
Can a quantum computer run the von Neumann architecture ?

Peter Hines

Dept. of Computer Science,
University of York,
York. UK

Summary. *At the core of nearly every modern computer is a central processing unit running the von Neumann architecture. This computer architecture gives computationally universal machines, and non-trivial control structures arise naturally, leading to high-level programming constructs.*

At the core of the von Neumann architecture is the notion that program code may be stored and manipulated in the same way as data. A datum describing an operation may be stored and processed in the same way as any other form of data, but may also be ‘promoted’ to an operation, and applied.

Classically, this is well-studied — particularly from a categorical point of view. We consider such operations in the quantum setting, including Nielsen & Chuang’s orthonormal encoding, Abramsky & Coecke’s categorical foundations, the BBC protocol, and the Choi-Jamiołkowski correspondence.

Obstacles to a quantum analogue of the von Neumann architecture are also considered, including the no-cloning and no-deleting theorems, the ‘no-programming principle’, and the Gottesman-Knill theorem.

1 Introduction

For impatient readers, the answer to the question posed in the title is, ‘No’. Readers familiar with quantum information and computation may well think that the answer should be, ‘Of course not!’ — although our intention is to prove that the question is more subtle than that. A more accurate, although less concise, title for this paper would therefore be, “*Why is the von Neumann architecture so significant for classical computation, what are the differences between quantum and classical information that mean classical computers can implement the von Neumann architecture but quantum computers cannot, and what are the implications of this for models of quantum computation and information?*”

In order to answer these questions, we analyse the von Neumann architecture from a classical point of view, in order to decide what features give

it both practical utility and computational power, and then consider whether or not these essential features are shared by quantum systems.

In the classical world, the von Neumann architecture is ubiquitous. This is partly for practical reasons; binary values, logic gates, and a global clock are readily implementable via electronic circuits. However, it also bridges the gap between theoretical notions of computer science, and the underlying physical structures. Universality, programmability, compilation and higher-level control structures all arise in a natural way from the underlying architecture. All these are key concepts of computing — and are much less well-established in the quantum case.

Our claim is that the practical core of the von Neumann architecture is the interchangeability of code and data. This is a fundamental concept of theoretical computer science, with close connections to formal logic and λ -calculus. From the categorical point of view, the code/data correspondence is exactly *Cartesian closure* — a special form of categorical closure. Quantum information also admits code/data correspondences — we consider similarities and differences, and their implications for machine architectures.

We emphasise that this paper is expository in the sense that results presented are generally well-known, or at least consequences of well-known theory. The aim is to view this categorical picture through the frame of the von Neumann architecture, and to consider implications from this very practical point of view.

2 The von Neumann architecture

2.1 The origins of the vN architecture

In 1945, whilst on an extended stay at Los Alamos, J. von Neumann laid out in formal logical terms the basic operating principles of the EDVAC computer [79]. This was an (incomplete) draft report on the work of a team¹ on a project by the University of Philadelphia for the U.S. Army Ballistics Research Laboratory. Famously, this incomplete draft was widely distributed by H. Goldstine, an army mathematician who had originally introduced von Neumann to the project [20].

This distribution of an incomplete draft, listing von Neumann as sole author, later caused considerable bad feeling within the EDVAC team (see [11, 94, 72]), and it has been claimed [11] that von Neumann himself intended for a final completed version to be jointly co-authored by the entire team.

The EDVAC computer itself did not become fully operational until 1951, partly due to a dispute with the University of Philadelphia over intellectual property and patent rights resulting from this prior publication [71]. However,

¹ The team was lead by J. Mauchley and J. P. Eckert, with J. von Neumann acting as a consultant. See [11] for details of the team members.

when finally operational, the EDVAC computer was highly successful [11], and the general principles outlined in [79] have become an almost universally accepted standard for processor design, known as the von Neumann, or vN, architecture.

2.2 The fetch-execute cycle

Although technical details have changed beyond recognition (e.g. a significant advance of the EDVAC machine was the use of acoustic waves in mercury-filled tubes as a form of random-access memory [39]), the underlying principles of the von Neumann architecture were spectacularly successful, and remain in use today, in the form of the core operation of the Central Processing Unit of a computer. This *fetch-execute cycle* is a simple iterative step, performed on every clock cycle, that leads to the full range of behaviour of modern computers.

The fetch-execute step is as follows :

At the beginning of each cycle the **program counter** contains the value of a memory location.

1. The CPU *copies* the contents of the memory location referenced by program counter into the **instruction register**.
2. The data in the instruction register is *decoded* and the **control unit** *performs the action described*. This may be :
 - a) *Copy a value* from memory into the **accumulator**.
 - b) *Apply an instruction* (logic gate) to the accumulator.
 - c) *Copy the contents* of the accumulator into a **memory location**.
 - d) *Overwrite* the contents of the program counter with a **new value**.
3. The program counter is then *incremented* (in order to address the next instruction).

2.3 The utility of the von Neumann architecture

Practically, the vN architecture is significant for a number of reasons :

1. A *computationally universal* machine can be constructed. Up to memory constraints, any computation that may be performed by a Turing machine or the untyped lambda calculus may be performed on a von Neumann computer.
2. Computers may be *programmed* — the program executed is dependent on the contents of the computer memory, and no hardware reconfiguration is required in order to run a different computer program.
3. Manipulation of program code in a similar manner to data allows for branching, conditional execution, and subroutines. This opens the way to meaningful *control structures*.

4. Meaningful control structures allow high-level languages to be built on top of the basic machine code, and the equal treatment of code and data allows a von Neumann machine to run *compilers*, *interpreters* and *assemblers*.

Our interest in the vN architecture from a quantum-mechanical perspective is in order to seek analogues of 1.–4. above.

In terms of quantum computation, 1. has been intensively studied — we consider this further in Section 13.1. Also, despite the current paucity of quantum algorithms, 2. may become important at a later stage — we refer to [96] for some interesting ideas regarding stored-code quantum computers.

In terms of languages and control structures, several high-level languages for quantum computers have been proposed (see [35] for a survey). These are generally based on a ‘classical control, quantum data’ paradigm, although purely quantum (i.e. superposition-preserving — see Section 3.3) conditionals have been proposed in [9, 41], and [53] considers control structures for conditional iteration based on purely quantum control. However, a comparison of [35] with any standard QM computation text (such as [42, 75, 81]) demonstrates that there are many more quantum programming languages than quantum algorithms².

This brings us to 4. In the classical world, this feature has become so deeply ingrained into modern computing as to be almost invisible. The control structures and, to some extent, high-level languages, used in modern computation arise naturally from the underlying structure of the vN architecture. It is this feature that is of particular interest — that intuitive and useful structure arises from the underlying architecture of computer processors.

Finally, our interest in higher-level languages and control structures is in stark contrast to von Neumann’s attitude [72]. With regard to his FORTRAN language, J. Backus recalls von Neumann as begin unimpressed, asking, “Why would you want more than machine code?”. (See the title of [12] for a contrary view!) D. Gillies also recalls von Neumann’s anger at his programming of the first assembler, on the grounds that this was, ‘using a sophisticated scientific tool to perform clerical tasks that could easily be carried out by graduate students’ [72].

² This comment is deliberately unfair, in that languages presented in [35] have been designed for many purposes — including quantum communication protocols (of which there are many), proving correctness of both protocols and algorithms, formal proofs of security for quantum encryption and communication, &c. — and none of them have the creation of new algorithms as a stated objective. However, the point remains that going from quantum programming languages to quantum programs is a highly non-trivial exercise.

3 Relevant Quantum Information theory

3.1 Basic quantum information

We briefly reprise some fundamentals of quantum information and computation. No attempt is made to give a full or consistent exposition – we concentrate on the basic building blocks and properties relevant to this paper. For a full introduction, we refer to either [41, 65, 81], and [19] for comprehensive mathematical background on complex Hilbert spaces. We also use a pure state description, rather than considering mixed states, density matrices and completely positive maps — any of the previous references will also give a good exposition of this approach.

The atomic building-blocks of quantum information are *quantum bits*, or **qubits**, norm-1 vectors in a 2-dimensional complex Hilbert space Qu . Concatenation of qubits is given by the tensor product of Hilbert spaces, so n qubits are modelled by the 2^n -dimensional space $\otimes_{i=1}^n Qu$. Spaces of this form are called **quantum registers** of n qubits.

We will sometimes work in arbitrary finite-dimensional spaces, not just tensor product spaces of qubits. Norm-1 vectors in such spaces are sometimes known as *qudits*.

Operations on quantum registers are either **unitary maps**, or **measurements**. A unitary map, describing the evolution of an isolated quantum system, is simply an inner-product preserving linear isomorphism, and it is standard to talk about applying a unitary map to a quantum register. When given as matrices, unitaries are exactly those invertible matrices whose inverse is given by the complex conjugate.

A *measurement* is determined by a self-adjoint operator, or Hermitian matrix. By the spectral decomposition theorem, every finite Hermitian matrix has a unique decomposition as the sum of a complete set of projection operators, and the corresponding subspaces are taken to be the experimental outcomes of a measurement — we refer to [32] for details.

In quantum computation, as opposed to quantum mechanics generally, Hilbert spaces are equipped with a fixed orthonormal basis, known as the *computational basis*. Information-theoretically, this is a non-trivial step — the specification of a computational basis may be considered as *classical knowledge* about a quantum system.

A significant difference between quantum and classical information is the phenomenon of *entanglement*. Given Hilbert spaces H, K , a vector $\zeta \in H \otimes K$ is called **separable** when it may be written as $\phi \otimes \psi$, for some $\phi \in H, \psi \in K$, and **entangled** otherwise. Entanglement gives rise to many of the counter-intuitive and non-local effects of quantum mechanics, and is heavily studied. It is this phenomenon that is widely believed to provide a computational advantage in using quantum-mechanical rather than classical computing devices (We refer to [62] for analyses of the origins of speedup in quantum algorithms).

3.2 Dirac notation, and measurement probabilities

An exceedingly useful formalism for manipulating quantum information is *Dirac notation*. This is based on the (very categorical – see [5]) notion we may work with linear maps only — instead of referring to a state vector $\psi \in \mathcal{H}$ we consider the linear map $|\psi\rangle : \mathbb{C} \rightarrow \mathcal{H}$, defined in the natural way as $|\psi\rangle(z) = z.\psi$. These linear maps are known as *Ket vectors*, and have duals, the *Bra vectors*, which are linear maps (functionals) $\langle\phi| : \mathcal{H} \rightarrow \mathbb{C}$ defined by the condition that the composite $\langle\phi| \circ |\psi\rangle$, as a linear endomap of \mathbb{C} , is the inner product of ϕ and ψ .

The physical interpretation of this composite is one of the key points of the Hilbert space formalism for quantum mechanics. Consider a state vector ψ , and a measurement specified by the Hermitian operator ζ , and a vector ϕ which is an eigenstate of ζ . The probability of observing the state ϕ by the measurement ζ is exactly the norm square of the above inner product, so

$$\text{prob. of observing } \phi = |\langle\phi|\psi\rangle|^2$$

Strictly, $\langle\phi|\psi\rangle$ is a linear map from \mathbb{C} to itself, given by multiplication with the inner product of ϕ and ψ . However, it is standard to abuse notation and refer to the complex number $\langle\phi|\psi\rangle \in \mathbb{C}$. Similarly, we refer to the state vector $|\psi\rangle \in \mathcal{H}$, with the understanding that it is in fact a linear map.

3.3 Superpositions and Coherent operations

State vectors are norm-1 vectors in some Hilbert space H . Hence, given state vectors $|\phi\rangle$ and $|\psi\rangle$, the norm-1 vector $|\zeta\rangle = \alpha|\phi\rangle + \beta|\psi\rangle$ is also a state vector, for all $\|\alpha\|^2 + \|\beta\|^2 = 1$ — we say that $|\zeta\rangle$ is a *superposition* of $|\phi\rangle$ and $|\psi\rangle$. The phenomenon of superposition is not uniquely quantum-mechanical (e.g. it is a feature of classical wave-mechanics and linear optics, although interpretations differ — see [28] for an early, but very readable account of superposition). However, *superposition-preserving* processes form an important part of quantum computation. From [73],

“Any quantum algorithm relies on the fact that if an arbitrary input state $|\Phi_i\rangle$ evolves to the final state $|\Psi_i\rangle$ then the superposition $\sum_{i \in I} \alpha_i |\Phi_i\rangle$ evolves as $\sum_{i \in I} \alpha_i |\Phi_i\rangle \mapsto \sum_{i \in I} \alpha_i |\Psi_i\rangle$ ”.

This is also taken as the definition of ‘*fully quantum*’ in both the original specification of a quantum Turing machine [29], and subsequent criticisms of this definition [78]. We refer to superposition-preserving processes as **coherent**. Note that unitary processes are by definition coherent. However, these are not the only coherent quantum processes; see [16, 17] for coherent processes involving unitaries, measurements, and classically-conditioned operations.

