, form the C*-algebra containing the observables of qubits.

The 'direct sum' $X \oplus Y$ of C*-algebras is given by the cartesian product of the underlying sets. It has the universal property of the categorical product. The C*-algebra $\mathbb{C}\oplus\mathbb{C}$ represents classical bits.

If X is a C*-algebra then the $k \times k$ matrices valued in X form a C*-algebra, $M_k(X)$. For instance $M_k(\mathbb{C}) = M_k$, and $M_k(M_l) \cong M_{k\times l}$. Informally, we can think of the C*-algebra $M_k(X)$ as representing k entangled copies of X.

Any linear map $f : X \to Y$ extends in the obvious way to a linear map $M_k(f) : M_k(X) \to M_k(Y)$, and $M_k(f)$ is a *-homomorphism if f is. We can thus use this matrix construction to understand $\mathbf{Cstar}$ as a category suitable for modelling algebraic theories with linear parameters. Consider the action of $\mathbf{Bij}$ on $\mathbf{Cstar}$ given by $m \bullet X = M_{2^m}(X)$. The direct sum distributes over the action: $M_k(X \oplus Y) \cong M_k(X) \oplus M_k(Y)$.

In Section 2.2 we defined a notion of model for an algebraic theory with linear parameters. We now investigate models of the theory of quantum computation (§3) whose carrier is the complex numbers, considered as a C*-algebra. Note that $m \bullet \mathbb{C} = M_{2^m}$, the C*-algebra of $2^m \times 2^m$ complex matrices, representing m entangled qubits. Thus a term $a_1 \dots a_p \mid x_1 : m_1 \dots x_k : m_k \vdash t$ is interpreted as a linear map $[\![t]\!] : M_{m_1} \times \cdots \times M_{m_k} \to M_p$. This can be read, informally, as in predicate transformer semantics: 'if $[\![t]\!](o_1 \dots o_k)$ is observed of $a_1 \dots a_p$ then $o_1 \dots o_k$ will be observed of $x_1 \dots x_k$'. Our interpretation of quantum programs as maps between C*-algebras follows other recent work in this direction (e.g. [13, 22, 41]).

4.2 Full completeness for measurement

We begin with the subtheory built from measurement and unitaries. We will add qubit allocation in Section 4.3. The operations $\mathsf{measure}$ and and $\mathsf{apply}_{\mathtt{U}}$ are interpreted using the following *-homomorphisms, $\mathsf{measure} : M_1 \oplus M_1 \to M_2$ and $\mathsf{apply}_{\mathtt{U}} : M_p \to M_p$ (for each $p \times p$ unitary matrix $\mathtt{U}$).

$$\mathsf{measure}(\alpha, \beta) = \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix} \qquad \mathsf{apply}_{\mathtt{U}}(A) = \mathtt{U}^* A \mathtt{U}.$$

Proposition 8. *In the category of C*-algebras and *-homomorphisms, the complex numbers $\mathbb{C}$ form a model of the subtheory of quantum computation involving* $\mathsf{measure}$ *and* apply_U *(but not* new*), and all the relevant axioms ((A)–(C), (F)–(J)).*

This is shown by direct calculation.

Theorem 9. *The complex numbers form a fully complete model:*

1. *For any *-homomorphism $f : M_{2^{m_1}} \oplus \cdots \oplus M_{2^{m_k}} \to M_{2^p}$ there is a term in the algebraic theory not involving* new*, $x_1 : m_1 \dots x_k : m_k \mid a_1 \dots a_p \vdash t$, such that $f = [\![t]\!]$.*
2. *If $\Gamma \mid \Delta \vdash t, u$ and t and u do not contain* new*, and $[\![t]\!] = [\![u]\!]$ then $\Gamma \mid \Delta \vdash t = u$ is derivable.*

We give a rough outline of our proof here, some more detail is in the Appendix. The theorem is proved by noting that a term not involving new can always be rearranged to a term $\mathsf{apply}_{\mathtt{U}}(\vec{a}, \vec{a}.t)$ where t is built from $\mathsf{measure}$ and variables. We can understand a term built only from $\mathsf{measure}$ as a Bratteli diagram, which is a combinatorial way of understanding the *-homomorphisms.

4.3 Full completeness for all quantum computations

Completely positive unital maps. To interpret allocation, we move beyond *-homomorphisms. Recall that an element x of a C*-algebra is called *positive* if $\exists y.\, x = y^*y$. A linear map $f : X \to Y$ is *completely positive* if for all k the map $M_k(f) : M_k(X) \to M_k(Y)$ preserves positive elements. (If either X or Y has commutative multiplication then it is sufficient to check the case $k = 1$.) We will focus on the completely positive maps that preserve units. These form a category $\mathbf{Cstar}_{\mathrm{CPU}}$ and the products and $\mathbf{Bij}$-action extend from $\mathbf{Cstar}$ (*-homomorphisms) to $\mathbf{Cstar}_{\mathrm{CPU}}$.

Interpretation. The operation new is interpreted using the following map, $\mathsf{new} : M_2 \to M_1$.

$$\mathsf{new}\begin{pmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{pmatrix} = \alpha_{11}.$$

This is not a *-homomorphism, but it is completely positive and unital.

Proposition 10. *In the category of C*-algebras and completely positive unital maps, the complex numbers $\mathbb{C}$ form a model of the theory of quantum computation.*

Theorem 11. *The complex numbers form a fully complete model:*

1. *For any linear map $f : M_{2^{m_1}} \oplus \cdots \oplus M_{2^{m_k}} \to M_{2^p}$ that is completely positive and unital, there is a term in the algebraic theory, $x_1 : m_1 \dots x_k : m_k \mid a_1 \dots a_p \vdash t$, such that $f = [\![t]\!]$.*
2. *If $\Gamma \mid \Delta \vdash t, u$ and $[\![t]\!] = [\![u]\!]$ then $\Gamma \mid \Delta \vdash t = u$ is derivable.*

(NB. Part (1) of the theorem is essentially Thm. 6.14 of [42].)

