

A CLASSICAL GROUPOID MODEL FOR QUANTUM NETWORKS

DAVID J. REUTTER^a AND JAMIE VICARY^{a,b}

e-mail address: david.reutter@cs.ox.ac.uk, j.o.vicary@bham.ac.uk

^a Department of Computer Science, University of Oxford

^b School of Computer Science, University of Birmingham

ABSTRACT. We give a mathematical analysis of a new type of classical computer network architecture, intended as a model of a new technology that has recently been proposed in industry. Our approach is based on groubits, generalizations of classical bits based on groupoids. This network architecture allows the direct execution of a number of protocols that are usually associated with quantum networks, including teleportation, dense coding and secure key distribution.

1. INTRODUCTION

Borrill and Karp have recently introduced the notion of *timeless network* [Bor16], a new paradigm for distributed communication currently under commercial development by Earth Computing¹. Inspired by their proposal, we introduce a new network architecture based on *groubits*—group-theoretical generalizations of classical bits, with similar behaviour to qubits in quantum information—and go on to show that groubits can be manipulated to achieve a wide range of surprising informatic tasks. We give a categorical syntax and semantics for groubits, and develop a graphical calculus to prove correctness of groubit protocols.

Groubits. A groubit is a computational device storing two ordinary bits (A_L, A_I) , a *logical bit* A_L and an *internal bit* A_I , and supporting the primitive operations **Init**, **Swap**, **Read**, **Write** and **Tick**. Some of these operations in turn make use of the procedure **Rand**, a function with no arguments which returns either 0 or 1 nondeterministically. We describe these procedures as follows. The **Init** operation takes no arguments, and creates a new groubit in the following state:

- **Init** = **(Rand, 0)**

Here and throughout, we intend that the **Rand** function is executed freshly each time. The **Swap** operation acts on a groubit, exchanging the logical and internal bits:

- **Swap** $(A_L, A_I) = (A_I, A_L)$

Conventional single bits $[B]$ can be stored in groubits, using the following read and write procedures:

Key words and phrases: Groupoids, Networks, Quantum, Semantics, Key distribution.

¹See <http://www.earthcomputing.io>.

- **Read** $(A_L, A_I) = [A_L]$
- **Write** $[B] = (B, \mathbf{Rand})$

The **Read** operation destroys a groubit and creates a conventional bit, while the **Write** operation destroys a conventional bit and creates a new groubit. By ‘destroy’, we mean that the corresponding structure is no longer available for interaction; of course, in a real-world implementation, it may not be physically destroyed, but rather have its informational content somehow rendered inaccessible. Pairs of groubits can also be connected by a *link*, enabling the **Tick** operation, where A and B label the two connected nodes, and \oplus is addition modulo 2:

- **Tick** $((A_L, A_I), (B_L, B_I)) = ((A_L, A_I \oplus B_L), (B_L, B_I \oplus A_L))$

Intuitively, for each node in the pair, we flip the internal bit just when the other node has logical bit equal to 1. Nodes can belong to multiple links, forming a graph topology.

Assumptions. We make some assumptions about these groubit operations.

- **Atomicity.** The operations **Init**, **Swap**, **Read**, **Write** and **Tick** are atomic; that is, they either succeed or have no effect at all, with the parties involved being aware which of these two possibilities has occurred.
- **Security.** The state of a node cannot be accessed, except via **Read**.

We emphasize that claims we make about the functionality of groubit networks—in particular, security properties—rest on the validity of these assumptions.² We suggest that these assumptions are within the realm of technological plausibility; for example, separation kernels [Zha15] are a well-developed technology for guaranteeing strong security properties of private memory states within embedded devices. Our focus here is on the logical properties of these devices, rather than on questions of implementation, so we do not discuss these aspects further. Note however that we do not assume that devices cannot fail; to satisfy the assumptions, it would be valid for a device to self-destruct if tampering was detected.

1.1. Significance. We claim that groubits have exotic properties making them interesting to study. In particular, they allow timeout-free atomic message routing (the origin of the term ‘timeless network’), and they have the ability to replicate a variety of quantum protocols.

Message routing. Linear chains of groubits allow message routing between nodes with guaranteed message atomicity, and without timeouts (see Section 3.1). We understand that developing this idea is the primary commercial interest of Earth Computing, with a focus on high-resilience network architectures for data centres; this is potentially significant, since the timeout properties of the standard TCP transport protocol [IAC99] can cause reliability issues in a data centre environment [AM15, Bor16].

Quantum behaviour. A range of quantum protocols—entanglement creation, teleportation, dense coding, and secure key distribution—can be implemented on a groubit network, almost without modification.

If groubit networks can be implemented at scale in the real world, this may prove technologically significant, given the possibility that quantum computers may within decades be able to break in polynomial time the RSA public-key encryption scheme which is currently technologically dominant [Ber]. Should this possibility be realized, it has been suggested that

²For quantum protocols such as quantum key distribution, security is derivable from the laws of physics [VV14]; this is not the case here.

quantum key distribution could be used as an alternative technology to enable long-range information–theoretically secure communication [ea14]; we suggest that key distribution running on a large-scale groubit network may be an alternative worth investigating.

Some points must be made completely clear. Information–theoretically secure key distribution is known to be impossible in a classical computation setting. Our claim that it can be implemented using networks of groubits rests on the atomicity and security assumptions given in Section 1, and will hold for any real-world implementation only to some approximation. Also, we do not claim that *all* quantum protocols or algorithms can be implemented on groubits; in particular, we expect no analogue of ‘quantum speedup’, and give no classical model for important procedures such as the Grover or Shor algorithms [NC09].