3.4 No-cloning, no-deleting, and fan-out

Two important constraints on quantum information are the *no-cloning* and *no-deleting* theorems.

The no-cloning theorem is due to [98] :

Theorem 1. *Let $|\phi\rangle$ be an arbitrary state vector in some Hilbert space H , and let $|e\rangle$ be a fixed state in the same space. There does not exist a quantum process that acts as $|\phi\rangle|e\rangle \mapsto |\phi\rangle|\phi\rangle$. \square*

The no-deleting theorem is not simply the dual of the no-cloning theorem – rather, it states that an unknown quantum state cannot be deleted, *even in the presence of a copy*³. The significantly simpler statement that an unknown state cannot be deleted is known as no-erasure, and is a simple consequence of linearity.

The no-deleting theorem is due to [82] :

Theorem 2. *Let $|\psi\rangle$ be an arbitrary state vector in some Hilbert space H , let $|e\rangle$ be a fixed state vector, and let $|A\rangle$ be some ancilla. Then any quantum process that acts as $|\psi\rangle|\psi\rangle|A\rangle \mapsto |\psi\rangle|e\rangle|A_\psi\rangle$, is simply (up to local unitary operations) a swap map on the second and third subspaces. \square*

The presence of either copying or deleting operations in quantum systems would allow for superluminal (i.e. faster-than light) signalling [83]. Thus, these fundamental theorems of quantum information play an important rôle in ensuring the consistency of quantum physics with classical relativity.

The no-cloning and no-deleting theorems above state that *arbitrary* quantum states cannot be copied or deleted. However, *computational basis states* may be copied using the **fan-out** operation. Consider an n -dimensional space H with computational basis $\{|0\rangle, \dots, |n-1\rangle\}$. The (general) fan-out operation $F : H \otimes H \rightarrow H \otimes H$ is defined by its action on the computational basis states as :

$$F(|i\rangle|j\rangle) = |i\rangle|i+j \pmod n\rangle$$

In particular, $F(|k\rangle|0\rangle) = |k\rangle|k\rangle$. However, this is *not* a general copying operation; given the superposition of two basis states, $|\phi\rangle = \alpha|j\rangle + \beta|k\rangle$, then

$$F(|\phi\rangle|0\rangle) = \alpha|j\rangle|j\rangle + \beta|k\rangle|k\rangle \neq |\phi\rangle|\phi\rangle$$

3.5 Resource-sensitivity and the von Neumann architecture

Resource-sensitivity will prove a key point in understanding why a quantum computer cannot implement the von Neumann architecture. On a very simple

³ The no-deleting property is what logicians would refer to as the failure of the *contraction rule* (see [85] for an in-depth discussion of this), whereas the no-cloning property is a (special case of) failure of the *weakening rule*. These may be treated separately (i.e. we may consider logics with weakening, but not contraction, or vice versa), as in the field of *substructural logics* [86].

level, there are several explicit copying or deleting steps listed in Section 2.2, and these violate the no-cloning and no-deleting principles. A simple fix would be to replace the irreversible step

- **Copy a value** from memory location x into the accumulator.

by the reversible step

- **Swap the value** in memory location x with the contents of the accumulator.

and a similar change may be made to the

- **Overwrite** the contents of the program counter with a new value.

Replacing ‘copy’ and ‘overwrite’ by ‘swap’ is not the focus of this paper; embeddings of irreversible computation into reversible computation have been well-studied [15], including from a physical point of view [66]. The objection is more fundamental, and has to do with the way that program code may be stored and manipulated in the same way as data — and in particular, the ‘*decode the data in the instruction register, and perform the action described*’ step.

4 Data / code interchangeability, and Evaluation

Our claim is that the computational core of the von Neumann architecture is the “interpret a byte as an operation” step – and this is exactly the step that is problematic for quantum computation.

In the vN architecture, the *datum* in the arithmetic logic unit is interpreted as an *instruction* which may be applied to the contents of the data register. This is a physical implementation of the notion that an object may be ‘promoted’ to a function, and functions may be stored and manipulated in the same way as any other data object (In Section 7.1 we show how this arises from *Currying*, and, in general, the related property of *monoidal closure*).

In simple terms, the von Neumann architecture implements an *Evaluation* operation: Consider a datum P of type *Byte* that specifies some operation $\Theta : \text{Byte} \rightarrow \text{Byte}$. We call P the *name* of Θ , and write $P = \ulcorner \Theta \urcorner$. An evaluation operation then takes the byte P , and another byte Q , and returns $\Theta(Q)$. At the core of the von Neumann architecture is a *physical* process that implements such an evaluation operation. Our aim is to consider how, or whether, such an operation may be physically implemented in a quantum setting — with all the implications this has for primitive machine architectures.

We will need to consider two variants of an evaluation operation :

1. **Resource-sensitive evaluation** $(P, Q) \mapsto \Theta(Q)$.
2. **Non- resource-sensitive evaluation** $(P, Q) \mapsto (P, \Theta(Q))$.

The most familiar form of evaluation is 2. above; in 1., the program code is ‘consumed’ as the program is executed, which is certainly not a feature of classical computation. From either a logical or category-theoretic point of view, 1. is considered fundamental, and 2. arises from 1. in the presence of a primitive copying operation — we say that a map $(P, Q) \mapsto (P, \Theta(Q))$ contains an implicit copying step⁴. We thus need to bear in mind the essential resource-sensitivity of quantum information (as in Section 3.4), and its implications for evaluation operations.

4.1 Indirect addressing, and Evaluation operations

We have claimed that the core of the von Neumann architecture is the ‘Evaluation’ operation that promotes data to code. An alternative point of view is that the power of the vN architecture arises from *indirect addressing* — i.e. operations such as the, ‘*copy the contents of the memory location referenced by the program counter into the instruction register*’ step in the fetch-execute cycle. We now show that the existence of a suitable evaluation operation naturally allows for indirect addressing.

Consider a quantum computer with a suitable evaluation operation (which may, or may not, exist). Relative addressing may easily be implemented. Let us assume the hypothetical quantum computer has

- a **program counter register** \mathcal{P} ,
- an **instruction register** \mathcal{I} .
- n **memory registers**, $\mathcal{M}_1, \dots, \mathcal{M}_n$,

The complete ‘configuration space’ of this computer is thus the space

$$\mathcal{C} = \mathcal{P} \otimes \mathcal{I} \otimes \mathcal{M}_1 \otimes \dots \otimes \mathcal{M}_n$$

The first question is, given the value $|j\rangle$ in the program counter register, \mathcal{P} , can we ‘copy the contents of the memory register \mathcal{M}_j into the instruction register \mathcal{I} ’? Of course, (as per Section 3.5), we cannot copy the contents of \mathcal{M}_j — nor can we irreversibly erase the contents of \mathcal{I} . However, we can swap the contents of \mathcal{M}_j and \mathcal{I} using a unitary operation⁵. Let us call this unitary $Load_j : \mathcal{C} \rightarrow \mathcal{C}$.

⁴ This statement is, of course, more general than formal. However, it may be formalised in a wide range of settings. In a logical setting, the observation that in his semantic models implication is not primitive but contains an implicit copying operation famously motivated J.-Y. Girard’s Linear Logic [37, 38]. The connection between evaluation and logical rules is beyond the scope of this paper — we refer to [30, 6] for logical interpretations of the particular structures presented, in terms of linear logic operations.

⁵ The existence this swap is exactly the *symmetry* of the tensor product. For Hilbert spaces H and K , there is a natural isomorphism $\sigma_{H,K} : H \otimes K \rightarrow K \otimes H$. We shall also see in Section 9.1 that the existence of such a symmetry is required for a categorical treatment of evaluation — at least, in the quantum-mechanical setting.

Let us now assume the existence of a suitable evaluation operation, and assume without loss of generality, that the name of the operation $Load_j$ is exactly the state $|j\rangle = \lceil Load_j \rceil$. Applying the evaluation operation to the contents of the program counter register then provides exactly the relative addressing required.

The above discussion makes no mention of what would, or should, happen when a superposition of values is held in the program counter register, or when two registers of the quantum computer are entangled. These are questions that need to be answered in a discussion of the general properties of a quantum-mechanical evaluation operation.

We now consider, as a ‘toy example’, the simplest possible classical case, and use this to motivate discussion of a quantum-mechanical form of evaluation due to [80].

5 Evaluation in the one-bit computer

As a starting point we consider evaluation in the simplest possible case : the one-bit classical computer. The data types are single bits, $\{0, 1\}$, and there are exactly two program instructions :

$$\begin{aligned} \text{The identity map} & : Id(b) = b \\ \text{Negation} & : Not(b) = b + 1 \text{ (Mod 2)} \end{aligned}$$

We let 0 be the name of the identity map, and 1 be the name of the negation map, so the (non- resource-sensitive) evaluation map is an isomorphism that takes pairs of bits to pairs of bits $(Program, Data) \rightarrow (Program, newData)$ as shown in Figure 1.

Fig. 1. Evaluation in the one-bit computer

Before		After	
Prog.	Data	Prog.	Data
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Thus the *Eval* operation for the one-bit computer is simply the classical controlled-not logic gate. The quantum version of this logic gate is one of the

basic building blocks of the quantum circuit model. It is therefore natural to consider whether a general such evaluation operation may be implemented by unitary maps.

6 Implementing *evaluation* by unitary operations ?

We now consider whether an ‘evaluation’ operation may be implemented using unitary maps. However, as we are working in the finite-dimensional case, we are forced to consider non- resource-sensitive evaluation : Consider quantum registers \mathbf{C}, \mathbf{D} (the *code* and *data* registers). A resource-sensitive evaluation operation would have type $Eval_{rs} : C \otimes D \rightarrow D$ — but as $C \otimes D$ and D have different dimensions, no such unitary map can exist.

At this point, we should be suspicious. From a logical or category-theoretic point of view, a non- resource-sensitive evaluation operation involves an implicit copying step, and arbitrary quantum states cannot be copied.

This intuition is confirmed by the technique of ‘encoding unitary maps on an orthonormal basis’, and the ‘no-programming theorem’, presented in [80]. We will see that unitary evaluation may exist, but the names of maps must be computational basis vectors — recall that we cannot copy *arbitrary* quantum states, but a form of copying (i.e. the fan-out operation of Section 3.4) exists for *computational basis* vectors.

Definition 1. Unitary evaluation

Consider a family of unitary maps, $U_1, \dots, U_k : \mathbf{S} \rightarrow \mathbf{S}$ that we wish to encode as members of the (sufficiently large) quantum register \mathbf{R} . We may encode these as orthogonal vectors $\{\psi_1, \dots, \psi_k\}$, and assume (without loss of generality) that this set of vectors is a subset of the computational basis of \mathbf{R} .

The corresponding **unitary evaluation operator** $Eval_U$ is given by

$$Eval_U = \sum_{i=0}^k |\psi_i\rangle\langle\psi_i| \otimes U_i$$

and this satisfies the condition

$$Eval_U(\psi_i \otimes s) = \psi_i \otimes U(s) \quad \forall s \in \mathbf{S}, \psi_i \in \{\psi_1, \psi_2, \dots, \psi_k\}$$

When we take the quantum analogue of the 1-bit computer described above, both the identity and qubit negation (defined on the computational basis as $Not(|0\rangle) = |1\rangle, Not(|1\rangle) = |0\rangle$), may be implemented as unitary maps, so the above prescription gives the quantum **CNOT** gate.

$$Eval_U = CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

6.1 A limit to unitary evaluation operations

It may be shown that at most n unitary maps may be encoded (with respect to a unitary evaluation map), as orthonormal vectors, on an n -dimensional space. The following theorem is based on that of [80] :

Theorem 3. The Nielsen-Chuang ‘no-programming’ theorem

Consider n distinct unitary maps $U_1, \dots, U_n : D \rightarrow D$. Let C be an n -dimensional space, and let $Eval : C \otimes D \rightarrow C \otimes D$ be a unitary map that satisfies

$$Eval(|c_i\rangle \otimes |d\rangle) = |c_i\rangle \otimes U_i(|d\rangle) \quad , \quad \{|c_i\rangle\}_{i=1}^n \subseteq C$$

Then

1. no non-trivial superposition $\alpha|c_i\rangle + \beta|c_j\rangle$ encodes a unitary map.
2. the vectors $\{c_i\}_{i=1}^n$ are all orthogonal.

Taken together, these imply that at most n unitary maps may be encoded in this way.