A proof is in the Appendix. In brief, we use Stinespring's theorem (e.g. [34]) to factor a completely positive unital map into a *-homomorphism followed by a restriction. This corresponds to the way that every term can be written with all the new's at the front, then the apply's, and finally the measure's. To obtain full completeness we use the minimal dilation that Stinespring's theorem yields. However, there is a complication: in this model we can only work with C*-algebras of the form $\bigoplus_{i=1}^{k} M_{2^{m_i}}$, and the minimal dilation need not be of this form. For instance, consider example (6): its minimal dilation is a *-homomorphism $M_2 \oplus M_1 \oplus M_1 \to M_3$. We resolve this by generalizing our derivation of (6) from the axioms.

5. A quantum programming language

The syntax of our equational theory describes quantum computation but it is not immediately amenable to practical programming because it focuses on continuing computations rather than intermediate results.

There is a standard way of moving between an equational theory like ours and a syntax more oriented towards programming. This applies to many different notions of computation [37]. In the setting of quantum computation, we can illustrate it by suggesting that a programmer might find it helpful to have a typed function $\underline{\mathsf{measure}} : \mathsf{qubit} \to \mathsf{bit}$ which returns the result of a standard basis measurement. Our measurement operation can then be derived: $\mathsf{measure}(a, x, y) = \mathsf{if}\ \underline{\mathsf{measure}}(a) = 0\ \mathsf{then}\ x\ \mathsf{else}\ y$. Conversely, if we have a language with a bit type that is a model of our equational theory, we can derive $\underline{\mathsf{measure}}(a) = \mathsf{measure}(a, 0, 1)$.

This relationship between 'algebraic operations' (like measure) and 'generic effects' (underlined, like measure) is a crucial one in the theory of computational effects [37]. It allows us to pass from our algebraic syntax to a language rather like Selinger's QPL [42]. A categorical way to understand this relationship is in terms of the duality between Lawvere theories, which are an abstract description of algebraic theories, and Freyd categories, which are an abstract description of first order programming languages [49].

Program equations. To be more precise, we briefly demonstrate the programming language that corresponds automatically to our algebraic theory. The types are built from the grammar:

$$A, B \ ::= \ \mathsf{qubit} \mid I \mid A \otimes B \mid 0 \mid A + B.$$

We write bool for $I + I$. A context in this language is just an assignment of types to variables. The typed terms are built from the standard rules for a linear type theory (e.g. [8]), e.g.

$$\frac{-}{x : A \vdash x : A} \qquad \frac{\Gamma \vdash t : A \otimes B \quad \Delta, x : A, y : B \vdash u : C}{\Gamma, \Delta \vdash \mathsf{let}\ (x, y) = t\ \mathsf{in}\ u : C}$$

To this standard linear type theory we add the following generic effects:

$$\frac{-}{\vdash \underline{\mathsf{new}}() : \mathsf{qubit}} \qquad \frac{\Gamma \vdash t : \mathsf{qubit}^{\otimes n}}{\Gamma \vdash \underline{\mathsf{apply}}_{\mathtt{U}}(t) : \mathsf{qubit}^{\otimes n}}$$

$$\frac{\Gamma \vdash t : \mathsf{qubit}}{\Gamma \vdash \underline{\mathsf{measure}}(t) : \mathsf{bool}}$$

The resulting language is essentially Selinger's QPL [42] (see also [3, 19, 33, 53, 55]).

Our axioms (§3.3) between algebraic expressions have counterparts as program equations, e.g. our five key axioms can be written:

$$\underline{\mathsf{measure}}(\underline{\mathsf{apply}}_{\mathtt{X}}(a)) = \neg(\underline{\mathsf{measure}}(a)) \qquad (A)$$

$$\mathsf{let}\ (a', x') = \underline{\mathsf{apply}}_{D(\mathtt{U},\mathtt{V})}(a, x)\ \mathsf{in}\ (\underline{\mathsf{measure}}(a'), x') \qquad (B)$$
$$= \mathsf{if}\ \underline{\mathsf{measure}}(a){=}0\ \mathsf{then}\ (0, \underline{\mathsf{apply}}_{\mathtt{U}}(x))\ \mathsf{else}\ (1, \underline{\mathsf{apply}}_{\mathtt{V}}(x))$$

$$\underline{\mathsf{discard}}(\underline{\mathsf{apply}}_{\mathtt{U}}(x)) = \underline{\mathsf{discard}}(x) \qquad (C)$$

$$\underline{\mathsf{measure}}(\underline{\mathsf{new}}()) = 0 \qquad (D)$$

$$\underline{\mathsf{apply}}_{D(\mathtt{U},\mathtt{V})}(\underline{\mathsf{new}}(), x) = (\underline{\mathsf{new}}(), \underline{\mathsf{apply}}_{\mathtt{U}}(x)) \qquad (E)$$

The commutativity equations amount to the commutativity of let (e.g. [50]). We combine our axioms with the standard equational theory of a linearly typed language (e.g. [8]). In this way we build a language in which every context-type pair is a model of the theory in Section 3. For instance, if $\Gamma, a : \mathsf{qubit} \vdash t : A$ then let

$$\mathsf{new}_{\Gamma,A}(a.t) \stackrel{\text{def}}{=} t[\underline{\mathsf{new}}()/a];$$

if $\Gamma \vdash t : A$ and $\Gamma \vdash u : A$ then let

$$\mathsf{measure}_{\Gamma,A}(a, t, u) \stackrel{\text{def}}{=} \mathsf{if}\ \underline{\mathsf{measure}}(a){=}0\ \mathsf{then}\ t\ \mathsf{else}\ u.$$

Higher order computation and monads. By understanding quantum computation as an algebraic effect, we are able to begin applying other techniques developed for algebraic effects in general, such as compiler optimizations and static analyses [24] and normalization by evaluation [4]. Another general method is building models of higher order computation with effects [38] by using monads.