Nonetheless, for those quantum protocols that we claim can operate on a groubit network, we mean this in a strong sense. In Section 4.3, we present a quantization functor which gives a structure-preserving mapping from our setting into quantum theory, sending groubit protocols to quantum protocols, and sending a groubit to a Hadamard matrix [RV16, Vic12b]. In other words, groubits yield a local hidden variable model for the part of quantum theory in the image of this quantization functor.

1.2. Overview. The structure of this article is as follows. In Section 2, we give the definition of a groubit in terms of groupoids with extra structure. We define the 2-category \mathbf{GpdAct}_f of finite groupoids, free profunctors and spans, and in our central technical result, show that groubits correspond precisely to biunitary connections in \mathbf{GpdAct}_f ³. We also give a 2-dimensional graphical programming language for groubits, and give a thorough development of its syntax and semantics. In Section 3 we give programs for state transfer, entanglement creation, teleportation and dense coding on networks of groubits, and verify these protocols using the rules of our abstract 2-dimensional syntax. We comment on the potential applicability of these protocols for message transfer and key distribution within networks of groubits. Further technical details on \mathbf{GpdAct}_f and its quantization functor are given in Section 4.

1.3. Related work. Timeless networks. The concept of timeless networking and the possibility of timeout-free atomic message routing is due to Borrill and Karp [Bor16], who also described the quantum properties of the technology. Our treatment here is inspired in part by their ideas, but does not follow the technical details of their approach.

Spekkens’ toy model. A toy model for quantum phenomena has been developed by Spekkens and others [Spe07, CES11, BD15, DM16, Pus12] based on the *knowledge balance* principle, in which quantum-like effects arise by restricting an observer’s ability to gain information about the state of a classical system. This principle can be seen as playing a role here, since groubits exhibit precisely such a balance between observable and unobservable states. Work on the toy model includes classical versions of several quantum procedures, including teleportation and dense coding which we also analyze here; furthermore, the low-level combinatorics are strongly similar in places (compare for example [Spe07, Section I] with Figure 12 here.)

Our work goes beyond these results in a number of ways, including: identification of biunitary structures in \mathbf{GpdAct}_f as a mathematical foundation; classification of these structures in terms of groubits; applications to timeless networks, key distribution and state

³See Section 1.3 for background on biunitaries.

transfer; the 2-dimensional high-level language for designing and verifying groubit programs; and the identification of a functorial mapping from our calculus to quantum theory. Also, we have a fundamentally different perspective: while the work cited above studies the toy model as a ‘foil theory’—an exercise in quantum foundations which sheds light on the distinction between quantum and classical reality—our perspective is technological, focussed on writing and verifying programs for these hypothetical devices, which may be implementable and practically useful in the real world.

Groupoidification. Our work is close in spirit to the groupoidification programme developed by Baez, Morton and others [BD01, BHW10, Mor06, BLLR97] from the combinatorial species of Joyal [Joy81]; as here, they develop a 2-categorical groupoid-based model for quantum-like phenomena, equipped with a functorial mapping into traditional quantum theory. Yet there is a surprising disconnect: their work is based on groupoids, spans, and spans of spans, while ours is based on groupoids, free profunctors and spans. This technical distinction seems mild, yet is essential for our results, and we are not aware of a direct relationship between the settings.

Classical key distribution. Maurer [Mau93] has suggested classical procedures for secure key distribution based on noisy communication channels. In his words, he drops the “apparently innocent assumption that, except for the secret key, the enemy has access to precisely the same information as the legitimate receiver”. This is fundamentally different to our model, in which—just as in quantum key distribution—the “enemy” has access to the entire apparatus.

Biunitaries. Our main proof technique is the technology of *biunitaries* (see Section 2.3.) Introduced by Ocneanu [Ocn89] in 1989 and since developed by Jones, Morrison and others [Jon99, JMS13, MP14], they are a central tool in the classification of subfactors, a major research effort in pure mathematics. Biunitaries belong to the theory of *planar algebras*, which studies the linear representation theory of algebraic structures in the plane. The 2-dimensional syntax we use in this paper derives heavily from the work of this community. These planar algebra techniques have been used by the present authors and others [Vic12b, RV16, RV18, JLW16] to give a high-level language for quantum computation.

Unpublished work. Related ideas have been described by Bar and the second author in an unpublished note [BV14].

1.4. Acknowledgements. We are grateful to Paul Borrill and Alan Karp for conversations about timeless networking, Steve Vickers for many helpful comments on an earlier version of this paper, and to Krzysztof Bar for substantial discussions about groupoid semantics and a classical model for key distribution.

2. FOUNDATIONS

2.1. Grouditis and dits. Grouditis. We begin with the definition of a groudit. Here and throughout, given a groupoid \mathbf{G} with a chosen object a , we write $\text{Aut}_{\mathbf{G}}(a)$ for the set of morphisms in \mathbf{G} of type $a \rightarrow a$.

Definition 2.1. A *groudit* \mathcal{G} is a skeletal groupoid of the form $\mathbf{G} = \coprod_a G_a$, where G_a are finite groups, equipped for each $a \in \text{Ob}(\mathbf{G})$ with bijections $\epsilon_a, \tau_a : \text{Aut}_{\mathbf{G}}(a) \rightarrow \text{Ob}(\mathbf{G})$.

Thinking about the consequences of this definition, we see that the underlying groupoid of a groudit is a disjoint union of d finite groups for some $d \in \mathbb{N}$, each with d elements. Note that the bijection data is not required to satisfy any properties, so groudits are easy to construct.