Proof. Assume that, for some complex α, β satisfying $|\alpha|^2 + |\beta|^2 = 1$, the sum $\alpha|c_i\rangle + \beta|c_j\rangle$ encodes a unitary map V , so for arbitrary $|s\rangle \in \mathbf{S}$,

$$Eval((\alpha|c_i\rangle + \beta|c_j\rangle) \otimes |s\rangle) = (\alpha|c_i\rangle + \beta|c_j\rangle)V(|s\rangle)$$

However, by linearity,

$$\begin{aligned} Eval((\alpha|c_i\rangle + \beta|c_j\rangle) \otimes |s\rangle) &= \alpha Eval(|c_i\rangle \otimes |s\rangle) + \beta Eval(|c_j\rangle \otimes |s\rangle) \\ &= \alpha|c_i\rangle \otimes U_i(|s\rangle) + \beta|c_j\rangle \otimes U_j(|s\rangle) \end{aligned}$$

This may be factorised as $(\alpha|c_i\rangle + \beta|c_j\rangle) \otimes V(|s\rangle)$ under any of the following conditions :

- $\alpha = 0$, in which case $V = U_j$
- $\beta = 0$, in which case $V = U_i$
- $U_j(|s\rangle) = V(|s\rangle) = U_i(|s\rangle)$, for all $|s\rangle \in \mathbf{S}$.

Either the superposition is trivial, or U_i, U_j , and V are all the same unitary operation. Using similar techniques, it may be shown that c_i is orthogonal to c_j , for all $i \neq j$. \square

The above negative result states that (in an entirely unitary, finite-dimensional setting), operations may only be encoded on a fixed orthonormal basis — taken, for convenience, to be the computational basis.

This is sometimes interpreted as stating that quantum computers cannot operate on a ‘stored-code’ principle, since an n -dimensional Hilbert space H can encode at most n unitary operations, whereas there are an infinite number of distinct unitaries from H to itself. In reality, practical proposals for quantum

computation work with a small number of gates, up to a well-defined notion of approximation [87].

A stronger critique is that there is nothing particularly ‘quantum’ about the way operations are encoded — computational basis vectors may be freely copied and deleted (via analogues of fan-out, Section 3.4), and are undisturbed by measurements in the computational basis. This is used to make the much stronger case that there is no advantage to storing program code as *quantum* rather than *classical* information.

There are, of course, two loopholes in the above interpretation, if not in the theorem itself. The first is that it only applies to finite-dimensional spaces. This feature will be the joker in the pack throughout this paper : many results presented (for both quantum and classical information) do not hold in the infinite-dimensional case — this becomes particularly relevant in Section 13.2. The second loophole is more practical: Theorem 3 above only applies to *unitary* evaluation operations.

As well as the orthonormal basis encoding technique, [80] considers using teleportation-like protocols to implement evaluation *probabilistically*, with reference to the Choi-Jamiołkowski correspondence [21, 56]. In [17] it is shown how, in certain cases, the probability of success may be increased to 1 by using unitary operations conditioned on the result of the measurement.

A post-selected form of teleportation, related to the Choi-Jamiołkowski correspondence, is at the core of the ‘categorical foundations’ of quantum mechanics of [4]. We now take a categorical approach, and consider how evaluation arises from general principles, in both the classical and quantum worlds.

To illustrate the general category theory, and to make a link with the von Neumann architecture, we first present the theory of sets and functions. We then use this to motivate the general theory of categorical closure, and consider the particular form of categorical closure exhibited in the quantum world.

7 Evaluation as Currying

From a theoretical computer science perspective, evaluation operations arise from an abstract notion of Currying called *categorical closure*, and the theory of (*monoidal*) *closed categories*. We first present the classical motivation based on the theory of sets and functions (including logic gates and binary words as a special case).

7.1 Evaluation with Sets and Functions

The informal setting for an *Eval* operation is where we can find a representation of a function between two sets as a single element of another set, and can ‘promote’ this to an operation to be applied. From either a category-theoretic

or theoretical computer science point of view, this notion is not primitive but arises naturally as a consequence of the structure of sets and functions — notably the existence of *Currying*.

Definition 2. Cartesian products, Currying

Given sets A and B , their **Cartesian product** is the set defined by

$$A \times B = \{(a, b) : A \in A, b \in B\}$$

Given a function $f : X \times Y \rightarrow Z$, Currying is simply the process of, for each element $x \in X$, defining a function $f_x : Y \rightarrow Z$ by $f_x(y) = f(x, y)$.

Let us use categorical notation (formal definitions follow in Section 8). The category of sets, **Set** has the (proper class of) all sets as its *objects*, denoted $Ob(\mathbf{Set})$. Between any two objects $A, B \in Ob(\mathbf{Set})$ is the collection of *arrows*, $\mathbf{Set}(A, B)$. These are simply the functions from A to B .

For any sets $A, B \in Ob(\mathbf{Set})$,

1. the collection of all functions from A to B is itself a set, that we denote $[A, B] \in Ob(\mathbf{Set})$.
2. the Cartesian product of A and B is itself a set $A \times B \in Ob(\mathbf{Set})$.

The existence of Currying can then be expressed succinctly, as

$$\mathbf{Set}(A \times B, C) \cong \mathbf{Set}(A, [B, C])$$

Now consider the one-object set $I = \{*\}$. It is immediate that for all sets X ,

$$I \times X \cong X \cong X \times I$$

since there is an obvious bijection between $\{(x, *) : x \in X\}$ and X itself.

Hence,

$$\mathbf{Set}(A, B) \cong \mathbf{Set}(I \times A, B)$$

and by Currying,

$$\mathbf{Set}(A, B) \cong \mathbf{Set}(I, [A, B])$$

Finally, for all sets X ,

$$[I, X] \cong X$$

since for all $x \in X$, we have the function $\iota_x : \{*\} \rightarrow X$ defined by $\iota_x(*) = x$.

Hence, Currying and the properties of the one-object set give the bijection $\mathbf{Set}(A, B) \cong \mathbf{Set}(I, [A, B])$ — that is, the existence of representations of functions from A to B as elements of some set. This formalises the intuitive notion of the ‘name’ of an operation (from Section 4), at least in the category of sets and functions.

Definition 3. Given a function $g : A \rightarrow B$ we refer to the arrow $\ulcorner g \urcorner : \{*\} \rightarrow [A, B]$ (or equivalently, the corresponding element of $[A, B]$, considered as a set) as the **name** of the function g .

7.2 Is any of this non-trivial ?

It may be objected that the above manipulations are trivialities — this is to some extent correct⁶. Of more interest is the strong similarity, between $I \times X \cong X \cong X \times I$ and the Hilbert space identity $\mathbb{C} \otimes H \cong H \cong H \otimes \mathbb{C}$. In particular (as formalised in [5]) the identity $[I, X] \cong X$ is strongly reminiscent of Dirac notation, so $\iota_x : \{*\} \rightarrow X$ is the direct analogue of a Ket $|\psi\rangle : \mathbb{C} \rightarrow H$.

In order to consider similarities and differences more closely, we now to take the category-theoretic approach seriously, rather than simply as a form of notation.

8 Basic category theory

As with the section on basic quantum information (Section 3.1), we make no attempt to given anything approaching a comprehensive account of category theory — rather we pick and choose (and to some extent, simplify⁷) topics relevant to our discussion. A comprehensive account may be found in [74], with a physics-oriented approach given by [36]. We also refer to [67] for connections between category theory, logic and lambda calculus, and [14] for a computer science perspective.

Definition 4. Categories

A category \mathbf{C} has a class of **objects**, denoted $Ob(\mathbf{C})$, and between any two objects $X, Y \in Ob(\mathbf{C})$ is a set of **arrows**, denoted $\mathbf{C}(X, Y)$. We often write $f : X \rightarrow Y$ for $f \in \mathbf{C}(X, Y)$, when the category \mathbf{C} is clear from the context.

Arrows $f \in \mathbf{C}(X, Y)$ and $g \in \mathbf{C}(Y, Z)$ may be composed, giving $gf \in \mathbf{C}(X, Z)$, and composition is associative, so $h(gf) = (hg)f$. For each object $Y \in Ob(\mathbf{C})$ there is also an identity arrow $1_Y \in \mathbf{C}(Y, Y)$ satisfying $1_Y f = f$ and $g 1_Y = g$.

As a category \mathbf{C} may have a *proper class* of objects, we cannot use set-theoretic operations on its objects. However, we may define — for example — the category \mathbf{Set} to have all sets as objects, and the arrows $\mathbf{Set}(X, Y)$ to be exactly the set-theoretic functions $f : X \rightarrow Y$.

⁶ We refer to [58] for P. Freyd’s perhaps controversial suggestion that the real function of category theory is to demonstrate that the trivial parts of mathematics are trivial for trivial reasons. Another point of view is that it allows us to formalise similarities and differences between the behaviour of mathematical structures — and we have a special interest in comparing the behaviour of Sets and Hilbert spaces.

⁷ In particular, we will refer to indexed families of arrows in a category as ‘natural’, without giving a formal definition. We refer to [74] for natural families as components of natural transformations, and [22] for an exposition without explicit reference to natural transformations.

8.1 New categories from old

The following operations on categories will be useful:

Definition 5. Opposite categories, product categories

Given a category \mathbf{C} , its **opposite category** \mathbf{C}^{op} has the same objects, and the set of arrows $\mathbf{C}^{\text{op}}(X, Y)$ is exactly the set of arrows $\mathbf{C}(Y, X)$. Given $f \in \mathbf{C}^{\text{op}}(Y, X)$ and $g \in \mathbf{C}^{\text{op}}(Z, Y)$, the arrow $fg \in \mathbf{C}^{\text{op}}(Z, X)$ is exactly the composite $gf \in \mathbf{C}(X, Z)$.

Given categories \mathbf{C}, \mathbf{D} , the **product category** is defined to have, as objects, all pairs (X, A) , where $X \in \text{Ob}(\mathbf{C})$ and $A \in \text{Ob}(\mathbf{D})$. Similarly, an arrow $f : (X, A) \rightarrow (Y, B)$ is just a pair $f = (f_1, f_2)$, where $f_1 : X \rightarrow Y$ and $f_2 : A \rightarrow B$.

Intuitively, the opposite category \mathbf{C}^{op} may be thought of as ‘taking the category \mathbf{C} , and reversing all the arrows’ — although a simple step, this gives many interesting dualities of mathematics (such as Stone dualities between topological spaces and lattices [57], and Pontryagin duality [88]). Also, for every definition or theorem in category theory, we derive a dual definition or theorem by moving to the dual category.

8.2 Structure-preserving maps between categories

As well as categories themselves, it is natural to define structure-preserving maps between categories.

Definition 6. Functors, adjoint pairs

A **functor** between categories $\Gamma : \mathbf{C} \rightarrow \mathbf{D}$ is a map that assigns

- to each object $X \in \text{Ob}(\mathbf{C})$, an object $\Gamma(X) \in \text{Ob}(\mathbf{D})$,
- to each arrow $f \in \mathbf{C}(X, Y)$, an arrow $\Gamma(f) \in \mathbf{D}(\Gamma(X), \Gamma(Y))$.

Functors are required to ‘preserve the categorical structure’ in that for all $f \in \mathbf{C}(X, Y)$ and $g \in \mathbf{C}(Y, Z)$,

$$\Gamma(g)\Gamma(f) = \Gamma(gf) \in \mathbf{D}(\Gamma(X), \Gamma(Z)) \quad \text{and} \quad \Gamma(1_Y) = 1_{\Gamma(Y)}$$

Much of category theory is built on the notion of adjointness. Two functors $\Gamma : \mathbf{C} \rightarrow \mathbf{D}$ and $\Delta : \mathbf{D} \rightarrow \mathbf{C}$ are said to form an **adjoint pair** when, for all $X \in \text{Ob}(\mathbf{C})$ and $Y \in \text{Ob}(\mathbf{D})$, there exists a natural bijection

$$\mathbf{C}(X, \Delta(Y)) \cong \mathbf{D}(\Gamma(X), Y)$$

We say that Γ is **left adjoint** to Δ , or equivalently, that Δ is **right adjoint** to Γ .

Adjointness is a generalisation of the order-theoretic notion of *Galois connections* — indeed, partially ordered sets are themselves categories, and Galois connections are a special case of adjoint functors. The terminology ‘adjoint’ comes from the notion of the adjoint of a continuous linear map of Hilbert spaces, defined by $\langle L(\phi)|\psi\rangle = \langle\phi|L^*(\psi)\rangle$.

A very practical tool in this field is Freyd’s *adjoint functor theorem* that characterises when a given functor has a left (or, by working in the opposite category, a right) adjoint. In [74] it is demonstrated how the tensor product of Abelian groups may be derived, and [53] uses similar techniques to deduce the existence of a tensor product in a very different setting.

For our purposes, adjoint functors will play a key rôle in defining *categorical closure*, giving the general category-theoretic approach to evaluation.

8.3 Monoidal categories

A monoidal tensor for a category is a general notion covering operations such as the Cartesian product of sets and functions, the tensor product or direct sum of Hilbert spaces, disjoint union of Relations, &c. We follow the treatment given in in [74].