Indeed, the equational theory of quantum computation is not a theory in the sense of classical universal algebra, but rather a theory enriched in the functor category $[\mathbf{Bij}, \mathbf{Set}]$, which is why we used actions of $\mathbf{Bij}$ to discuss models.

To be precise: it is a general fact that to give an algebraic theory with linear parameters in the style of Section 2 is to give a sifted-colimit-preserving strong monad on the symmetric monoidal closed functor category $[\mathbf{Bij} \to \mathbf{Set}]$. This follows from our syntactic completeness result (Prop. 6; see also [45, Cor. 1], [47, §VII]). For each computation context $\Gamma = (x_1 : m_1 \ldots x_k : m_k)$, we define a functor $F_\Gamma : \mathbf{Bij} \to \mathbf{Set}$ by $F_\Gamma = \mathbf{Bij}(m_1, -) + \cdots + \mathbf{Bij}(m_k, -)$. Every functor $\mathbf{Bij} \to \mathbf{Set}$ is a sifted colimit of functors of this form, so to define a sifted-colimit-preserving monad T on the functor category $(\mathbf{Bij} \to \mathbf{Set})$ it suffices to specify its action on functors of the form F_Γ. Let

$$T(F_\Gamma)p \ = \ \{\Gamma \mid a_1 \ldots a_p \vdash t\}/_{=}$$
$$\cong \ \mathbf{Cstar}_{\mathrm{CPU}}(M_{2^{m_1}} \oplus \cdots \oplus M_{2^{m_k}}, M_{2^p}).$$

In the proof of Proposition 6, $T(F_\Gamma)$ was called T_Γ. (See also e.g. [25] and [5, 9, 29].)

Since our algebraic theory (§3) is commutative, the Kleisli category of the monad T is monoidal [26] (see also [50]). This makes a connection with other work on monoidal categories for quantum computation. The monad-based model seems to be closely related to the one proposed by [27]; it would be interesting to compare it with the model of [33]. One can also study the full subcategory of the Kleisli category that is spanned by objects of the form F_Γ. This is essentially the syntactic category of our programming language, and its dual can be thought of as the Lawvere theory for our algebraic theory [49]. It is sometimes called a Freyd category, which also makes a connection with [53], since Freyd categories are a categorical formulation of arrows.

6. Variations on the algebraic theory

A distinct advantage of specifying notions of computation by algebraic theories is that it is very easy to combine different theories and to investigate the consequences of further operations and equations.

6.1 Characterizing different linear maps

Non-returning computations, sub-unital maps and recursion. We can model computations whose results are partially undefined (e.g. they 'fail') by adding to our theory a constant symbol $\mathsf{undef} : (0 \mid -)$. This should commute with the other operations:

$$- \mid a \vdash \mathsf{discard}(a, \mathsf{undef}) = \mathsf{undef}.$$

We interpret undef as the zero map $\mathsf{undef} : M_0 \to M_1$ between C*-algebras: $\mathsf{undef}() = 0$. The correspondence with C*-algebras extends to 'undef' when we relax the preservation of units to the requirement that $(1 - f(1))$ is positive: these maps are called *subunital*.

Proposition 12. *In the category of C*-algebras and completely positive subunital maps, the complex numbers form a fully complete model of the algebraic theory of quantum computation extended with* undef.

(In brief, this is because to give a subunital map $f : X \to Y$ is to give a unital map $f : X \oplus \mathbb{C} \to Y$.) Selinger has noted that the sub-unital maps between finite-dimensional C*-algebras form a pointed dcpo, and proposed to use this to interpret recursion in a quantum programming language [42]; this ought to be related to standard algebraic ways of analyzing iteration [10].

Non-determinism and non-unital maps. Some authors drop the requirement of preserving the unit altogether (e.g. [21]). I am not aware of any attempt to justify this with physical intuitions, but we can consider the idea in this algebraic framework. We do this by adding to the algebraic theory a commutative monoid, that is, a binary operation $\mathbin{⦶} : (0 \mid 0, 0)$ satisfying the monoid laws:

$$(x \mathbin{⦶} y) \mathbin{⦶} z = x \mathbin{⦶} (y \mathbin{⦶} z) \quad x \mathbin{⦶} y = y \mathbin{⦶} x \quad \mathsf{undef} \mathbin{⦶} x = x$$

Let us add axioms imposing that $⦶$ commutes with the other operations ($\mathsf{measure}$, apply and new). The operation $⦶$ is reminiscent of non-deterministic computation, although we do *not* impose idempotence, $x \mathbin{⦶} x = x$; in fact, there is no non-degenerate commutative algebraic theory that combines idempotent nondeterminism with probabilistic choice.

We can interpret $⦶$ as a linear map between C*-algebras: let $⦶ : M_1 \oplus M_1 \to M_1$ be given by $⦶(\alpha, \beta) = \alpha + \beta$. This is completely positive, but it does not preserve units.

Proposition 13. *In the category of C*-algebras and all completely positive maps, the complex numbers form a fully complete model of the algebraic theory of quantum computation extended with* undef *and* $⦶$.

To show this, we define one-sided measurement operation, $\mathsf{m}_0 : (1 \mid 0)$ by $\mathsf{m}_0(a, x) = \mathsf{measure}(a, x, \mathsf{undef})$, so that

$$\mathsf{measure}(a, x, y) = \mathsf{m}_0(a, x) \mathbin{⦶} \mathsf{apply}_{\mathtt{x}}(a, a.\mathsf{m}_0(a, y)).$$

Now all terms can be rearranged into an sum of operators, as in Choi's theorem.