Our main result of this section is Theorem 2.8, where we show that groudits classify biunitary structures in a 2-category of finite groupoids, free profunctors and invariant spans; in particular, this theorem shows how for every groudit we can obtain analogues of the **Swap** and **Tick** maps, which we preview here as follows, using a notation (a, b) where $a \in \text{Ob}(\mathbf{G})$ and $b : a \rightarrow a$:

$$\mathbf{Swap}_{\mathcal{G}}(a, b) := \left(\epsilon_a(b), \tau_{\epsilon_a(b)}^{-1}(a) \right) \tag{2.1}$$

$$\mathbf{Tick}_{\mathcal{G}}((a, b), (c, d)) := \left((a, b (\epsilon_a^{-1}(c))^{-1}), (c, \tau_c^{-1}(a)d) \right) \tag{2.2}$$

Much later, in Section 4.3, we show that groudits yield Hadamard matrices, quantum combinatorial structures of deep importance in quantum information; in this sense, groudits are the classical combinatorial analogues of Hadamard matrices.

Just as classical bits are special cases of dits, so groubits are special cases of groudits.

Definition 2.2. A *groubit* \mathcal{B} is the groudit with identity bijections, and with underlying groupoid \mathbf{B} defined as follows, where s, t are the source and target functions:

$$\text{Ob}(\mathbf{B}) := \mathbb{Z}_2 \quad \text{Mor}(\mathbf{B}) := \mathbb{Z}_2 \times \mathbb{Z}_2 \quad s, t := \mathbb{Z}_2 \times \mathbb{Z}_2 \xrightarrow{\pi_1} \mathbb{Z}_2 \tag{2.3}$$

Composition is defined as follows: $(a, b) \circ (a, c) := (a, b \oplus c)$.

So for $a, b \in \mathbb{Z}_2$, we write (a, b) to denote a morphism of type $a \rightarrow a$. Using the terminology of Section 1, we interpret a as the logical bit, and b as the internal bit. It follows from the composition law that the identity morphisms are of the form $(a, 0)$. For the bijection data, we exploit the fact that the groupoid is in a natural way the disjoint union of two copies of \mathbb{Z}_2 , and so the bijections have the type $\epsilon_i, \tau_i : G_i = \mathbb{Z}_2 \rightarrow \text{Ob}(B) = \mathbb{Z}_2$. We choose all 4 of these bijections to be the identity.

For every protocol we give in this paper, we describe an implementation for an arbitrary groudit, and prove correctness at this general level. However, for informal discussions of groudit phenomena, and for the explicit traces of each protocol that we give throughout Section 3, we talk in terms of groubits.

Dits. We can also describe classical dits using groupoids.

Definition 2.3. A *dit* \mathcal{D} is a discrete skeletal groupoid \mathbf{D} with d morphisms.

We recall that a groupoid is discrete when every morphism is an identity. For a dit \mathcal{D} , we write $[i]$ to denote a morphism $i \in \text{Mor}(\mathbf{D})$. An ordinary classical bit is a dit with $d = 2$.

States. A *state* of a groudit or dit is a morphism in the corresponding groupoid. Our dynamics are nondeterministic, so after a protocol, the final state of a system is in general



FIGURE 1. Notation for states of a groubit and a bit.

a multiset drawn from the set of states. We indicate these multisets with a sum notation, with coefficients drawn from \mathbb{N} .

In our graphical calculus, a groudit is represented by a blue region, and a classical dit by a yellow region. To indicate the state of the system at a given time, we draw a horizontal dashed line, and write the state to the left; see Figure 1.

Operations. In our graphical calculus we define *atomic operations*, and also *derived operations* which are built from the atomic operations. We summarize these here, and show explicitly how they act on groubits and bits. This notation is all that is required to follow the protocol traces illustrated in Section 3. In all our diagrams, time flows from bottom to top. All operations listed here map every input state to a nonempty multiset, meaning that they will not fail. That makes them suitable building blocks for a groudit programming language.

Atomic operations. In Figure 2 we list the atomic operations involving a bit and a groudit. Figure 2(a)–(c) shows the three groudit-only operations: **Swap** and **Tick** are deterministic, while **Init** creates a groudit in a nondeterministic logical state. Figure 2(c) uses a rotated and reflected letter to label the vertex, since it is represented algebraically by a rotated and reflected version of Figure 2(b) under the dagger pivotal structure (see the discussion in Section 2.2 – graphical calculus).

Note that the result of performing two successive **Tick** operations between neighboring parties Alice and Bob, and Bob and Charlie, does not depend on the order of the operations; there is no race condition. Using the expression in Figure 2(c) this becomes a simple isotopy, a crucial feature of our 2-dimensional graphical calculus.

Figure 2(d)–(e) represents nondeterministic generation and erasure of a classical bit. Figure 2(f)–(g) give the basic interactions between a groudit and a bit: **Read** depicts the read-out of the logical state of a groudit, and **Write** depicts the initialization of a groudit with given logical bit and random internal bit.

Derived operations. In Figure 3 we list the derived operations **IRead**, **IWrite**, **CTick** and **Split**. Note that **CTick** comes in both left and right versions, distinguished by their images. We will see how they are defined in terms of atomic operations later in this section.

2.2. Graphical calculus. Definition. Our graphical calculus represents groupoids, free actions and spans. We begin with an informal definition of the 2-category formed by these

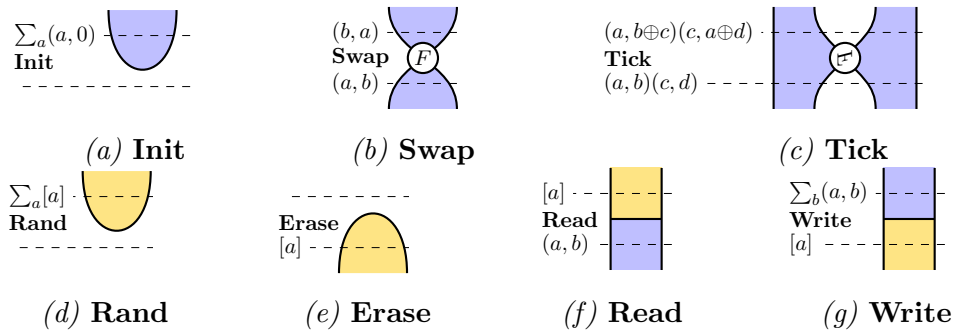


FIGURE 2. Atomic groudit and bit operations.

structures. Throughout, we write ‘2-category’ to refer to the weak structure, which is sometimes called ‘bicategory’. Similarly, we write ‘2-functor’ to refer to a weak 2-functor, which is sometimes called a ‘pseudofunctor’.