Definition 7. Symmetric monoidal categories

A **monoidal category** is defined to be a category \mathbf{C} , together with a functor $\otimes : \mathbf{C} \times \mathbf{C} \rightarrow \mathbf{C}$ that satisfies, for all $A, B, C \in \text{Ob}(\mathbf{C})$:

- **Unit objects** There exists $I \in \text{Ob}(\mathbf{C})$ satisfying $I \otimes A \cong A \cong A \otimes I$.
- **Associativity** $A \otimes (B \otimes C) \cong (A \otimes B) \otimes C$.

If a monoidal category satisfies the additional condition

- **symmetry** $A \otimes B \cong B \otimes A$.

it is called a **symmetric monoidal category**.

The isomorphisms above are required to be natural, and to satisfy various coherence conditions. However, MacLane’s coherence theorems [74] mean that these conditions can generally be ignored with no harmful side-effects. In particular, we treat the associativity isomorphism $A \otimes (B \otimes C) \cong (A \otimes B) \otimes C$ as though it were a strict identity.

From a strongly category-theoretic point of view, monoidal tensors may often be characterised by universal properties that they satisfy, without explicit reference to their behaviour on (for example) elements of some set of objects. For example, The Cartesian product of sets and functions may be characterised as a *categorical product*, as follows :

Definition 8. Products

Let (\mathbf{C}, \otimes) be a monoidal category. The monoidal tensor \otimes is a **product** when, for all objects $X_1, X_2 \in \text{Ob}(\mathbf{C})$ there exist arrows

$$X_1 \xleftarrow{\pi_1} X_1 \otimes X_2 \xrightarrow{\pi_2} X_2$$

such that, for all arrows $f_1 \in \mathcal{C}(Y, X_1)$ and $f_2 \in \mathcal{C}(Y, X_2)$ there exists a unique arrow $\langle f_1, f_2 \rangle \in \mathcal{C}(Y, X_1 \otimes X_2)$ making the following diagram commute :

$$\begin{array}{ccc} & Y & \\ f_1 \swarrow & \downarrow \langle f_1, f_2 \rangle & \searrow f_2 \\ X_1 & \xleftarrow{\pi_1} X_1 \otimes X_2 \xrightarrow{\pi_2} & X_2 \end{array}$$

Recall that by considering the dual category, we may derive a dual definition. Reversing all the arrows in the diagram above gives the following :

Definition 9. Coproducts

Let (\mathbf{C}, \otimes) be a monoidal category. The monoidal tensor \otimes is a **coproduct** when, for all objects $X_1, X_2 \in \text{Ob}(\mathcal{C})$ there exist arrows

$$X_1 \xrightarrow{\iota_1} X_1 \otimes X_2 \xleftarrow{\iota_2} X_2$$

such that, for all arrows $f_1 \in \mathcal{C}(X_1, Y)$ and $f_2 \in \mathcal{C}(X_2, Y)$ there exists a unique arrow $[f_1, f_2] \in \mathcal{C}(X_1 \otimes X_2, Y)$ that makes the following diagram commute :

$$\begin{array}{ccc} & Y & \\ f_1 \nearrow & \uparrow [f_1, f_2] & \nwarrow f_2 \\ X_1 & \xrightarrow{\iota_1} X_1 \otimes X_2 \xleftarrow{\iota_2} & X_2 \end{array}$$

Examples

The Cartesian product of sets and functions is, as noted above, a categorical product. Dually, the disjoint union of sets and functions is a categorical coproduct. The direct sum of Hilbert spaces is both a product, and a coproduct (and hence a **biproduct**). The tensor product of Hilbert spaces is neither a product nor a coproduct — however, it may also be defined in terms of a universal property, as we now show.

8.4 (Monoidal) closed categories

The most general setting for a correspondence between objects and arrows in a category, and evaluation operations, is the field of *closed categories* [31, 68, 69]. We do not attempt an exposition of this, but work with the special case of *monoidal closed categories* — partly for simplicity, and partly because the specific examples we wish to consider are monoidal closed rather than simply closed.

Definition 10. Monoidal closed categories

Let (\mathbf{C}, \otimes) be a monoidal category. We say that it is **monoidal closed** when there exists a functor

$$[_, _]: \mathbf{C}^{op} \times \mathbf{C} \rightarrow \mathbf{C}$$

called the **internal hom functor**, such that for fixed $B \in Ob(\mathbf{C})$, the functors given by

$$[B, _] : \mathbf{C} \rightarrow \mathbf{C} \quad \text{and} \quad _ \otimes B : \mathbf{C} \rightarrow \mathbf{C}$$

form an adjoint pair.

This definition, although concise, is relatively abstract (and is only strictly accurate in the symmetric monoidal case). The following characterisations make a link to both an abstract form of Currying, and evaluation maps.

Theorem 4. The following are equivalent to the definition of a monoidal closed category, in Definition 10 above :

1. There exists an internal hom functor

$$[_, _]: \mathbf{C}^{op} \times \mathbf{C} \rightarrow \mathbf{C}$$

satisfying

$$\mathbf{C}(A \otimes B, C) \cong \mathbf{C}(B, [A, C])$$

2. For every pair of objects $A, B \in Ob(\mathbf{C})$, there exists

- an object $[A, B]$,
- an arrow $ev_{A,B} \in \mathbf{C}(A \otimes [A, B], B)$

where, for all $f : A \otimes X \rightarrow B$, there exists unique $g \in \mathbf{C}(X, [A, B])$ such that the following diagram commutes :

$$\begin{array}{ccc} A \otimes X & \xrightarrow{f} & B \\ & \searrow 1_A \otimes g & \uparrow ev_{A,B} \\ & & A \otimes [A, B] \end{array}$$

Proof. Proofs may be found in any text on category theory or categorical logic (e.g. [74, 67]). \square

Part 1. of Theorem 4 above provides the link with an abstract notion of Currying, and part 2. of the same theorem demonstrates that this is equivalent to the existence of an evaluation map satisfying the expected properties.

9 Categorical closure and Hilbert spaces

We now describe the particular form of compact closure exhibited by the category of (finite-dimensional) Hilbert spaces and linear maps, and its interpretation as quantum-mechanical protocols.

It has long been known that the collection of all linear maps between (finite-dimensional) Hilbert spaces H, K is itself a Hilbert space. This is a reflection of the categorical closure of the category of finite-dimensional Hilbert spaces – however, the categorical closure is of a particularly simple form. We first give an abstract exposition of this form of closure, including Hilbert spaces as a concrete example, and then give an overview of the ‘categorical foundations’ approach to quantum mechanics of [4].

9.1 Compact closed categories

Definition 11. Compact closure

A symmetric monoidal category (\mathbf{C}, \otimes) is called **compact closed** when, for all $A \in \text{Ob}(\mathbf{C})$, there exists a **dual object** $A^* \in \text{Ob}(\mathbf{C})$ such that the functor $A \otimes _ : \mathbf{C} \rightarrow \mathbf{C}$ is left adjoint to the functor $A^* \otimes _$.

Although this definition is category-theoretically elegant — particularly when viewed as 2-category theory [63] — for our purposes it will be easier to work with the following characterisation, also given in [63] :

Theorem 5. A symmetric monoidal category (\mathbf{C}, \otimes) is compact closed when, for every object $A \in \text{Ob}(\mathbf{C})$, there exists a **dual object** $A^* \in \text{Ob}(\mathbf{C})$ together with distinguished arrows

- The unit arrow $\epsilon_A : A \otimes A^* \rightarrow I$
- The counit arrow $\eta_A : I \rightarrow A^* \otimes A$

that satisfy

$$\lambda_A(\epsilon_A \otimes 1_A)(1_A \otimes \eta_A)\rho_A^{-1} = 1_A \quad \text{and} \quad \rho_{A^*}(1_{A^*} \otimes \epsilon_A)(\eta_A \otimes 1_{A^*})\lambda_{A^*}^{-1} = 1_{A^*}$$

(where λ_X, ρ_X are the indexed isomorphisms exhibiting the identity $I \otimes X \cong X \cong X \otimes I$).

Using the diagrammatic notation of [59, 60, 61], this may be drawn as shown in Figure 2.

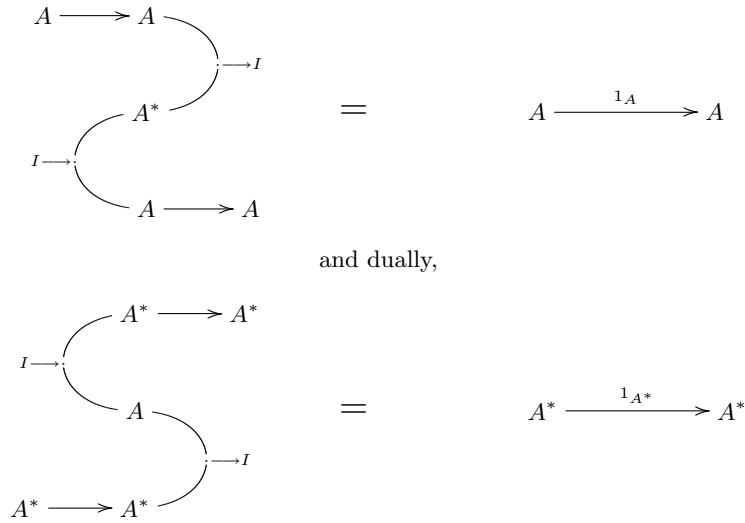
The dual operation on objects $(_)^*$, together with the unit and counit arrows may be used to define the **dual on arrows**. Given $f \in \mathcal{C}(A, B)$, then $f^* \in \mathcal{C}(B^*, A^*)$ is defined by

$$f^* = (1_{A^*} \otimes \epsilon_B)(1_{A^*} \otimes f \otimes 1_{B^*})(\eta_A \otimes 1_{B^*}) : B^* \rightarrow A^*$$

Diagrammatically, this is as shown in Figure 3.

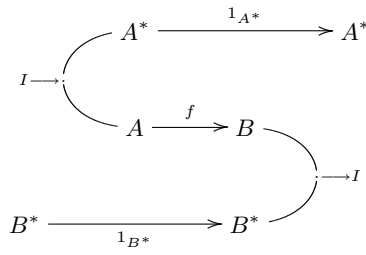
Note that in a compact closed category, the arrows η_A and μ_A are dual, so $\eta_A^* = \epsilon_A$.

Fig. 2. Axioms for compact closure



Note that (following the usual convention) the above diagrams omit the unit object isomorphisms $A \cong A \otimes I$, &c.

Fig. 3. The dual operation on arrows



9.2 The internal hom, and names in compact closed categories

Compact closed categories have a particularly simple description of both the internal hom object, and names of arrows.

Theorem 6. *Let $(\mathbf{C}, \otimes, \epsilon, \eta)$ be a compact closed category, with $A, B \in Ob(\mathbf{C})$. Then*

- *The internal hom is given by $[A, B] = B \otimes A^*$.*
- *Given an arrow $f \in \mathbf{C}(A, B)$, its name $\ulcorner f \urcorner : I \rightarrow B \otimes A^*$ is given, up to a symmetry map, by*

$$\lceil f \rceil = (1_A^* \otimes f) \eta_A$$

Using the same diagrammatic notation as previously, the name of an arrow f is given by

$$I \rightarrow \begin{array}{c} \curvearrowright \\ A^* \xrightarrow{1_{A^*}} A^* \\ \curvearrowleft \\ A \xrightarrow{f} B \end{array}$$

Note that both the ability to name arrows, and the adjunction giving monoidal closure, rely on the existence of symmetry isomorphisms $\sigma_{X,Y} : X \otimes Y \rightarrow Y \otimes X$, although this requirement can be weakened somewhat [76, 92].

Definition 12. conames - the dual notion to names

In compact closed categories — by contrast to monoidal closed categories generally — there is a dual notion to naming; that of the coname of an arrow. Given an arrow $f \in \mathbf{C}(A, B)$, the **coname** is an arrow $\lfloor f \rfloor \in \mathbf{C}([X, Y], I)$ given (dually to Theorem 6) by $\lfloor f \rfloor = \epsilon_B(f \otimes 1_{B^*})$.

To see that conames do not exist in all monoidal closed categories, consider the category of sets and functions — here, for any set X , there exists exactly one function in $\mathbf{Set}(X, \{*\})$. This is a strong, and indeed significant, difference between the behaviour of evaluation for Sets and functions, and Hilbert spaces and linear maps. This difference accounts for the different behaviour of classical and quantum systems⁸ given in Section 12.

9.3 Compact closure and Hilbert spaces

It has long been known that finite-dimensional Hilbert spaces are a canonical example of compact closed categories — and equally, Hilbert spaces in the general setting (i.e. allowing for infinite-dimensional spaces) are not compact closed [2].