6.2 Relationship with classical data and QRAM.

Changing of unitaries. The algebraic presentation of quantum computation does not assume very much about the unitaries, only they that they form a groupoid with two monoidal structures. To focus on classical computation, we can cut down to the $\{0, 1\}$ valued unitary matrices. In this classical setting the following extra axiom is reasonable.

$$x(a) = \mathsf{measure}(a, \mathsf{new}_0(a.x(a)), \mathsf{new}_1(a.x(a))) \qquad \text{(M)}$$

Axiom (M) is consistent with classical computation:

Proposition 14. *The two-element set,* 2*, is a fully complete model in the category* $\mathbf{Set}^{\mathrm{op}}$*. In other words, the following data are equivalent:*

- *A term* $x_1{:}m_1 \ldots x_k{:}m_k \mid a_1 \ldots a_p \vdash t$ *where all the unitaries are valued in* $\{0, 1\}$*, modulo all the axioms including (M).*
- *A function* $2^p \to (2^{m_1} + \cdots + 2^{m_k})$ *between sets.*

Note that Axiom (M) is is inconsistent with quantum computation, since we can use it to derive $x = y$ (for all x and y):

$$\begin{aligned}
x &= \nu a.\, (x \mathbin{?_a} y) && \textit{(D)}\\
&= \nu a.\, \mathtt{Had}_a(\mathtt{Had}_a(x \mathbin{?_a} y)) && \textit{(G),(H)}\\
&= \nu a.\, \mathtt{Had}_a((\nu a.\, \mathtt{Had}_a(x ?_a y)) \mathbin{?_a} (\nu a.\, \mathtt{X}_a(\mathtt{Had}_a(x ?_a y)))) && \textit{(M)}\\
&= \nu a.\, \mathtt{Had}_a((\nu a.\, \mathtt{Had}_a(y ?_a x)) \mathbin{?_a} (\nu a.\, \mathtt{X}_a(\mathtt{Had}_a(y ?_a x)))) && \textit{(5)}\\
&= \nu a.\, (y \mathbin{?_a} x) && \textit{(M),(H),(G)}\\
&= y && \textit{(D)}
\end{aligned}$$

Notice that the algebraic framework allows us to make this very strong statement: we are not only saying that (M) fails in the particular model of C*-algebras, but moreover that it fails in *every* consistent model of quantum computation.

Reference cells versus their contents. A rather different approach to quantum data is to step away from the actual qubits, and instead consider pointers to memory cells that store qubits (in 'QRAM'). This approach is taken in the QIO monad [6].

We can analyze this in the context of our algebraic theory by understanding the parameter variables (a, b) not as qubits but as distinct references to memory cells. Thus new does not create a new qubit, but rather allocates a new memory cell containing a qubit initialized with $|0\rangle$. With this interpretation, $\mathsf{discard}$ is not discarding a bit, but rather discarding the name of the pointer: the memory itself might remain active.

In this context it is appropriate to omit Axiom (C). We now explain this by analogy with the classical situation.

Algebraic theories with a discard operation. The theory of quantum computation has, as a subtheory, the simple theory of discarding. The signature is $\mathsf{discard} : (1 \mid 0)$, and there is one equation: $\mathsf{discard}(a, \mathsf{discard}(b, x)) = \mathsf{discard}(b, \mathsf{discard}(a, x))$. In Prop. 6 we saw that the functor category $[\mathbf{Bij}, \mathbf{Set}]$ is a canonical category for models of algebraic theories with linear parameters. In fact, as is quite well known (e.g. [17, 28, 39]), the category of algebras for the theory of discard alone is equivalent to the functor category $[\mathbf{Inj}, \mathbf{Set}]$, where $\mathbf{Inj}$ is the category of natural numbers and injections between them. This means that every algebraic theory with linear parameters and with a discard operation induces a monad on the category $[\mathbf{Inj}, \mathbf{Set}]$. This category has long been used to model dynamic allocation [44] and separation [32].

The category $\mathbf{Inj}$ has a monoidal structure, given by addition of natural numbers, and this extends to a monoidal structure on the category $[\mathbf{Inj}, \mathbf{Set}]$. If we have an algebraic theory with linear parameters and a the discard operation commutes with all other operations, then this induces a strong monad on $[\mathbf{Inj}, \mathbf{Set}]$ for this monoidal structure. There is a well-studied strong monad on $[\mathbf{Inj}, \mathbf{Set}]$ that describes local store of classical data: it is attributed to O'Hearn and Tennant, but first appeared in [36]. We analyze that monad by considering an algebraic theory of classical local store, which is a variation on the theory of quantum computation (§3) found by restricting to $\{0, 1\}$-valued matrices; omitting Axiom (C); and including Axiom (M).

Proposition 15. *The (enriched) category of algebras for the theory of classical local store is equivalent to the category of LS-algebras from [36], over* $[\mathbf{Inj}, \mathbf{Set}]$.

(This is proved in much the same way as [28, 36, 39], only within the formal framework of algebraic theories with linear parameters.) An important property of the monad for local store of classical data is that it is enriched in $[\mathbf{Inj}, \mathbf{Set}]$ in two ways: firstly with the monoidal structure, which gives a notion of separation; and secondly with the cartesian product, which enables standard programming with ordinary cartesian products.

If we add Axiom (C) then we retain the enrichment in the monoidal structure (see also [48, §4.2.2]) but the cartesian enrichment is broken. Indeed, no further axioms can be added without breaking the cartesian enrichment [46].

Following Proposition 15, we can also consider an algebraic theory of quantum local store, by omitting Axiom (C) (except for $n = 0$); omitting Axiom (M), but using all the unitaries. This again supports the two different kinds of enrichment in $[\mathbf{Inj}, \mathbf{Set}]$. We propose our algebraic theory of quantum local store as a semantic theory of the notion of computation describing the QRAM model.

Summary.	*Axioms omitted*	*Unitaries valued in*
Quantum computation	(M)	Complex numbers
Classical computation	None	$\{0, 1\}$
Classical local store	(C)	$\{0, 1\}$
QRAM	(C),(M)	Complex numbers

It would be interesting to add other operations to the signature, such as communication primitives, e.g. following [47].

Overall summary. We have presented a framework for algebraic theories with linear parameters (§2) and used it to axiomatize quantum computing (§3). We showed that our axiomatization completely describes quantum computing, by referring to an old model built from operator algebra (§4). We showed how to extend the notion of quantum computing to a programming language (§5), and considered several variations on the theory (§6), demonstrating the flexibility of the algebraic framework.