Definition 2.4. The 2-category \mathbf{GpdAct}_f is built from the following structures:

- *objects* are finite skeletal groupoids $\mathbf{G}, \mathbf{H}, \dots$;
- a *morphism* $S : \mathbf{G} \rightarrow \mathbf{H}$ comprises, for any $a \in \text{Ob}(\mathbf{G})$ and $b \in \text{Ob}(\mathbf{H})$, a finite set $S_{a,b}$ equipped with commuting free⁴ left- and right-actions of $\text{Aut}_{\mathbf{G}}(a)$ and $\text{Aut}_{\mathbf{H}}(b)$ respectively;
- for morphisms $S, T : \mathbf{G} \rightarrow \mathbf{H}$, a *2-morphism* $\sigma : S \Rightarrow T$ is an *equivariant span*, comprising for all $a \in \text{Ob}(\mathbf{G})$ and $b \in \text{Ob}(\mathbf{H})$ a function $\sigma_{a,b} : S_{a,b} \times T_{a,b} \rightarrow \mathbb{N}$, such that for all $g \in \text{Aut}_{\mathbf{G}}(a)$, $h \in \text{Aut}_{\mathbf{H}}(b)$, $s \in S_{a,b}$ and $t \in T_{a,b}$ we have $\sigma_{a,b}(s, t) = \sigma_{a,b}(g.s.h, g.t.h)$.

For the definition of composition, see Section 4.1.

Here $g.s.h \in S_{a,b}$ denotes the action on s by g on the left and h on the right; since these actions commute, this is well-defined. Note the requirement that these left- and right-actions are free, which is important for guaranteeing that our constructions are well-defined. In the main part of this paper we will work with these structures informally; we give a formal 2-categorical analysis in Section 4.

Definition 2.5. For an equivariant span $\sigma : S \Rightarrow T$, we define its *dagger* $\sigma^\dagger : T \Rightarrow S$ as the converse: $\sigma_{a,b}^\dagger(t, s) := \sigma_{a,b}(s, t)$.

Graphical calculus. We use a 2-dimensional graphical calculus (see Figure 4(a)) to denote a 2-morphism σ in \mathbf{GpdAct}_f . This is the standard graphical calculus for 2-categories [Sel10]: objects \mathbf{G}, \mathbf{H} label the regions, morphisms $S, T : \mathbf{G} \rightarrow \mathbf{H}$ label the wires, and 2-morphisms $\sigma : S \Rightarrow T$ label the vertices. We often drop the labels; also, white regions will always correspond to the trivial groupoid $\mathbf{1}$ with one morphism. Stacking these pictures vertically performs composition of spans, stacking them horizontally performs bimodule composition, and reflecting them about a horizontal axis corresponds to the dagger operation of Definition 2.5. In fact, \mathbf{GpdAct}_f is a *dagger pivotal 2-category* [CR16, Section 2.1], giving immense freedom in the graphical calculus: one may reflect, rotate and deform the pictures arbitrarily (holding the boundaries fixed), and these manipulations preserve equality of the diagram. For

⁴An action of a group H on a set X is *free*, if for any $x \in X$, $h.x = x$ implies that $h = e_H$.

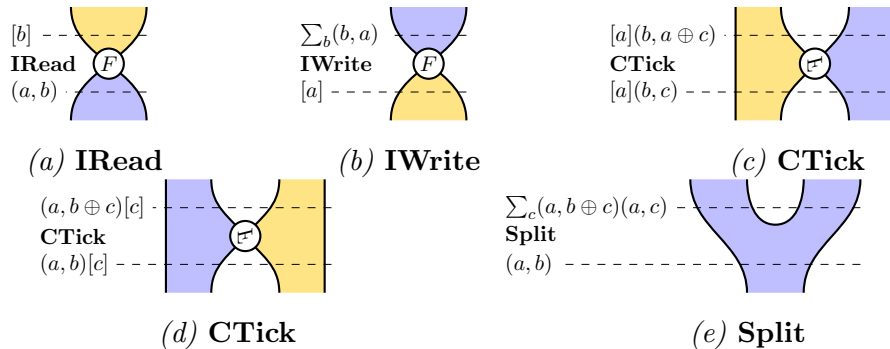




FIGURE 3. Derived groubit and bit operations.

example, since the images of Figure 4(a) and (b) are deformations of each other with constant boundary, they represent equal 2-morphisms.

Boundaries. For every shaded region labelled by a skeletal groupoid \mathbf{G} , we have canonical boundaries drawn as follows:



$L^{\mathbf{G}} : \mathbf{1} \rightarrow \mathbf{G}$



$R^{\mathbf{G}} : \mathbf{G} \rightarrow \mathbf{1}$

(2.4)

We define these as the following sets with free right (or left) $\text{Aut}_{\mathbf{G}}(a)$ -action, for all objects $a \in \mathbf{G}$:

$$L_{\bullet, a}^{\mathbf{G}} := \text{Aut}_{\mathbf{G}}(a) \quad (*, g, g') \mapsto gg' \quad (2.5)$$

$$R_{a, \bullet}^{\mathbf{G}} := \text{Aut}_{\mathbf{G}}(a) \quad (g', g, *) \mapsto g'g \quad (2.6)$$

Here, \bullet and $*$ denote the unique object and morphism of the terminal groupoid $\mathbf{1}$, respectively. That is, these boundaries are defined as the groupoid acting on itself, by left or right action. Using the pivotal structure, these boundaries give rise to the operations **Init**, **Erase** and **Split** as presented in Section 2.1.