Let us denote the category of finite-dimensional Hilbert spaces by $\mathbf{Hilb}_{\text{fd}}$. Arrows are linear maps (and hence, as we are working in the finite-dimensional case, both bounded and continuous), and we use the tensor product as the monoidal tensor.

Compact closure is easily exhibited. Consider a space H , with orthonormal basis $\{e_j\}_{j=1}^N$. Objects are self-dual, so $H^* = H$, and the unit and counit arrows $\epsilon : H \otimes H \rightarrow \mathbb{C}$ and $\eta_H : \mathbb{C} \rightarrow H \otimes H$ are given (using Dirac notation) by

⁸ We emphasise that, although the category \mathbf{Set} does not admit conames, they are by no means an exclusively quantum phenomenon — rather, they are simply a property associated with compact closure. For example, [47] uses compact closure in modelling classical Turing machines.

$$\epsilon_H = \sum_{j=1}^N \langle \mathbf{e}_j \mathbf{e}_j | : H \otimes H \rightarrow \mathbb{C} \quad \text{and} \quad \eta_H = \sum_{j=1}^N |\mathbf{e}_j \mathbf{e}_j\rangle : \mathbb{C} \rightarrow H \otimes H$$

It is straightforward to check that the axioms of Definition 11 are satisfied.

The compact closure of Hilbert space, and its physical interpretation in terms of teleportation protocols, is at the core of Abramsky and Coecke’s ‘categorical foundations’ program for quantum mechanics [4]. We consider this in from Section 10 onwards, but first characterise states that are the names of unitary maps.

9.4 Naming unitary maps

Our ultimate aim is to describe what the state-map correspondence given by the categorical foundations program and the compact closure of finite-dimensional Hilbert space can tell us about the existence, or otherwise, of quantum-mechanical versions of evaluation. From the motivation given in Sections 3.1 and 3.3, we are particularly interested in coherent quantum operations.

We now consider the state / map correspondence provided by the compact closure. We emphasise that this is *not* original to this paper. It is given explicitly in [4], and is implicit in the Choi-Jamiołkowski correspondence between density matrices and completely positive maps [21, 56, 91].

Unwinding the definition of the name in a compact closed category gives a 1:1 correspondence between arrows $\mathbf{Hilb}_{\mathbf{fd}}(\mathbb{C}, H \otimes H)$, and arrows $\mathbf{Hilb}_{\mathbf{fd}}(H, H)$. Let us chose an orthonormal basis $\{\mathbf{e}_i\}_{i=1}^N$ for H , and consider a linear map described as a matrix $M = (m_{ij})_{i,j=1}^N$ on this basis. For any such matrix M we may define $\ulcorner M \urcorner \in \mathbf{Hilb}_{\mathbf{fd}}(\mathbb{C}, H \otimes H)$ by

$$\ulcorner M \urcorner = \frac{1}{\sqrt{N}} \sum_{\alpha, \beta=1}^N m_{\alpha\beta} (|\mathbf{e}_\alpha\rangle \otimes |\mathbf{e}_\beta\rangle)$$

This naming operation is invertible; consider a vector $\psi \in H \otimes H$. Then $\psi = \ulcorner L \urcorner$, where

$$L = \begin{pmatrix} l_{11} & \dots & l_{1N} \\ \dots & & \dots \\ l_{N1} & \dots & l_{NN} \end{pmatrix}$$

satisfies

$$l_{ij} = \sqrt{N} \langle \mathbf{e}_i \otimes \mathbf{e}_j | \psi \rangle$$

Given this explicit description, it is clear that arbitrary linear maps on H may be named. We now characterise those states that name unitary maps :

Theorem 7. *The pure states that name unitary maps $U : H \rightarrow H$ are exactly the maximally entangled states of $H \otimes H$.*

Proof. In order not to disrupt the expository flow of this paper, the proof of this result is given in Appendix A.

9.5 The Choi-Jamiołkowski correspondence

The correspondence between states and linear maps is a special case of the **Choi - Jamiołkowski correspondence** between completely positive maps and density matrices, discovered independently by M. Choi [21] and A. Jamiołkowski [56]. As our exposition is in terms of *pure states* and *unitary maps*, rather than the more general *density matrices* and *completely positive maps*, so have given the correspondence in this restricted case, following [4].

It is demonstrated in [91] that the category of density matrices and completely positive maps is compact closed, and the state corresponding to a completely positive map under the Choi-Jamiołkowski correspondence is exactly the name of that map.

Finally, the connection between teleportation and the Choi-Jamiołkowski correspondence had been commented on (although not explored in detail) in [80]. The interpretation of teleportation as compact closure was developed in the *categorical foundations* program of [4].

10 Abramsky & Coecke's categorical foundations for quantum mechanics

As with the introductory sections on both quantum information and category theory, we make no attempt to give a coherent account or consistent history of the categorical foundations program – rather, we concentrate on those topics relevant to our interest in evaluation operations and the von Neumann architecture. We refer to [4, 5, 23, 24] for details, and other articles in this volume for the current state of research.

10.1 Teleportation, traditionally

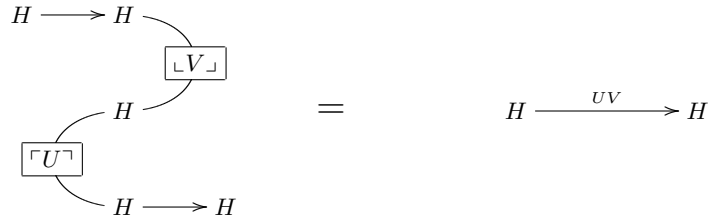
The traditional description of teleportation is as follows : **Alice** has a quantum bit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ that she wishes to send to **Bob**. Alice is spatially separated from Bob, but they had previously shared a maximally entangled pair of particles $|\mathcal{B}ell\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. This gives the overall state of the system as $|\psi\rangle \otimes |\mathcal{B}ell\rangle =$

$$\frac{1}{\sqrt{2}}(\alpha|0\rangle + \beta|1\rangle) \otimes (|00\rangle + |11\rangle) = \frac{\alpha}{\sqrt{2}}(|000\rangle + |011\rangle) + \frac{\beta}{\sqrt{2}}(|100\rangle + |111\rangle)$$

Alice then performs a measurement of her subsystem (i.e. the qubit $|\psi\rangle$, together with her half of the maximally entangled pair $|\mathcal{B}ell\rangle$) against a maximally entangled basis that contains the $\mathcal{B}ell$ state.

Let us assume that Alice observes the state $\mathcal{B}ell$. Bashing through the appropriate Hilbert space calculations will demonstrate that the system is now in the overall state $\frac{\alpha}{\sqrt{2}}(|000\rangle + |110\rangle) + \frac{\beta}{\sqrt{2}}(|001\rangle + |111\rangle)$. Of course, this

Fig. 4. Applying maps via teleportation (categorically)



11 Evaluation by teleportation, and the vN architecture

11.1 The story so far

So far, we have seen that ‘evaluation’ is the key part of the von Neumann architecture (Section 4). The only possible competition for this rôle is ‘relative addressing’, and we have seen in Section 4.1 that this arises quite naturally from evaluation. We have also seen that evaluation is a categorical property that arises from *monoidal closure* — an abstract form of Currying, and the notion of *naming* an arrow.

The link with quantum mechanics follows from the categorical foundations program where compact closed categories are not only used, but are a key part of the program. Physically, compact closure is interpreted as the teleportation protocol of [16], and in general, the *implementing a logic gate by teleportation* of [17]. This strongly suggests that a (resource-sensitive) form of evaluation is available, and may be implemented in the quantum world. Modulo questions of reversibility and resource-sensitivity, can we therefore describe some form of von Neumann architecture for quantum computers?

The question that this section aims to answer is therefore:

“Can we apply an unknown unitary map to a quantum state?”

The ‘unknown unitary map’ is given as a quantum resource – i.e. we are given its name¹⁰; a maximally entangled state vector $\lceil U \rceil \in H \otimes H$. Our question now is, given $\lceil U \rceil \in H \otimes H$, and a state vector $|\psi\rangle \in H$, can we reliably produce $U(|\psi\rangle) \in H$?

¹⁰ A natural question at this point is, ‘why not give the unknown map as a *coname*, rather than a name? From the physical interpretation of Section 10.2, a coname is interpreted as a (successful) measurement; that is, it is derived from a classically determined measurement apparatus. It is hard to see how we may go from an arbitrary quantum state to a measurement against some basis containing that state — thus the coname can only be given as classical information.

11.2 Postselection, and unitary corrections

Recall how, as shown in Figure 4., unitary maps may be applied using a *teleportation protocol*. By letting the name in this diagram be our ‘unknown map’, and taking the coname to be the coname of the identity, we are able to apply our unknown map to an arbitrary state $|\psi\rangle \in H$.

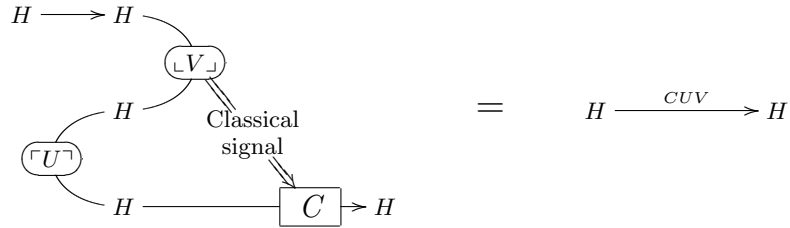
Unfortunately, as it stands, this diagram describes a *post-selected* protocol; if the measurement does not yield the required result (i.e the coname of the identity) we abandon the experiment and start again. Unfortunately, the experimenter has no control over the actual result of measurement — at best, he may specify a complete maximally entangled orthonormal basis set $\{\lceil V_j \rceil\}_{j=1\dots n^2} \subseteq H \otimes H$, and measure against that. Thus, when working with a single qubit, we expect to observe the ‘correct’ outcome with a probability of $\frac{1}{4}$.

This feature is why Nielsen & Chuang [80] refer to evaluation via teleportation protocols as ‘probabilistic evaluation’. However, in the original teleportation protocol, [16], Bob applies a unitary operation to correct for this ‘incorrect experimental outcome’. In [17], it is demonstrated how a similar technique can be used to implement certain quantum logic gates, with unit probability.

In the categorical foundations model, as presented in [4], the classical information flow and conditioning of a unitary correction on the result of a measurement are modelled using biproducts and canonical distributivity and associativity isomorphisms. We do not give an exposition of the categorical treatment of classical information here (see [24] for more details), but simply consider under what conditions a unitary correction may be applied, to give the desired result.

With a unitary correction, Figure 4. becomes as shown in Figure 5. Of

Fig. 5. Application by teleportation, with unitary correction



course, what we wish to apply is the unitary $U : H \rightarrow H$, rather than the composite $CUV : H \rightarrow H$ — thus we need conditions for these two to be equal. The most general solution is $C = UV^{-1}U^{-1} : H \rightarrow H$ — however, C is a classically determined correction, and there is no classical information

available about the operation U itself (from a von Neumann architecture point of view, it may have been loaded into some ‘quantum instruction register’ from memory, and be the outcome of some previous quantum computation).

Therefore, the unitary $C : H \rightarrow H$ may be conditioned on the measurement outcome $\perp V \perp$, but cannot be dependent on U . This immediately imposes a restriction on the class of unitaries that may be implemented by teleportation.

It is immediate that operations that commute with all members of $\{V_j\}_{j=1}^{n^2}$ may be implemented deterministically, simply by taking $C = V^{-1}$. Slightly more generally, let us assume (without loss of generality — see Section 11.3 below) that $\{V_j\}_{j=1}^{n^2}$ form a group \mathcal{G} . In this case, we may implement the group \mathcal{C} of unitary operations that satisfy $V_j^{-1}UV_j \in C$, for all $V_j \in \mathcal{G}$ and $U \in \mathcal{C}$.

Thus we cannot deterministically implement all unitary maps in using a teleportation protocol — the question now is : how severe is this restriction, and can we still do useful quantum computation ?

11.3 Choosing a measurement basis

The only choice the experimenter has in the protocol shown in Figure 5. is the choice of measurement basis — this must be a maximally entangled basis set for $H \otimes H$. As noted above, the basic assumption is that we have no information about the unitary map U , so in the general case there is no particular reason to favour one maximally entangled orthonormal basis over another. In any case, they are all equivalent up to some unitary isomorphism.

In a more explicitly computational setting, let us assume that H is the tensor product of a number of qubits. In this case, for both experimental and theoretical reasons, it is common to choose a basis based on the *Pauli group*.

Definition 13. Pauli groups, Bell basis

*Consider the 2-dimensional Hilbert space of qubits Qu , with orthonormal basis $\{|0\rangle, |1\rangle\}$. The **(1-qubit) Pauli group** \mathcal{G}_1 that acts on Qu consists of the following unitary operations (given as matrices)*

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

The names of these operations form a (maximally entangled) orthonormal basis for the space $Qu \otimes Qu$ called the Bell basis.