Acknowledgements: This research was supported by ERC Project QCLS. It wouldn't have been possible without the QCLS group in Nijmegen: Robin Adams, Kenta Cho, Robert Furber, Bart Jacobs, Mathys Rennela, Sander Uijlen, Bas Westerbaan, and Bram Westerbaan.

I presented this work at the Workshop on Quantum Physics and Logic (QPL, Kyoto 2014) and it was included in the informal proceedings of that workshop. I am grateful to the anonymous reviewers and the organizers of QPL 2014, and to the reviewers of POPL 2014, for their valuable comments. It has been helpful to discuss this work with many other people too, including Samson Abramsky, Rasmus Møgelberg, Gordon Plotkin, Jamie Vicary, and Mingsheng Ying.

References

[1] S. Abramsky and B. Coecke. A categorical semantics of quantum protocols. In *Proc. LICS 2004*, 2004.

[2] J. Aczél. On mean values. *Bull. Am. Math. Soc.*, 54:392–400, 1948.

[3] R. Adams. QPEL: Quantum program and effect language. In *Proc. QPL 2014*, 2014. To appear.

[4] D. Ahman and S. Staton. Normalization by evaluation and algebraic effects. In *Proc. MFPS XXIX*, pages 51–69, 2013.

[5] T. Altenkirch, J. Chapman, and T. Uustalu. Monads need not be endofunctors. In *FOSSACS 2010*.

[6] T. Altenkirch and A. Green. The Quantum IO Monad. In *Semantic Techniques in Quantum Computation*. CUP, 2009.

[7] J. Baez and J. Dolan. From finite sets to Feynman diagrams. In *Mathematics Unlimited — 2001 and Beyond*. Springer, 2001.

[8] N. Benton and P. Wadler. Linear logic, monads and the lambda calculus. In *LICS 1996*.

[9] C. Berger, P.-A. Melliès, and M. Weber. Monads with arities and their associated theories. *J. Pure Appl. Algebra*, 216, 2012.

[10] S. L. Bloom and Z. Esik. *Iteration Theories: The Equational Logic of Iterative Processes*. Springer, 1993.

[11] I. Cervesato and F. Pfenning. A Linear Logical Framework. *Inform. Comput.*, 179(1):19–75, 2002.

[12] G. Chiribella, G. M. D'Ariano, and P. Perinotti. Informational derivation of quantum theory. *Phys. Rev. A*, 84, 2011.

[13] K. Cho. Semantics for a quantum programming language by operator algebras. In *Proc. QPL 2014*, 2014. To appear.

[14] B. Coecke and R. Duncan. Interacting quantum observables. In *Proc. ICALP 2008*, pages 298–310, 2008.

[15] V. Danos, E. Kashefi, and P. Panangaden. The measurement calculus. *J. ACM*, 54(2), 2007.

[16] P. A. Fillmore. *A User's Guide to Operator Algebras*. Wiley-Interscience, 1996.

[17] M. Fiore. Notes on combinatorial functors. Draft, 2001.

[18] M. P. Fiore and C.-K. Hur. Second-order equational logic. In *CSL'10*.

[19] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron. Quipper: a scalable quantum programming language. In *PLDI 2013*.

[20] L. Hardy. Reformulating and reconstructing quantum theory. arXiv:1104.2066, 2011.

[21] C. Heunen, A. Kissinger, and P. Selinger. Completely positive projections and biproducts. arXiv:1308.4557.

[22] B. Jacobs. On block structures in quantum computation. In *Proc. MFPS XXIX*, 2013.

[23] G. Janelidze and G. Kelly. A note on actions of a monoidal category. *Theory Appl. Categ.*, 9(4), 2001.

[24] O. Kammar and G. D. Plotkin. Algebraic foundations for effect-dependent optimisations. In *POPL 2012*, pages 349–360, 2012.

[25] G. M. Kelly and A. J. Power. Adjunctions whose counits are coequalisers. *J. Pure Appl. Algebra*, 89:163–179, 1993.

[26] F. E. J. Linton. Autonomous equational categories. *J. Math. Mech.*, 15:637–642, 1966.

[27] O. Malherbe, P. J. Scott, and P. Selinger. Presheaf models of quantum computation: an outline. In *Computation, Logic, Games, and Quantum Foundations*. Springer, 2013.

[28] P.-A. Melliès. Local stores in string diagrams. In *RTA-TLCA 2014*.

[29] P.-A. Melliès. Segal condition meets computational effects. In *Proc. LICS 2010*, pages 150–159, 2010.

[30] R. E. Møgelberg and S. Staton. Linear usage of state. *Logical Methods Comput. Sci.*, 10(1), 2014.

[31] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. CUP, 2011.

[32] P. W. O'Hearn. On bunched typing. *J. Funct. Program.*, 2003.

[33] M. Pagani, P. Selinger, and B. Valiron. Applying quantitative semantics to higher-order quantum computing. In *Proc. POPL 2014*.

[34] V. Paulsen. *Completely Bounded Maps and Operator Algebras*. CUP, 2003.

[35] G. Plotkin. Some varieties of equational logic. In *Algebra, meaning and computation*. Springer, 2006.

[36] G. D. Plotkin and J. Power. Notions of computation determine monads. In *Proc. FOSSACS'02*, 2002.

[37] G. D. Plotkin and J. Power. Algebraic operations and generic effects. *Applied Categorical Structures*, 11(1):69–94, 2003.

[38] J. Power. Generic models for computational effects. *Theor. Comput. Sci.*, 364(2):254–269, 2006.

[39] J. Power. Indexed Lawvere theories for local state. In *Models, logics and higher-dimensional categories: a tribute to the work of Mihály Makkai*, pages 213–230. American Mathematical Society, 2011.

[40] J. Power and M. Tanaka. Binding signatures for generic contexts. In *Proc. TLCA 2005*, volume 308–323, 2005.

[41] M. Rennela. Towards a quantum domain theory: Order-enrichment and fixpoints in W*-algebras. In *Proc. MFPS XXX*, 2014.

[42] P. Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004.