2.3. Biunitaries. Biunitaries are important structures from the theory of planar algebras (see Section 1.3) which play an essential role in our calculus.

Definition 2.6. In \mathbf{GpdAct}_f , a biunitary on a skeletal groupoid \mathbf{G} is a unitary 2-morphism



(2.7)

fulfilling the equations depicted in Figure 5.

The source and target of the biunitary is the composite 1-morphism $R^{\mathbf{G}} \circ L^{\mathbf{G}} : \mathbf{1} \rightarrow \mathbf{1}$ which evaluates to the set $\text{Mor}(\mathbf{G})$ of morphisms of the skeletal groupoid. Concretely, therefore, a biunitary is an automorphism of $\text{Mor}(\mathbf{G})$ satisfying an algebraic condition; it plays the role of a generalized **Swap** map in our groupoid programming language. The following theorem determines this condition precisely.

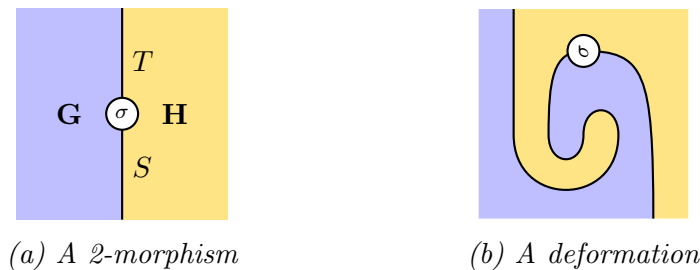


FIGURE 4. Examples of the graphical calculus.

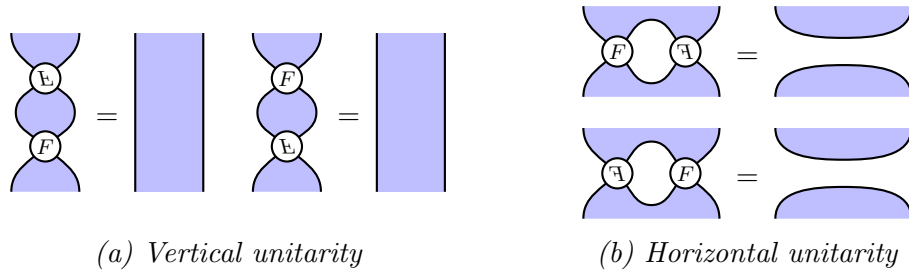


FIGURE 5. The biunitarity equations.

Theorem 2.7 . *A biunitary on a skeletal groupoid \mathbf{G} is a bijection $F : \text{Mor}(\mathbf{G}) \rightarrow \text{Mor}(\mathbf{G})$ such that for all $a, b \in \text{Ob}(\mathbf{G})$, we have:*

$$|F(\text{Aut}_{\mathbf{G}}(a)) \cap \text{Aut}_{\mathbf{G}}(b)| = 1 \tag{2.8}$$

Proof. The equations of Figure 5(a) say that F is unitary, which means precisely that it acts as a permutation on $\text{Mor}(\mathbf{G})$. The equations of Figure 5(b) are equivalent to the composite of Figure 6 being the identity. This holds just when, for all $a, b \in \text{Ob}(\mathbf{G})$ and for all $g \in \text{Aut}_{\mathbf{G}}(a)$ and $g' \in \text{Aut}_{\mathbf{G}}(b)$, there are unique $h \in \text{Aut}_{\mathbf{G}}(a)$, $r \in \text{Aut}_{\mathbf{G}}(b)$ with $s(F(h)) = b$ and $s(F^{-1}(r)) = a$ satisfying the following conditions:

$$gh^{-1}F^{-1}(r) = g \qquad r^{-1}F(h)g' = g'$$

In other words for any two objects a, b in the groupoid there is a unique pair $(h, r) \in \text{Aut}_{\mathbf{G}}(a) \times \text{Aut}_{\mathbf{G}}(b)$ such that $F(h) = r$. More concisely, $|F(\text{Aut}_{\mathbf{G}}(a)) \cap \text{Aut}_{\mathbf{G}}(b)| = 1$. \square

Classification. We now classify biunitaries in terms of groudits. This shows that biunitaries are tractable algebraic objects.

Theorem 2.8 . *For a skeletal groupoid \mathbf{G} , groudits on \mathbf{G} are in bijective correspondence with biunitaries on \mathbf{G} .*

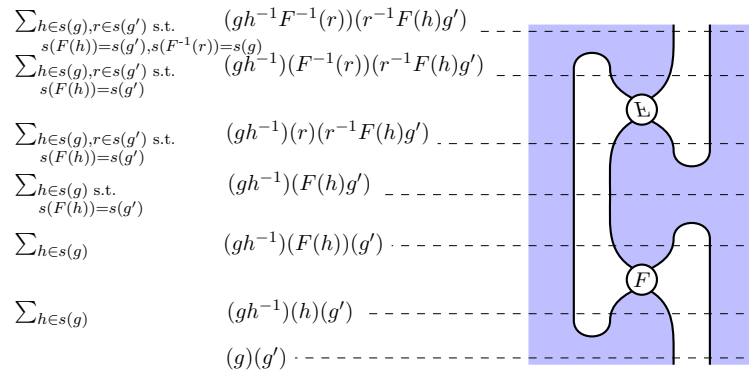


FIGURE 6. Verifying the action of a biunitary.