*In the general case, the **Pauli group** G_n is the n -fold tensor product of G , so $\mathcal{G}_n = \{\otimes_{j=1}^n W_j : W_j \in \mathcal{G}_1\}$. Note that the names of the members of \mathcal{G}_n also form a maximally entangled orthonormal basis for the space $\otimes_{j=1}^n Qu$.*

Physically, Pauli matrices correspond to the observables of spin $\frac{1}{2}$ particles — i.e. fermions such as protons, neutrons, &c. The names of Pauli group operations also form a very convenient maximally entangled basis for teleportation experiments, because of the following property:

Proposition 1. *Let U be a member of the Pauli group \mathcal{G}_n , and consider an n -qubit state $|\psi\rangle \in \otimes_{j=1}^n \text{Qu}$. Then, experimentally, $U(|\psi\rangle)$ may be realised by 1-qubit operations.*

Proof. This is immediate from the definition of \mathcal{G}_n as the tensor product of a number of copies of \mathcal{G}_1 . \square

Corollary 1. *When using the names of the Pauli group \mathcal{G}_n as the measurement basis for a teleportation protocol (as shown in Figure 5.), the required unitary corrections may be carried out as a series of 1-qubit operations.*

Corollary 2. *When using the names of the Pauli group \mathcal{G}_n in a teleportation protocol, the operations that may be implemented deterministically are exactly the group \mathcal{C}_n of operations that satisfy $\mathcal{C}_n = \mathcal{G}_n^{-1}\mathcal{C}_n\mathcal{G}_n$.*

11.4 The Clifford group, and the Gottesman-Knill theorem

The group specified in Corollary 2 above is well-known as the **Clifford group** \mathcal{C}_n — the stabiliser of the Pauli group \mathcal{G}_n . It is key to a number of different fields, including quantum error-correction, and measurement-based computation.

It is also the basis of one of the most significant recent results on quantum computation — the **Gottesman-Knill theorem** :

Theorem 8. *Any quantum circuit built up from :*

- *Computational basis preparations,*
- *Clifford group operations,*
- *Computational basis measurements*

may be efficiently simulated on a classical computer.

Proof. This is proved in [40] — see also [81] for a good exposition.

Corollary 3. *If we wish to use teleportation to provide a deterministic evaluation operation, satisfying the conditions laid out in Section 11.1, we are restricted to quantum operations that can be efficiently simulated by a classical computer.*

A comment : measurement-based computing

For readers familiar with measurement-based computation, and the correspondence with implementing operations via teleportation given in [7], nothing in the above discussion should be interpreted as stating that measurement-based computation is restricted to the Clifford group. In particular, measurement-based computing is about applying *known* unitary operations, and the resulting feed-forward of classical information on experimental outcomes and corresponding unitary corrections is well-studied — see [26, 27] for a formal approach with a very theoretical computer science flavour.

A Question : other measurement bases

A natural question at this point is whether choosing an alternative measurement basis (i.e. not based on the Pauli operations) will give us qualitatively different results. However, recall that given two orthonormal basis sets B_1, B_2 for a space $H \otimes H$, we may give a unitary map $D : H \otimes H \rightarrow H \otimes H$ that maps one to the other — all orthonormal basis sets are equivalent up to isomorphism.

12 Naming an unknown arrow

After presenting such a negative result, it would be nice to discover something that quantum evaluation can do that classical evaluation cannot. So far, we have been considering how *data* may be interpreted as *code*, and applied to some other datum. Where the quantum setting wins out is in the dual process — given some implementation of a fragment of code, how may we *physically* produce the datum representing it (the *name* of the function)?

We treat a physical implementation of an instruction as a **black box** that we may give an input, and in return receive an output. In the classical case, we assume the black box is simply a function between the input and output sets, and in the quantum case, we assume a unitary map from the input to the output space¹¹.

The question is then, given such a black box, how may we produce its name, as a physical resource ?

¹¹ In both cases, we ignore questions of timing & assume that the time spent processing is constant, regardless of the input. However, if the processing time is *not* constant, it maybe measured, and gives additional (classical) information about the operation of both quantum and classical black boxes. See [64] for applications of this in classical cryptography, and [18, 43, 53, 73] for the key rôle that processing time plays in quantum computation.

12.1 The classical case

In the classical case, the situation is depressingly straightforward. We are given a black box that implements some function between finite sets, $f : X \rightarrow Y$. We have no additional information about the function f , but we wish to produce $\ulcorner f \urcorner \in Y \times X$, the ‘name’ of $f : X \rightarrow Y$.

In the absence of other information about f , the only option is a brute-force investigation — we must feed into the black box every element of X , and record the output in each case¹². Hence, the number of steps required to give the name of a function $f : X \rightarrow Y$ is exactly $|X|$. Note that the success of this procedure depends on

- X is a finite set.
- the black box has no ‘internal states’ — given an input $x \in X$ it return the same output $f(x) \in Y$ regardless of the previous set of inputs to the black box.

12.2 The quantum case

The quantum case is also remarkably straightforward. Let us assume that the black box implements some unitary operation $U : H \rightarrow H$, and is subject to similar constraints to the classical case — i.e. a unitary map is implemented in constant time, and the input does not become entangled with some internal state of the black box.

Now recall the definition of the name of an arrow in a compact closed category, given diagrammatically as

$$I \rightarrow \left(\begin{array}{c} H \xrightarrow{1_H} H \\ H \xrightarrow{U} H \end{array} \right)$$

Interpreting the counit as the preparation of a maximally entangled pair (i.e. the name of the identity map), we simply create such an entangled pair, and pass one half of this pair through the black box, and do nothing at all to the other half. The resulting quantum state (taken as a whole) is then the name of the operation implemented by the black box.

Thus, creating the name of an unknown operation is a 1-step operation in the quantum case — compared to an arbitrarily long procedure in the classical case. Although this is arbitrarily more efficient for the quantum case, we may wonder what use it is ...

¹² Even in the presence of a number of identical copies of the black box, and finite input / output sets, this procedure is at best tedious. In the infinite case, it is straightforwardly impossible — thus from a physical point of view, *arbitrary* functions cannot be named, even in the classical world. Given space / time bounds, the question of which *computable* functions may be named is left as an interesting exercise.

12.3 A fiction about Alice and Bob

Few papers on quantum computation or information are complete without a story about the QM information researchers, *Alice* and *Bob*. This paper is no exception, although the presented interpretation is more fanciful than most.

- Let us assume that Bob has developed a quantum computer \mathcal{Q} that implements some interesting unitary map U on n -qubit registers. Alice, on the other hand, holds an n -qubit register R that she wishes to process using Bob's computer.
- Bob is happy for Alice to make this calculation, but does not wish to loan his computer to Alice — he may wish to prevent her reverse-engineering it by repeated applications to elements of the computational basis, or perhaps she still has not returned the textbooks she 'borrowed' when they used to share an office.

To get around this impasse, Bob works in the space of $2n$ -qubit registers (and hence a space of size 2^{2n} , and produces the maximally entangled resource $\mathbf{B} = \lceil Id \rceil$, where Id is the identity map. He then applies U to the final n registers of \mathbf{B} to get some new resource \mathbf{B}' , where \mathbf{B}' is the *name* of the unitary map U . Finally, he transmits the whole of \mathbf{B}' to Alice.

Alice then treats the resource \mathbf{B}' as the name of an unknown unitary map, and using the procedure described in Section 12.2 is able to produce $U(R)$, the result of applying Bob's quantum computer \mathcal{Q} to her quantum register R .

In this manner, Bob may give out 'samples' of his quantum computer \mathcal{Q} that can be used exactly once, without Alice ever getting her hands on the precious machine. Perhaps the best conclusion to draw from this is that Digital Rights Management is much easier to enforce in the quantum world !

Unfortunately, if Alice is ever to implement Bob's unknown operation *reliably*, it must be a member of the Clifford group. So, it seems that the Quantum Rights ManagementTM protocol is in fact about distributing limited copies of classical programs.

However, if Bob can be persuaded to be a little less paranoid about Alice reverse-engineering his computer, he may find out from [87] that all he needs to perform universal quantum computation is *Clifford group operations*, and *single $\frac{\pi}{8}$ phase-shifts*. He may then split the above protocol into several parts, sending Alice either the name of some Clifford group operation, or a classical instruction to perform a $\frac{\pi}{8}$ phase-shift to a particular qubit, until the computation is complete.

Open question : If Bob performs the procedure described above, how much information can Alice deduce about the structure of his quantum computer ?

13 Other Aspects

Our search for a quantum analogue of the von Neumann architecture has involved a lot of digressions into assorted, undeniably interesting, topics. Our perhaps unsurprising conclusion is that quantum computers cannot implement a suitable form of evaluation. However, our interest in quantum analogues of the vN architecture was not arbitrary — rather, we were interested in the computational features given in the list of Section 2.3. Although the behaviour of evaluation in the quantum setting prevents us from forming a direct analogue of the von Neumann architecture, it is worthwhile to consider topics related to this list directly.

13.1 Computational universality & quantum computers

Much has been made of the fact that the von Neumann architecture allows for *computationally universal* machines. By *computationally universal* we mean, ‘capable of performing any computation that may be performed on a Turing machine’. By the *Church-Turing* thesis, this is believed to cover any computation that can be effectively, or mechanically performed [44, 25].

The EDVAC machine, running the von Neumann architecture, was not the first computationally universal physical machine — this honour is believed to belong to the Z3 machine of Konrad Zuse [90] (also see [89] for an in-depth discussion, and the architecture of the Z3). Had the *analytic engine* of Charles Babbage [95] been completed, this would have claimed priority by at least 100 years.

Our interest in the von Neumann architecture has more been in the high-level control structures that arise naturally. However, it is worthwhile to consider whether a quantum computer can be computationally universal¹³. In one sense, this question is trivial; when restricted to a fixed computational basis, a quantum Turing machine [29] behaves exactly as a classical reversible Turing machine, and these are known to be computationally universal.

However, as discussed in Section 6.1, it is hard to see how a device restricted to a computational basis — for both ‘code’ and ‘data’ — may be considered in any way a *quantum* computer. A more interesting question is whether a computer may be both *coherent* and *computationally universal* —

¹³ We strongly distinguish this from the idea of a *universal quantum gate* that can simulate (up to a reasonable approximation) any other quantum gate. In particular, universal computation requires the possibility of non-termination of an algorithm . . . and indeed the undecidability of termination is a fundamental theorem of theoretical computer science [55]. Both the usual quantum circuit model [18] and the (restricted forms of) quantum Turing machines contained in [18] require unconditional termination in exactly K steps, where K is some a priori fixed value.

‘fully quantum’ in the terminology of [29, 78]. This question has been intensively studied, with no definite conclusion [78, 73, 43, 77] — and as always, the finite-dimensional case is much simpler [53].

Finally, the same features that make the von Neumann architecture universal contribute to its utility as a basis for high-level languages — although the connection between the ability to produce high-level languages, and computational universality, is far from clear. (The papers [48, 50, 52] attempt to axiomatise the notion of ‘high-level / low-level languages’ and ‘low-level languages’ in terms of domain theory, and make connections to computational universality. A quantum-mechanical version of this theory has also been presented in [51], but the situation there is even less clear).

13.2 Logical and lambda-calculi interpretations of monoidal closure

Although we have emphasised the interchangeability of code and data as the key to the von Neumann architecture, it is more traditionally associated with other fields of theoretical computer science — Church’s λ calculus, and categorical logic (and, of course, the connections between lambda calculi and logics given by the Curry-Howard isomorphism [93]). We refer to [70, 67, 14] for an introduction to categorical logic.

The logical interpretation of categories used to model quantum protocols is given by Abramsky and Duncan, in [30, 6]. We do not give an exposition of this, but rather consider the conditions required for an *untyped* analogue, in the search for a computational system similar to the untyped lambda calculus.

Traditionally, the pure untyped lambda calculus is modelled by a *C-monoid*, or one-object Cartesian closed category¹⁴ without a terminal object.

Given a Cartesian closed category (\mathbf{C}, \times) , a C-monoid is the endomorphism monoid of an object $D \in \text{Ob}(\mathbf{C})$ that satisfies

$$D \cong D \times D \cong [D, D]$$

The identity $D \cong D \times D$ is relatively easy to satisfy : for sets and functions, and many other categories, this is satisfied by any countably infinite set (see [45, 46] for the general setting). However, $D \cong [D, D]$ is less easy to satisfy — simple cardinality arguments demonstrate that no object in the category of sets and functions may satisfy this identity. The usual route to objects

¹⁴ The connection between C-monoids and Church’s lambda calculus is not straightforward. As observed in [67], the product structure is equivalent to requiring *surjective pairing* in the lambda calculus. We also refer to [67] for a demonstration — via Occam’s Razor — of how *combinatory logic* arises from C-monoids, without explicit reference to products, i.e. using the closed, but not monoidal closed, structure — giving what are referred to as ‘monstrous’ coherence conditions. Note also that [3] gives a (computationally universal) linear combinatory logic in terms of (untyped) *compact closure*, rather than *Cartesian closure*.

satisfying such identities is via *domain theory*. See [1] or [75] for a general perspective, and [67] for a categorical exposition related to C-monoids.