[43] P. Selinger. Generators and relations for n-qubit Clifford operators. arXiv:1310.6813, 2013.

[44] I. Stark. Categorical models for local names. *LISP and Symb. Comp.*, 9(1):77–107, 1996.

[45] S. Staton. An algebraic presentation of predicate logic. In *FOSSACS 2013*.

[46] S. Staton. Completeness for algebraic theories of local state. In *FOSSACS 2010*.

[47] S. Staton. Instances of computational effects. In *LICS 2013*.

[48] S. Staton. Two cotensors in one. In *Proc. MFPS XXV*, 2009.

[49] S. Staton. Freyd categories are enriched lawvere theories. In *Proc. Workshop on Algebra, Coalgebra and Topology*, volume 303 of *ENTCS*, pages 197–206, 2013.

[50] S. Staton and P. B. Levy. Universal properties for impure programming languages. In *POPL 2013*, 2013.

[51] A. van Tonder. A lambda calculus for quantum computation. *SIAM J. Comput.*, 33(5):1109–1135, 2004.

[52] J. Vicary. Higher semantics of quantum protocols. In *LICS 2012*.

[53] J. K. Vizzotto, G. R. Librelotto, and A. Sabry. Reasoning about general quantum programs over mixed states. In *SBMF 2009*, 2009.

[54] M. Ying and Y. Feng. An algebraic language for distributed quantum computing. *IEEE Trans. Computers*, 58(6):728–743, 2009.

[55] M. Ying, N. Yu, and Y. Feng. Alternation in quantum programming: from superposition of data to superposition of programs. arXiv:1402.5172, 2014.

A. Proof of full completeness

We use this appendix to give more details of our proof of the completeness theorems from Section 4. As a first step we consider the case without new.

Theorem 9.

1. *For any *-homomorphism $f : M_{2^{m_1}} \oplus \cdots \oplus M_{2^{m_k}} \to M_{2^p}$ there is a term in the algebraic theory not involving* new*, $x_1 : m_1 \dots x_k : m_k \mid a_1 \dots a_p \vdash t$, such that $f = [\![t]\!]$.*
2. *If $\Gamma \mid \Delta \vdash t, u$ and t and u do not contain* new*, and $[\![t]\!] = [\![u]\!]$ then $\Gamma \mid \Delta \vdash t = u$ is derivable.*

Proof notes. Recall that if $m_1 \dots m_k, p$ are non-zero natural numbers then the set of Bratteli diagrams, $\mathbf{Brat}(m_1 \dots m_k, p)$, comprises k-tuples of natural numbers $s_1 \dots s_k$ ('partial multiplicities') such that $\sum_{i=1}^k s_i m_i = p$. There is a function

$$\mu : \mathbf{Brat}(m_1 \dots m_k, p) \to \mathbf{Cstar}(M_{m_1} \oplus \cdots \oplus M_{m_k}, M_p)$$

where $\mu(s_1 \dots s_k)(A_1, \dots, A_k)$ is the block diagonal matrix formed by s_1 copies of A_1 followed by s_2 copies of A_2 and so on.

The following result is standard (e.g. [16, 1.1.3]).

Proposition 16. *The function μ has a retraction: there is a function $\rho : \mathbf{Cstar}(M_{m_1} \oplus \cdots \oplus M_{m_k}, M_p) \to \mathbf{Brat}(m_1 \dots m_k, p)$ such that $\rho\mu = \mathrm{id}$. Moreover $\rho(f) = \rho(g)$ if and only if there is a $p \times p$ unitary U such that $f(\vec{A}) = U^*(g(\vec{A}))U$.* □

Consider a context $(x_1 : m_1 \dots x_k : m_k \mid a_1 \dots a_p)$. Suppose that the second order variables are arranged in order of increasing valence. The following are in bijective correspondence:

1. Terms built only from measure and variables, such that the variables (of both kinds) appear in the same order as in the context. (These terms are not considered modulo any equations.)
2. Sequences of partial multiplicities in $\mathbf{Brat}(2^{m_1} \dots 2^{m_k}, 2^p)$.

To show this, we interpret a term t as a Bratteli diagram $(\!|t|\!)$ so that $\mu(\!|t|\!) = [\![t]\!]$. If the term is a variable $x_i(a_1 \dots a_q)$, then $m_i = q = p$, and we let $(\!|x_i(a_1 \dots a_q)|\!) \in \mathbf{Brat}(2^{m_1} \dots 2^{m_k}, 2^p)$ be given by $s_i = 1$, and $s_j = 0$ for $j \neq i$. If the term is a measurement, $\mathsf{measure}(a_{p+1}, t, u)$, then by induction t and u correspond to Bratteli diagrams $(\!|t|\!), (\!|u|\!) \in \mathbf{Brat}(2^{m_1} \dots 2^{m_k}, 2^p)$, and so $\sum_{i=1}^k (\!|t|\!)_i 2^{m_i} + \sum_{i=1}^k (\!|u|\!)_i 2^{m_i} = 2^{p+1}$. Let $(\!|\mathsf{measure}(a_{p+1}, u, v)|\!)$ in $\mathbf{Brat}(2^{m_1} \dots 2^{m_k}, 2^{p+1})$ be the coordinatewise sum of $(\!|t|\!)$ and $(\!|u|\!)$, i.e. $(\!|\mathsf{measure}(a_{p+1}, u, v)|\!)_i = (\!|t|\!)_i + (\!|u|\!)_i$.

The inverse to this interpretation maps Bratteli diagrams $\vec{s}$ in $\mathbf{Brat}(2^{m_1} \dots 2^{m_k}, 2^p)$ to terms $\mathrm{Measure}(\vec{s})$ of this special form, so that $\mathrm{Measure}((\!|t|\!)) = t$ and $(\!|\mathrm{Measure}(\vec{s})|\!) = \vec{s}$. This inverse is defined by induction on p. The special order on the computation variables is needed here.