Proof. Define a *balancer* ϵ for \mathbf{G} to be a choice for all objects $a \in \text{Ob}(\mathbf{G})$ of a bijection $\epsilon_a : \text{Aut}_{\mathbf{G}}(a) \rightarrow \text{Ob}(\mathbf{G})$. Clearly for any $b \in \text{Ob}(\mathbf{G})$ we have

$$s(\epsilon_a^{-1}(b)) = a. \tag{2.9}$$

It is easy to see from the definition that a groudit is precisely a skeletal groupoid equipped with a pair of balancers. Given a balancer ϵ , we define functions ϵ_1, ϵ_2 as follows:

$$\epsilon_1 : \text{Mor}(\mathbf{G}) \rightarrow \text{Ob}(\mathbf{G}) \times \text{Ob}(\mathbf{G}) \quad \epsilon_1(g) := (sg, \epsilon_{sg}(g)) \tag{2.10}$$

$$\epsilon_2 : \text{Ob}(\mathbf{G}) \times \text{Ob}(\mathbf{G}) \rightarrow \text{Mor}(\mathbf{G}) \quad \epsilon_2(a, b) := \epsilon_a^{-1}(b) \tag{2.11}$$

We can show that ϵ_1 and ϵ_2 are inverse:

$$\epsilon_1(\epsilon_2(a, b)) = (s(\epsilon_a^{-1}(b)), \epsilon_{s(\epsilon_a^{-1}(b))}(\epsilon_a^{-1}(b))) \stackrel{(2.9)}{=} (a, \epsilon_a(\epsilon_a^{-1}(b))) = (a, b)$$

$$\epsilon_2(\epsilon_1(g)) = \epsilon_{s(g)}^{-1}(\epsilon_{s(g)}(g)) = g$$

We now give the first direction of the main bijective correspondence. Suppose ϵ, τ are balancers for \mathbf{G} . Then we define a biunitary $F_{\epsilon, \tau} : \text{Mor}(\mathbf{G}) \rightarrow \text{Mor}(\mathbf{G})$ as the following composite, where γ is the swap map for the cartesian product:

$$\text{Mor}(\mathbf{G}) \xrightarrow{\epsilon_1} \text{Ob}(\mathbf{G}) \times \text{Ob}(\mathbf{G}) \xrightarrow{\gamma} \text{Ob}(\mathbf{G}) \times \text{Ob}(\mathbf{G}) \xrightarrow{\epsilon_2} \text{Mor}(\mathbf{G})$$

Then a simple calculation shows the following:

$$F_{\epsilon, \tau}(g) = \tau_{\epsilon_{s(g)}(g)}^{-1}(s(g)) \in \text{Aut}_{\mathbf{G}}(\epsilon_{s(g)}(g)) \tag{2.12}$$

By construction, $F_{\epsilon, \tau}$ is unitary, since it is a composite of bijections. To show it is biunitary, suppose now that $g, g' \in \text{Aut}_{\mathbf{G}}(a)$ such that $s(F_{\epsilon, \tau}(g)) = s(F_{\epsilon, \tau}(g'))$. Then by equation (2.9), we have $\epsilon_{s(g)}(g) = \epsilon_{s(g')}(g')$, and since $s(g) = s(g') = a$ we therefore have $\epsilon_a(g) = \epsilon_a(g')$, and since ϵ_a is a bijection we have $g = g'$.

We now give the reverse direction of the main bijective correspondence. Given a biunitary $F : \text{Mor}(\mathbf{G}) \rightarrow \text{Mor}(\mathbf{G})$, we define balancers ϵ^F, τ^F for all $a \in \text{Ob}(\mathbf{G})$ and $g \in \text{Mor}(\mathbf{G})$ as

$$\epsilon_a^F(g) := s(F(g)) \quad \tau_a^F(g) := s(F^{-1}(g)) \tag{2.13}$$

We must show that for all $a \in \text{Ob}(\mathbf{G})$, $\epsilon_a^F, \tau_a^F : \text{Aut}_{\mathbf{G}}(a) \rightarrow \text{Ob}(\mathbf{G})$ are bijections. First, surjectivity. For any $b \in \text{Ob}(\mathbf{G})$, using the biunitarity property (2.8), pick the unique morphism $g \in F(\text{Aut}_{\mathbf{G}}(a)) \cap \text{Aut}_{\mathbf{G}}(b)$. Then $F^{-1}(g) \in \text{Aut}_{\mathbf{G}}(a)$ and $b = s(g) = s(F(F^{-1}(g))) = \epsilon_a^F(F^{-1}g)$. A similar proof shows surjectivity of τ_a^F . Next, injectivity. Suppose that $g, h \in \text{Aut}_{\mathbf{G}}(a)$ with $\epsilon_a^F(g) = \epsilon_a^F(h)$; then $s(F(g)) = s(F(h))$. Then $F(g), F(h) \in F(\text{Aut}_{\mathbf{G}}(a)) \cap$

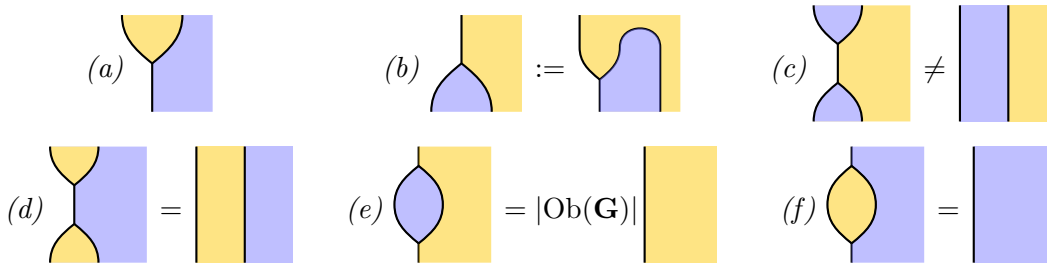


FIGURE 7. Building blocks for the measurement calculus.

$\text{Aut}_{\mathbf{G}}(s(F(g)))$. Then by the biunitarity property (2.8), we conclude that $F(g) = F(h)$ and therefore that $g = h$.