In compact closed categories, the situation is both simpler, and more subtle. The simple form of the internal hom object $[X, Y] = Y^* \otimes X$ means that $N \cong [N, N]$ is equivalent to self-duality $N = N^*$, together with the identity $N \cong N \otimes N$. Given any object $N \cong N \otimes N$, note also that $G = N \otimes N^*$ satisfies

$$G \cong G \otimes G \cong [G, G]$$

We refer to [45, 49] for an explicit description of one-object analogues of compact closed categories.

In the particular case of Hilbert spaces, any separable infinite-dimensional Hilbert space H certainly satisfies $H \cong H \otimes H$. However, recall from [2] that only the category of finite-dimensional spaces is compact closed — thus, it is hard to see how evaluation in the quantum setting may be used to produce some form of (untyped) lambda calculus or combinatory logic. Infinitary versions of the Choi-Jamiołkowski correspondence have been explored in [84], in the context of order theory and C* algebras, but the categorical interpretation is not straightforward.

13.3 Backus, Functional Languages, and non- von Neumann architectures

Throughout this paper, we have praised the von Neumann architecture as a significant advance in both theoretical and practical computer science. This is certainly the case; however, many programmers and theoreticians also see the near-universal reliance on it as an impediment, particularly with regard to either parallel or asynchronous computation. The ‘*von Neumann bottleneck*’ is a common term, first appearing in J. Backus’ Turing award acceptance speech, “Can Programming be Liberated from the von Neumann Style ?” [12]).

Thus, although we consider it unfortunate that quantum computers cannot run some analogue of the von Neumann architecture, it may instead be seen as an opportunity.

Backus’ speech gives a strong defence of *functional programming* and functional programming languages. Functional programs are based on the notion of evaluating functions, rather than updating states. Unfortunately, they are often easier to characterise in terms of features they do *not* possess, such as

“Functional programming languages have no variables, no assignment statements, and no iterative constructs. This design is based on the concept of mathematical functions, which are often defined by separation into various cases, each of which is separately defined by appealing (possibly recursively) to function applications.” — [33].

There is no space here for an exposition of the positive aspects of functional languages and programming — we refer to Backus’ speech and subsequent works [12, 13, 54] for positive properties such as *referential transparency*, *lazy evaluation*, *freedom from side-effects* and *state-freeness*¹⁵.

As well as a plea for *programming languages* based on different principles, it is clear that Backus considered the von Neumann style of programming to be a direct consequence of the von Neumann architecture, and the persistence of the von Neumann architecture to be due to the universality of languages based on it. From [12],

“Our fixation on von Neumann languages has continued the primacy of the von Neumann computer ... The absence of programming styles founded on non- von Neumann principles has deprived designers of an intellectual foundation for new computer architectures.”

A stated aim of [12] is thus to provide programming concepts and languages that naturally lead to different underlying computer architectures — unfortunately, functional programs still tend to be executed on vN architecture machines!

Another key point of this program is that functional programming languages could, or should, come equipped with an ‘algebra of combining forms’. Such an algebra is a system whereby one may, *“solve equations whose ‘unknowns’ are programs, in much the same way as one transforms programs in high school algebra”* — [12]. Whether such a system is possible for quantum algorithms remains open — although a step in this direction is [75], giving domain-theoretic analogues of differential equations with both classical and quantum search as their solutions.

Acknowledgements

The author wishes to thank many individuals, and is very grateful for many discussions with : S. Abramsky and B. Coecke on the categorical foundations program and category theory generally, T. Altenkirch on quantum conditionals, iteration, and programming languages, S. Braunstein on both teleportation and the Choi-Jamiołkowski correspondence from a physicists perspective, and the software problem for quantum computers, V. Danos and R. Duncan for interest in an early draft of this paper, K. Martin on the behaviour of quantum algorithms from a domain theory point of view, P. Scott on logical and categorical interpretations, and connections with lambda calculus, and P. Selinger on functional programming and algebras of combining forms.

¹⁵ Side-effects and states are often considered essential for input / output, storage, exception-handling, &c. We refer to [97] for how such features are handled in functional programming using the very categorical idea of *monads*.

References

1. S. Abramsky (1991) Domain theory in logical form *Annals of Pure and Applied Logic* 51 1-77
2. S. Abramsky, R. Blute, P. Panangaden (1999) Nuclear and Trace Ideals in Tensorred *-categories, *Journal of Pure and Applied Algebra* 143(1) 3-47
3. S. Abramsky, E. Haghverdi, P. Scott (2002) Geometry of Interaction and Linear Combinatory Algebras *Mathematical Structures in Computer Science* 12(5) 625-665
4. S. Abramsky, B. Coecke (2004) A categorical semantics of quantum protocols *Proc. 19th Annual IEEE Symp. on Logic in Computer Science (LICS 2004)*, IEEE Computer Soc. Press, 415-425
5. S. Abramsky (2005) Abstract Scalars, Loops, and Free Traced and Strongly Compact Closed Categories, in *Proceedings of CALCO 2005*, Springer LNCS 3629, 1-31
6. S. Abramsky, R. Duncan (2006) A categorical quantum logic *Math. Struct. Comp. Sci.* 16 (3) 469-489
7. P. Aliferis, D Leung (2004) Computation by Measurements : A unifying picture *Physical Review A* (70) 062314
8. T. Altenkirch, J. Grattage (2005) A functional quantum programming language, *Proc. 20th Ann. IEEE Symp. LICS 2005*, 26-29 249 - 258
9. T. Altenkirch, J. Grattage, J. Vizzotto (2007) An Algebra of Pure Quantum Programming *3rd International Workshop on Quantum Programming Languages*, ENTCS 170 23-47
10. C. Anantharaman-Delaroche (1997) C* - algèbres de Cuntz- Krieger et groupes Fuchsien in *Operator Theory, Operator Algebras and Related Topics (Timisoara 1996)* The Theta Foundation, Bucharest 17-35
11. W. F. Aspray (1982) Pioneer Day '82: History of the Stored Program Concept, *Ann. Hist. Comp.* 4(4) 358-53.
12. J. Backus (1978) Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs *Communications of the ACM.* 21(8) 613-641
13. J. Backus (1981) Function Level Programs as Mathematical Objects *Functional Programming Languages and Computer Architecture archive*, *Proc. 1981 conference on Functional programming languages and computer architecture*, Portsmouth, New Hampshire 1-10
14. M. Barr, C. Wells (1999) *Category Theory for Computer Science*, 3rd ed., Centre de Recherches Mathématique, Montreal
15. C. Bennett (1973), Logical Reversibility of Computation, *IBM Journal of Research and Development* 17 525-532
16. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, W. K. Wootters (1993) Teleporting an Unknown Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels *Phys. Rev. Lett.* 70 1895-1899
17. G. Brassard, S Braunstein, R Cleve (1998) Teleportation as a Quantum Computation, *Physica D* 120 43-47
18. E. Bernstein, U. Vazirani (1997) Quantum complexity theory, *SIAM journal on Computing* 26 (5) 1411 - 1473
19. A. Brown, A. Page (1970) *Elements of functional analysis* Van Nostrand Reinhold Publishing, London

20. A. Burks (2003) *Who invented the computer? The legal battle that changed computing history* Prometheus Books
21. M. Choi (1975) Completely Positive Linear Maps on Complex Matrices, *Linear Algebra and its Applications* 285-290
22. B. Coecke (2005) Introducing Categories to the practising physicist (*Manuscript*) Available as : <http://web.comlab.ox.ac.uk/oucl/work/bob.coecke/Cats.pdf>
23. B. Coecke (2006) Kindergarten Quantum Mechanics – lecture notes, In: *Quantum Theory: Reconsiderations of the Foundations III*, AIP Press 8198
24. B. Coecke, D. Pavlovic : Quantum Measurements without Sums, in: *The Mathematics of Quantum Computation and Technology*; Chen, Kauffman and Lomonaco (eds.); Taylor and Francis (to appear)
25. B. Copeland (2002) The Church-Turing Thesis, in *The Stanford Encyclopaedia of Philosophy (Fall 2002 Edition)*, Edward N. Zalta (ed.) available as <http://plato.stanford.edu/archives/fall2002/entries/church-turing/>
26. V. Danos, E. Kashefi, P. Panangaden (2006) The one way to quantum computation M. Bugliesi et al. (Eds.): *ICALP 2006, Part II*, Springer LNCS 4052 1321
27. V. Danos, E. Kashefi, P. Panangaden (2007) The Measurement Calculus *quant-ph : 0704.1263v1*
28. C. G. Darwin (1927) Free Motion in the Wave Mechanics, *Proc. Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 117 (766) 258-293
29. D. Deutsch (1985), Quantum theory, the Church-Turing principle and the universal quantum computer *Proc. Royal Society of London, A* 400 97-117
30. R. Duncan (2007) *Types for Quantum Computation*, DPhil. Thesis, Oxford University
31. S. Eilenburg, G. Kelly (1966) Closed Categories *Proceedings of the Conference on Categorical Algebra (La Jolla 1965)*, Springer 421562
32. R. Feynman, R. Leighton, M. Sands (1965) *The Feynman Lectures on Physics, Vol. 3* Addison-Wesley, Reading, Mass.
33. R. Finkel (1996) *Advanced Programming Language Design*, Addison-Wesley
34. W. Fouche, J. Heidema, G. Jones, and P. Potgieter (2007) Halting in quantum Turing computation, in A. Adamatzky, B. De Lacy Costello, L. Bull, S. Stepney, C. Teuscher, (Editors), *Unconventional Computing 2007*, Luminer Press
35. S. Gay (2006) Quantum Programming Languages: Survey and Bibliography *Mathematical Structures in Computer Science* 16(4) 581-600
36. R. Geroch (1985) *Mathematical Physics*, University of Chicago press
37. J.-Y. Girard, (1987) Linear logic *Theoretical Computer Science* 50(1) 1-102
38. J.-Y. Girard, Y. Lafont, P. Taylor (1989) *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press
39. M. Godfrey, D. Hendry (1993) The computer as von Neumann planned it, *IEEE Annals of the History of Computing* 15(1) 11-21
40. D. Gottesman (1999) The Heisenberg representation of quantum computers, in S.P. Corney, R Delbourgo, P.D. Jarvis (eds), *Group22: Proceedings of XXII International Colloquium on Group Theoretic Methods in Physics* 32-43 Cambridge, MA. Extended version at [arXiv:quant-ph/9807006](http://arXiv.org/abs/quant-ph/9807006)
41. J. Grattage (2006) QML - a functional quantum programming language, *PhD Thesis, Univ. Nottingham*

42. J. Gruska (1999) *Quantum Computing*, McGraw-Hill
43. A. Hagar, A. Korolev 2004, What is *quantum* in quantum computing? A lesson from two halting problems, *ESF Workshop on The Logic of Quantum Information 2004*, available as <http://www.webalice.it/shakush/ST1.pdf>
44. R. Herken (editor) (1995) *The universal Turing machine : a half-century survey, 2nd ed.* Springer Verlag
45. P. Hines (1999) The categorical theory of self-similarity *Theory and Applications of Categories (6)* 33-46
46. P. Hines (2002) A short note on coherence and self-similarity *Journal of Pure and Applied Algebra 175(1)* 135-139
47. P. Hines (2003) A categorical framework for finite state machines *Mathematical Structures in Computer Science 13(3)* 451-480
48. P. Hines (2006) Physical Systems as Constructive Logics, *Unconventional Computation, Springer LNCS (4135)* 101-112
49. P. Hines (2006) Compact closed monoids – definitions and constructions *GeoCal, CIRM Marseille*
50. P. Hines : Machine Semantics (*submitted*) Available as <http://www.peterhines.net/downloads/papers/MachineSemantics.pdf>
51. P. Hines (2007) Towards a quantum machine semantics, *Mathematical Foundations of Programming Semantics (2007)*, invited talk
52. P. Hines (2008) From Causality to Computational Models *International Journal of Unconventional Computation 4(3)* 249-272
53. P. Hines, P. Scott, Conditional quantum iteration from categorical traces *Mathematical Structures in Computer Science (to appear)*
54. J. Hughes (1989) Why Functional Programming Matters *Computer Journal 32(2)* 98-107
55. N. Immerman (2002) Computability and Complexity, in *The Stanford Encyclopaedia of Philosophy (Fall 2006 Edition)*, Edward N. Zalta (ed.) available as <http://plato.stanford.edu/archives/fall2006/entries/computability/>
56. A. Jamiolkowski (1972) Linear transformations which preserve trace and positive semidefiniteness of operators, *Rep. Math. Phys. 3* 275-278
57. P. Johnstone (1986) *Stone Spaces*, Cambridge Studies in Advanced Mathematics 3, Cambridge University Press
58. P. Johnstone (2000) Review of, ‘Natural dualities for the working algebraist’, by D. Clarke, B. Davey, *American Mathematical Society (2000)*
59. A. Joyal, R. Street (1991) The geometry of tensor calculus I. *Advances in Mathematics, 88(1)* 55-112.
60. A. Joyal, R. Street, The geometry of tensor calculus II *manuscript*.
61. A. Joyal, R. Street, D. Verity (1996) Traced Monoidal categories, *Math. Proc. Camb. Phil. Soc.* 425-446
62. R. Jozsa, N. Linden (2003) On the rôle of entanglement in quantum speedup *Proceedings of the Royal Society, Mathematical Physical and Engineering Sciences, Vol. 459(2038)* 2011-2032
63. M. Kelly, M. Laplaza (1980) Coherence for Compact Closed Categories, *Journal of Pure and Applied Algebra 19* 193-213
64. P. C. Kocher (1996) Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. *Advances in Cryptology - CRYPTO 96, N. Koblitz, editors, Springer-Verlag, LNCS vol. 1109* 104-113