We can thus define a section of the semantic map

$$[\![-]\!] : \{t \mid (x_1 : m_1 \dots x_k : m_k \mid a_1 \dots a_p \vdash t)\} \to \mathbf{Cstar}(M_{2_1^m} \oplus M_{2^{m_k}}, M_{2^p})$$

as follows. Every *-homomorphism $f : M_{2^{m_1}} \oplus \cdots \oplus M_{2^{m_k}} \to M_{2^p}$ factors as $f(x) = \mathtt{U}^* \cdot (\mu\,\vec{s})(x) \cdot \mathtt{U}$, for a $2^p \times 2^p$ unitary $\mathtt{U}$, so that $f = [\![\mathsf{apply}_{\mathtt{U}}(\vec{a}, \vec{a}.\mathrm{Measure}(\vec{s}))]\!]$. Thus the first part of the Theorem 9 is proved.

We show the second part of Theorem 9 in two stages.

1. We show that the section of the semantic map doesn't depend on the choice of $\mathtt{U}$;
2. We show that the section is a surjection, i.e. that every term is equal to one of the form $\mathsf{apply}_{\mathtt{U}}(\vec{a}, \vec{a}.\mathrm{Measure}(\vec{s}))$.

First, we show that for any Bratteli diagram $\vec{s} \in \mathbf{Brat}(2^{m_1} \dots 2^{m_k}, 2^p)$ and any $2^p \times 2^p$ unitaries $\mathtt{U}, \mathtt{V}$, if

$$\mathtt{U}^* \cdot (\mu\,\vec{s})(-) \cdot \mathtt{U} = \mathtt{V}^* \cdot (\mu\,\vec{s})(-) \cdot \mathtt{V} \tag{7}$$

as linear maps, then we can derive the equality

$$\mathsf{apply}_{\mathtt{U}}(\vec{a}, \vec{a}.\mathrm{Measure}(\vec{s})) = \mathsf{apply}_{\mathtt{V}}(\vec{a}, \vec{a}.\mathrm{Measure}(\vec{s})) \tag{8}$$

between terms. This fact can be understood along the same lines as the freedom in the operator-sum representation of completely positive maps (e.g. [31, Ch. 8]). If $(\mu\,\vec{s})(-) = \mathtt{U}^* \cdot (\mu\,\vec{s})(-) \cdot \mathtt{U}$ then $\mathtt{U}$ must be built from a tensor products of unitaries that are conditional on qubits that are being measured and that act on qubits that are being discarded. All this is taken care of by laws (A)–(C), (F)–(J). Thus the section of the semantic map does not depend on the choice of $\mathtt{U}$.

It remains for us to show that the section is a surjection, i.e. that every term is equivalent to one of the form $\mathsf{apply}_{\mathtt{U}}(\vec{a}, \vec{a}.\mathrm{Measure}(\vec{s}))$. In brief, we use the equations to rearrange a term as follows: first we push all measure operations inside apply operations using law (B); next we arrange all the second order variables to appear in the designated order by using controlled nots (possibly controlled-controlled-nots etc), via laws (A), (B) and (J); we arrange all the parameter variables to appear in the designated order, by using controlled swaps, via laws (B), (F); a sequence of apply operations can be combined into one apply operation, using tensors and composition, via laws (G)–(I). This concludes our proof of Theorem 9. □

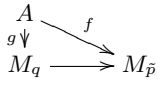
We now turn to prove Theorem 11, extending Theorem 9 to consider new and completely positive maps.

Theorem 11.

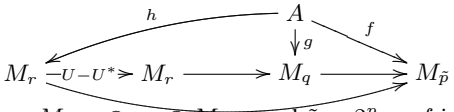
1. *For any linear map $f : M_{2^{m_1}} \oplus \cdots \oplus M_{2^{m_k}} \to M_{2^p}$ that is completely positive and unital, there is a term in the algebraic theory, $x_1 : m_1 \ldots x_k : m_k \mid a_1 \ldots a_p \vdash t$, such that $f = [\![t]\!]$.*
2. *If $\Gamma \mid \Delta \vdash t, u$ and $[\![t]\!] = [\![u]\!]$ then $\Gamma \mid \Delta \vdash t = u$ is derivable.*

(NB. Part (1) of the theorem is a variation on Selinger's [42, Thm. 6.14]. It is the complete axiomatization of equality that is new.)

Proof notes. Recall the following variant of Stinespring's theorem (e.g. [34, p. 45]). Let $\tilde{p}$ be a natural number. If $f : A \to M_{\tilde{p}}$ is completely positive and unital then there is a natural number $q \geq \tilde{p}$ and a *-homomorphism g making the following diagram commute:



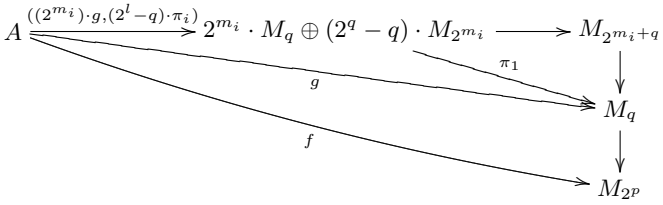
where the unlabelled horizontal arrow is $A \mapsto A|_{\tilde{p}}$, where $A|_{\tilde{p}}$ comprises the first $\tilde{p}$ rows and columns of the $q \times q$ matrix A. Moreover q can be chosen as the minimal such number: if $r \geq \tilde{p}$ and a *-homomorphism $h : A \to M_r$ is such that $h(-)|_{\tilde{p}} = f(-)$ then $r \geq q$ and there is an r-ary unitary U such that $g(a) = (U(h\,a)U^*)|_q$. As a diagram:



Now if $A = M_{2^{m_1}} \oplus \cdots \oplus M_{2^{m_k}}$ and $\tilde{p} = 2^p$, so f is as in the statement of the theorem, we can use the minimal dilation to show that f is definable, provided q is a power of 2. This is because the horizontal arrow can be defined using new and the *-homomorphism can be defined according to Theorem 9. Moreover the minimality essentially gives us the second part of the theorem.