Finally, we show that the main correspondence is indeed bijective. In one direction, for a pair of balancers (ϵ, τ) with associated biunitary $F_{\epsilon, \tau}$ and $g \in \text{Aut}_{\mathbf{G}}(a)$, then by (2.9) we have $\epsilon_a^{F_{\epsilon, \tau}}(g) = s(F_{\epsilon, \tau}(g)) = \epsilon_a(g)$ and similarly $\tau_a^{F_{\epsilon, \tau}}(g) = s(F_{\epsilon, \tau}^{-1}(g)) = \tau_a^F(g)$. In the other direction, given a biunitary F and $g \in \text{Aut}_{\mathbf{G}}(a)$, we observe that $F_{\epsilon^F, \tau^F}(g) = (\tau_{\epsilon_a^F}^F(g))^{-1}(a)$. To show that this equals $F(g)$, we have to show that $\tau_{\epsilon_a^F}^F(g)(F(g)) = a$. And indeed, we have $\tau_{\epsilon_a^F}^F(g)(F(g)) = s(F^{-1}(F(g))) = s(g) = a$. \square

2.4. Measurements. In Section 2.1 we described classical dits using discrete groupoids. In the graphical calculus we draw them as yellow regions, to distinguish them from groudits which we draw in blue. There is an important difference: while blue regions are equipped with a biunitary of the form (2.7), yellow regions are not equipped with any such structure.

Every groudit has its associated dit, with the logical states of the groudit corresponding to the elements of the dit. They interact via the 2-morphisms depicted in Figure 7(a) and (b). These are not physical elements of the groudit programming language (explaining why they do not appear in Section 2.1), but auxiliary mathematical structures that we will use

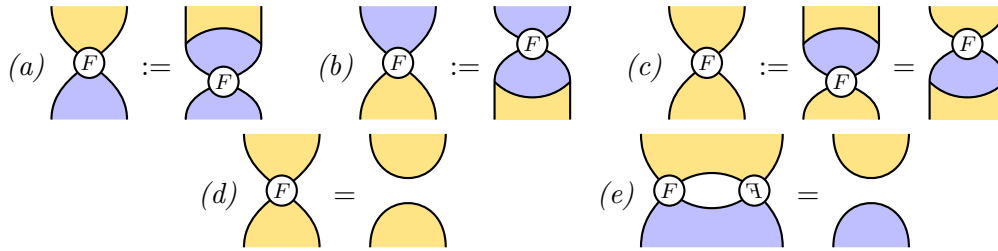


FIGURE 8. Yellow-blue and yellow-yellow versions of the biunitary.

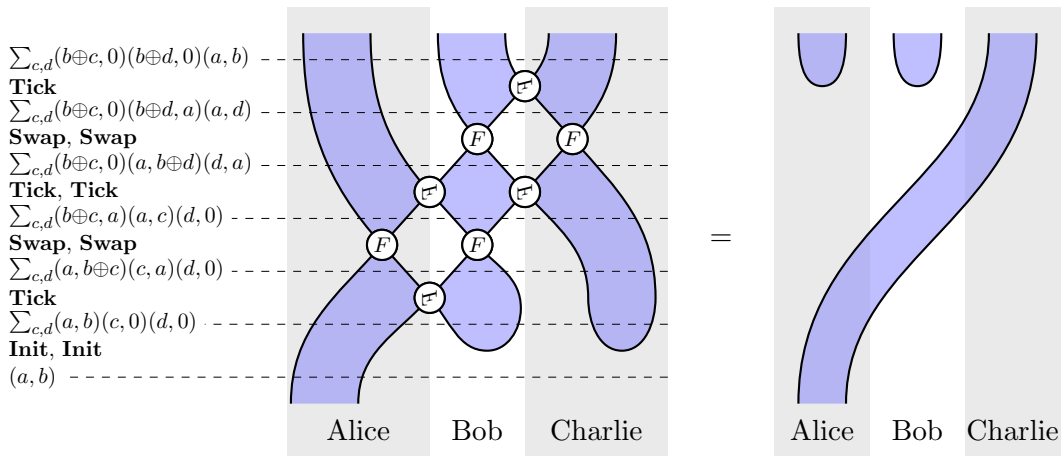


FIGURE 9. State transfer.

to verify our groudit programs. In Figure 7(a) we begin with a groudit, and we read it to extract some classical data indicated by the yellow region; the groudit itself still exists.

Semantically, the blue region represents a groudit \mathcal{G} with underlying groupoid \mathbf{G} , and the yellow region represents a classical dit \mathcal{B} with underlying groupoid \mathbf{B} , such that \mathbf{B} is a discrete groupoid with the same set of objects as \mathbf{G} . We define the yellow-blue morphism $S : \mathbf{B} \rightarrow \mathbf{G}$ as follows, where the set \emptyset is the empty set, and where $S_{b,g}$ are equipped with the ‘empty action’ and the right action of $\text{Aut}_{\mathbf{G}}(g)$ on itself, respectively:


$$S_{b,g} := \begin{cases} \text{Aut}_{\mathbf{G}}(g) & \text{if } b = g \\ \emptyset & \text{otherwise} \end{cases} \quad (2.14)$$

The blue-yellow morphism $S^* : \mathbf{G} \rightarrow \mathbf{B}$ is defined similarly. We define Figure 7(a) as follows, for all $a \in \text{Ob}(\mathbf{G}) = \text{Ob}(\mathbf{B})$ and $g \in \text{Aut}_{\mathbf{G}}(a)$:

$$\text{Figure 7(a)} \qquad g \mapsto (a, g) \quad (2.15)$$

The span (2.15) is unitary, and hence satisfies equations Figure 7(d) and (f); equation Figure 7(e) can be verified analogously. By way of warning, Figure 7(c) shows a *nonequation* that is not satisfied in general.