65. A. Kitaev, A. Shen, M. Vyalıy : *Classical and Quantum Computation*, Graduate Studies in Mathematics (47), American Mathematical Society, Providence.
66. R. Landauer (1961) Irreversibility and heat generation in the computing process, *IBM Journal of Research and Development* (5) 183-191
67. J. Lambek, P. Scott (1986) *An introduction to higher-order categorical logic*, Cambridge Studies in Advanced Mathematics 7, Cambridge University Press
68. M. Laplaza (1977) Coherence in non-monoidal closed categories *Transactions of the American Mathematical Society* (230) 293 - 311
69. M. Laplaza (1977) Embedding of closed categories into monoidal closed categories *Transactions of the American Mathematical Society* (233) 85-91
70. F. W. Lawvere (1963) Functorial Semantics of Algebraic Theories *Proc. Nat. Acad. Sci.* 50(5) 869-872
71. J. A. N. Lee (Feb. 1996) Looking Back, *IEEE Computer Magazine*
72. J. A. N. Lee (2002) John Louis von Neumann, The History of Computing Series (NSF project CDA-931261). available as <http://ei.cs.vt.edu/~history/VonNeumann.html>
73. N. Linden, L. Popsecu (1998), The halting problem for quantum computers, *arXiv:quant-ph:/9806054 v2* 1998
74. S. MacLane (1998), *Categories for the working mathematician*, 2nd ed. Springer
75. K. Martin (2008) Domain theory and Measurement *This volume*.
76. M. Hasegawa : On traced monoidal closed categories, *Mathematical Structures in Computer Science* (to appear)
77. T. Miyadera, M. Ohya (2002) On halting process of quantum Turing machine, *Open Systems & Information Dynamics* 12(3) 261-264
78. J. Myers (1997) Can a Universal Computer be Fully Quantum?, *Physical Review Letters* vol. 78 (9) 1823-1824
79. J. von Neumann (1945) First Draft of a Report on the EDVAC *U.S. Army report on Contract No. W-670-ORD-4926.* , University of Pennsylvania. Moore School of Electrical Engineering. See also [39].
80. M. Nielsen, I. Chuang (1997) Programmable quantum gate arrays, *Phys. Rev. Lett.* 79 (2) 321-324
81. Nielsen, Chuang (2000) *Quantum computation and quantum information* Cambridge Univ. Press
82. A. Pati, S. Braunstein (2000) Impossibility of deleting an unknown quantum state *Nature* 404 164-165
83. A. Pati, S. Braunstein (2003) Quantum deleting and signalling, *Physics Letters A* (315) 208-212
84. M Raginsky (2003) Radon-Nikodym Derivatives of Quantum Operations *arXiv:math-ph/0303056*
85. G. Renstall (1994) On logics without contraction, *PhD Thesis, University of Queensland*
86. G. Renstall (1999) *An introduction to sub-structural logics*, Routledge N.Y.
87. Yaoun Shi (2002) Both Toffoli and Controlled-NOT need a little help to do universal quantum computation *arXiv:quant-ph/0205115 v2*
88. D. Roeder (1974) Category theory applied to Pontryagin duality *Pacific Journal of Mathematics* 52(2) 519-527

89. R. Rojas (1997) Konrad Zuse's legacy *IEEE annals of the History of Computing* 19(2) 5-16
90. R. Rojas (1998) How to make Zuse's Z3 a universal computer *IEEE annals of the history of computing* 20 (3) 51-54
91. P. Selinger (2007) Dagger compact closed categories and completely positive maps, *Proc. 3rd International Workshop on Quantum Programming Languages, ENTCS 170* 139-163
92. M. C. Shum (1994) Tortile Tensor Categories *Journal of Pure and Applied Algebra* (93) 57-110
93. M. Sørensen, P. Urzyczyn (2006) *Lectures on the Curry-Howard isomorphism*, Studies in Logic and the Foundations of Mathematics (149), Abramsky, Artemov, Gabbay, Kechris, Pillay, Shore (ed.s), Elsevier
94. N. Stern (1981) *From ENIAC to UNIVAC: An Appraisal of the Eckert-Mauchly Computers*, Digital Press, Bedford MA
95. D. Swade (2001) The difference engine : Charles Babbage and the quest to build the first computer, *Penguin - Putnam*
96. A. Yu Vlasov (2005) Programmable Quantum Networks with Pure States, in *Computer Science and Quantum Computing, J. E. Stones (ed.) (Nova Science Publishers, Inc.)* 33-61
97. P. Wadler (1995) Monads for Functional Programming, in *Jearing & Meijer (ed.s) Advanced Functional Programming, Springer Verlag, LNCS 925*
98. W. Wootters, W. Zurek (1982) A Single Quantum Cannot be Cloned, *Nature* 299 802-803

Appendix A

We consider the conditions required for a state vector to correspond to a unitary map, and show that this is intimately connected with questions of entanglement.

Definition 14. Let H denote a complex Hilbert space with orthonormal (computational) basis $\{\mathbf{e}_i\}_{i=1..n}$. Then for each basis vector \mathbf{e}_i we define the **left-span** of \mathbf{e}_i to be the subspace of $H \otimes H$ generated by the basis vectors $\{\mathbf{e}_i \otimes \mathbf{e}_j\}_{j=1}^n$. Dually, we define the **right-span** of \mathbf{e}_j to be the subspace of $H \otimes H$ generated by the basis vector $\{\mathbf{e}_i \otimes \mathbf{e}_j\}_{i=1}^n$. We denote these spaces by $lSpan(\mathbf{e}_i)$ and $rSpan(\mathbf{e}_j)$ respectively.

Our claim is that the vectors that are the names of unitary maps are exactly those that are equidistant to the left span and the right span of each basis vector \mathbf{e}_i

Theorem 9. Let H denote a complex Hilbert space with orthonormal basis $\{\mathbf{e}_i\}_{i=1..N}$, and let $M : H \rightarrow H$ denote a linear map. Then the following two conditions are equivalent:

- (i) M is a unitary map.
- (ii) For each basis vector \mathbf{e}_i , the norm of the projection of $\lceil M \rceil$ onto either $lSpan(\mathbf{e}_i)$ or $rSpan(\mathbf{e}_i)$ is $\frac{1}{n}$.

Proof.

(i) \Rightarrow (ii) By definition of a unitary map, M satisfies

$$MM^* = I \text{ and } I = M^*M$$

(It is easier to state that $M^* = M^{-1}$. However, interesting and computationally important C^* algebras such as the Kuntz-Krieger algebras of [10] satisfy one-sided versions of these conditions, so we use them separately for future reference). Written in terms of matrix elements, we have that

$$(MM^*)_{ik} = \sum_{j=1}^n m_{ij} \overline{m_{kj}} = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases}$$

$$(M^*M)_{ik} = \sum_{j=1}^n \overline{m_{ji}} m_{jk} = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases}$$

These conditions can also be characterised as ‘the sum of the norms of the entries in each row is 1, as is the sum of the norms of the entries in each column’.

Moving to the name $\lceil M \rceil \in H \otimes H$, we have

$$\langle \lceil M \rceil | \lceil M \rceil \rangle = \frac{1}{\sqrt{n}} \frac{1}{\sqrt{n}} \left(\sum_{\alpha, \beta=1}^n \left(\sum_{i, j=1}^n \langle \overline{m_{\alpha\beta}}(\mathbf{e}_i \otimes \mathbf{e}_j) | m_{ij}(\mathbf{e}_i \otimes \mathbf{e}_j) \rangle \right) \right)$$

Using the Kronecker delta notation, $\langle \mathbf{e}_i \otimes \mathbf{e}_j | \mathbf{e}_\alpha \otimes \mathbf{e}_\beta \rangle = \delta_{\alpha i} \delta_{\beta j}$, so

$$\langle \lceil M \rceil | \lceil M \rceil \rangle = \frac{1}{n} \left(\sum_{\alpha, \beta, i, j=1}^n \overline{m_{\alpha\beta}} m_{ij} \delta_{\alpha i} \delta_{\beta j} \right)$$

Hence, by the condition imposed on the matrix elements by the unitarity requirement,

$$\langle \lceil M \rceil | \lceil M \rceil \rangle = \frac{1}{n} \sum_{\alpha, \beta=1}^n \overline{m_{\alpha\beta}} m_{\alpha\beta} = 1$$

So the unitarity of M implies that $\lceil M \rceil$ has norm 1.

For the next step, observe that we may isolate the individual m_{ij} by

$$m_{ij} = \sqrt{n} \cdot \langle \mathbf{e}_i \otimes \mathbf{e}_j | \lceil M \rceil \rangle$$

and so the first unitarity condition gives that

$$1 = \sum_{j=1}^n \frac{1}{\sqrt{n}} \langle \lceil M \rceil | \mathbf{e}_i \otimes \mathbf{e}_j \rangle \frac{1}{\sqrt{n}} \langle \mathbf{e}_i \otimes \mathbf{e}_j | \lceil M \rceil \rangle$$

Hence

$$\frac{1}{n} = \sum_{j=1}^n \langle \lceil M \rceil \mathbf{e}_i \otimes \mathbf{e}_j \rangle \langle \mathbf{e}_i \otimes \mathbf{e}_j | \lceil M \rceil \rangle$$

Using Dirac notation, for any orthonormal basis B the identity is given by $Id = \sum_{\mathbf{b} \in B} |\mathbf{b}\rangle \langle \mathbf{b}|$, and so the inner product of vectors ϕ, ψ may be written as $\langle \phi | \psi \rangle = \sum_{\mathbf{b} \in B} \langle \phi | \mathbf{b} \rangle \langle \mathbf{b} | \psi \rangle$. From the definition of the space $lSpan(\mathbf{e}_i)$ in terms of a basis set, we thus deduce that the projection of $\lceil M \rceil$ onto the space $lSpan(\mathbf{e}_i)$ has norm $\frac{1}{n}$.

The dual condition about the right spans $\{rSpan(\mathbf{e}_i)\}_{i=1}^n$ follows from the second unitarity condition.

(ii) \Rightarrow (i) Let $\psi = \lceil M \rceil$ satisfy the left and right span conditions. We may write these fully as

$$\frac{1}{n} = \sum_j \langle \psi | \mathbf{e}_i \otimes \mathbf{e}_j \rangle \langle \mathbf{e}_i \otimes \mathbf{e}_j | \psi \rangle$$

and

$$\frac{1}{n} = \sum_j \langle \psi | \mathbf{e}_i \otimes \mathbf{e}_j \rangle \langle \mathbf{e}_i \otimes \mathbf{e}_j | \psi \rangle$$

respectively. The definition of the naming operation $\lceil \cdot \rceil$ gives that

$$[M]_{ij} = \sqrt{n} \cdot \langle e_i \otimes e_j | \psi \rangle$$

and almost identical reasoning to above applied to the left span condition gives $\sum_{j=1}^n [M]_{ij} \cdot \overline{[M]_{ij}} = 1$. Similarly, the right span condition gives that $\sum_{i=1}^n [M]_{ij} \cdot \overline{[M]_{ij}} = 1$. From above, these are the two conditions required for unitarity, and hence our result follows. \square

Interpretation

Although the above is presented abstractly, a quantum computational interpretation is immediate: given a quantum register $r \in qByte$, and an observation on a single arbitrary qubit with respect to *any* orthonormal basis $\{\mathbf{b}_1, \mathbf{b}_2\}$, then r is the name of a unitary map exactly when the observation of r gives either \mathbf{b}_1 or \mathbf{b}_2 with equal probability.