However, q need not be a power of 2. For instance, consider example (6): its minimal dilation is a *-homomorphism $M_2 \oplus M_1 \oplus M_1 \to M_3$.

We resolve this as follows. First we note that A is not the trivial C*-algebra M_0, for if $A \cong M_0$ then, since f is unital, $2^p = 0$, which is absurd. So $k \neq 0$ and we pick $i \leq k$, giving us a *-homomorphism $\pi_i : A \to M_{2^{m_i}}$. Now the Stinespring dilation gives the following factorization of f:

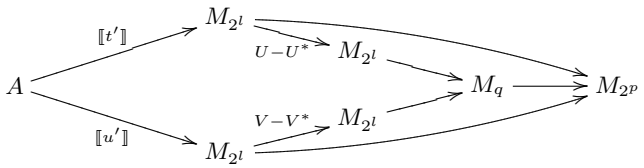


Note that the horizontal maps are *-homomorphisms, and the vertical maps are restrictions. Thus the completely positive map f is definable: there is a term t such that $f = [\![t]\!]$.

Now suppose that t and u are terms with the same interpretation ($[\![t]\!] = [\![u]\!]$). We will show that their equality can be derived. We can assume that the terms are written with the news on the outside, by using the commutativity equations, so that $t = \mathsf{new}(a_1 \ldots a_l.t')$ where t' doesn't contain new. Axiom (D) implies $\mathsf{new}(a.\mathsf{discard}(a, x)) = x$, so we can add new's to t or u so that they both have the same number of new's (l, say). In summary we have *-homomorphisms $[\![t']\!] : A \to M_{2^l}$ and $[\![u']\!] = A \to M_{2^l}$, such that

$$[\![t]\!] = A \xrightarrow{[\![t']\!]} M_{2^l} \to M_{2^p} \quad \text{and} \quad [\![u]\!] = A \xrightarrow{[\![u']\!]} M_{2^l} \to M_{2^p}.$$

By Stinespring's theorem there is a natural number q so that M_q is the minimal dilation of $[\![t]\!] = [\![u]\!]$; by minimality there are $2^l \times 2^l$ unitaries U and V such that



where the composites of the inner square are *-homomorphisms. Since U and V do not affect M_{2^p}, we have

$$\mathsf{new}(\vec{a}.t) = \mathsf{new}(\vec{a}.\mathsf{apply}_U(\vec{b}, \vec{b}.t'))$$

and

$$\mathsf{new}(\vec{a}.u) = \mathsf{new}(\vec{a}.\mathsf{apply}_V(\vec{b}, \vec{b}.u')).$$

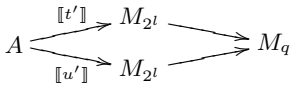
This follows from the following lemma.

If there are n parameter variables in context and U is a unitary of arity 2^n then we write $\mathsf{apply}_U(t)$ as shorthand for $\mathsf{apply}(\vec{a}^n, \vec{a}.t)$.

Lemma 17. *Let $m < n$ and let U be a unitary on 2^n square matrices that fixes the top 2^m rows. Then the following equation is derivable:*

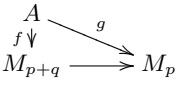
$$x : n \mid a_1 \ldots a_m \vdash \mathsf{new}(a_{m+1} \ldots a_n.\mathsf{apply}_U(x(\vec{a}))) = \mathsf{new}(a_{m+1} \ldots a_n.x(\vec{a}))$$

This lemma follows from Axiom (E). So we can assume wlog that U and V are identities, and



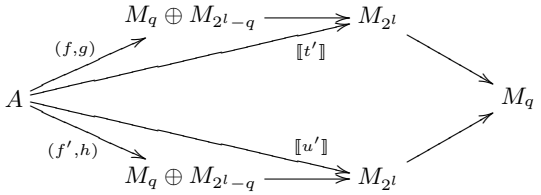
We now use another lemma:

Lemma 18. *If the following diagram commutes*



*and if f and g are *-homomorphisms then f factors through the block diagonal map $M_p \oplus M_q \to M_{p+q}$ (measurement).*

Returning to our main argument, the lemma (18) gives us f, f', g, h and the following situation:



We immediately have $f = f'$. Now,

$$t = \mathsf{new}(\vec{a}.\,\mathsf{new}(b.\mathsf{measure}(b, t', u'))) = \mathsf{new}(\vec{a}.\,\mathsf{new}(b.\mathsf{apply}_U(\mathsf{measure}(b, t', u'))))$$

where U is the unitary that swaps the two occurrences of $(2^l - q)$ in $M_{q+(2^l-q)+q+(2^l-q)} = M_{2^l} \otimes M_2$. This equation is a consequence of Lemma 17: U fixes the upper 2^p rows.

It remains for us to show that $\mathsf{apply}_U(\mathsf{measure}(b, t', u')) = \mathsf{measure}(b, u', t')$. These terms don't involve new, so by Theorem 9 the equality is derivable if and only if the corresponding

*-homomorphisms are equal. Indeed they are, because

$$\begin{aligned}
&[\![\mathsf{apply}_U(\mathsf{measure}(b,t',u'))]\!](a) \;=\; U\begin{pmatrix} [\![t']\!](x) & 0 \\ 0 & [\![u']\!](x) \end{pmatrix} U^* \\
&= U\begin{pmatrix} f(x) & 0 & 0 & 0 \\ 0 & g(x) & 0 & 0 \\ 0 & 0 & f(x) & 0 \\ 0 & 0 & 0 & h(x) \end{pmatrix} U^* \;=\; \begin{pmatrix} f(x) & 0 & 0 & 0 \\ 0 & h(x) & 0 & 0 \\ 0 & 0 & f(x) & 0 \\ 0 & 0 & 0 & g(x) \end{pmatrix} \\
&= \begin{pmatrix} [\![u']\!](x) & 0 \\ 0 & [\![t']\!](x) \end{pmatrix} \;=\; [\![\mathsf{measure}(b,u',t')]\!](x)
\end{aligned}$$

This concludes our proof of Theorem 11. □