Given the topological behaviour encoded in Figure 7(b), we can be relaxed about how we draw the interface between yellow and blue regions:



$$(2.16)$$

This gives us our composite **Write** operation; **Read** is the dagger of this. We also use this to define yellow-blue and yellow-yellow versions of the biunitary in Figure 8(a)–(c).

Proposition 2.9. *Equation Figure 8(d) is fulfilled by the biunitary associated to any groudit.*

Proof. The composite 2-morphism on the left hand side of Figure 8(d) maps a dit $[a]$ to the multiset $\sum_{x \in \text{Aut}_{\mathbf{G}}(a)} [s(F_{\epsilon, \tau}(x))]$, where $F_{\epsilon, \tau}$ is the function (2.12) and s denotes the source of the morphism $F_{\epsilon, \tau}(x)$. It follows from the explicit expression (2.12) that $s(F_{\epsilon, \tau}(x)) = \epsilon_a(x)$. Since $\epsilon_a : \text{Aut}_{\mathbf{G}}(a) \rightarrow \text{Ob}(\mathbf{G})$ is a bijection, $[a]$ is mapped to the multiset $\sum_{x \in \text{Aut}_{\mathbf{G}}(a)} [\epsilon_a(x)] = \sum_{b \in \text{Ob}(\mathbf{G})} [b]$ which is the right hand side of Figure 8(d). \square

Equation Figure 8(e) is a direct consequence of horizontal unitarity of F . These structures yield our composite operations **IRead**, **IWrite**, **LRead**, **RRead** and **CTick**.

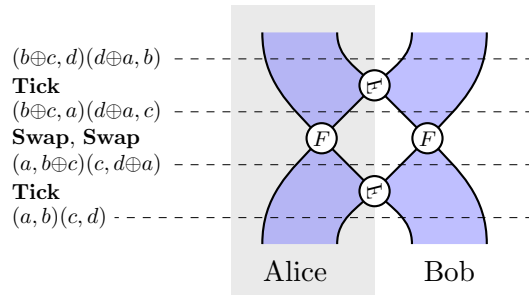


FIGURE 10. The basic state transfer repeating block.

3. PROTOCOLS

3.1. State transfer (Figures 9 and 10). Overview. The state transfer protocol communicates a groubit down a linear chain of nodes, such that each node is connected to its neighbour with a link. Our mathematical treatment is closely related to a state transfer protocol for cluster-based quantum computers proposed previously by the authors [RV18]. The adjective *timeless* arises from a specific property of this protocol, which we examine below.

Program. The state transfer program is illustrated in Figure 9(b) for three parties, Alice, Bob and Charlie, arranged in a linear chain. Each party has a node, and separate links connect Alice and Bob, and also Bob and Charlie, enabling **Tick** operations between connected parties. Alice has a groubit, which she would like to transfer to Charlie coherently; that is, preserving the internal state. The protocol is formed from repetitions of the *basic scheme* (see Figure 10), involving a **Tick** operation, two **Swap** operations, and a final **Tick**. In Figure 9(a) we use 2 copies of this basic building block, one between Alice and Bob, and one between Bob and Charlie. The generalization to arbitrary linear chains is clear.

Verification. The protocol is verified in the general case by observing that Figure 9(a) can be transformed into Figure 9(b) by applying the equations of Figure 5. On the left-hand side of Figure 9(a) we give an explicit program trace for the case of a $\mathbb{Z}_2 \sqcup \mathbb{Z}_2$ groubit, based on the lookup table in Section 2.1. The final state is $\sum_{c,d \in \mathbb{Z}_2} (b \oplus c, 0)(b \oplus d, 0)(a, b)$; by a simple change of variables it is clear that this equals $\sum_{c,d \in \mathbb{Z}_2} (c, 0)(d, 0)(a, b)$ as required.

Discussion. This protocol has certain limitations. While multiple messages can be sent from left-to-right along such a linear chain of nodes, if one attempts to send a message from right-to-left at the same time using a reflected version of the protocol, then both messages will be corrupted. Of course, this could be overcome by having a pair of parallel chains, keeping left-to-right and right-to-left communications on separate tracks. Furthermore, we do not have a clear analysis of communication on a network with a more interesting topology.

Timelessness. A key property of this protocol is that it makes no use of timeouts, thanks to atomicity properties that are part of our basic assumptions (see Section 1.) This is desirable, since timeouts are a basic feature of the dominant TCP protocol for internet communication [IAC99] which are the source of reliability issues in data centre environments [Bor16, AM15]. If the final **Tick** event of the scheme given in Figure 10 succeeds, then Bob assumes ownership of the message and continues to propagate it. Otherwise, if the final **Tick** is not successful—which could be because one of the 3 earlier events were not successful—Alice maintains ownership of the message, and is free to direct it by another route, or to return it as undeliverable to the sender.

3.2. Entanglement creation (Figure 11). Overview. This is a procedure to create an ‘entangled pair’ of groubits. Entangled groubits are required for the dense coding and teleportation protocols described later.

Program. Alice and Bob each initialize a groubit. They then perform a **Tick** operation involving both their groubits. Finally, Bob performs a **Swap** operation.

Verification. Immediate by Figure 5(a).

Discussion. To implement this protocol, Alice and Bob must be connected by a link enabling the **Tick** operation.

3.3. Dense coding (Figure 12). Overview. The dense coding procedure allows 2 classical bits to be transmitted between two parties, by transferring only 1 groubit. The parties must share an entangled pair of groubits, which could have been generated by the procedure discussed in Section 3.2.

Program. Alice begins with two classical bits, and Alice and Bob share an entangled pair of groubits. Alice begins by performing **CTick** operations (see Section 2.1) between her classical bits and her groubit, with a **Swap** operation in between. She then transfers the groubit to Bob, who performs a **Tick** operation between his two groubits, and then **IRead** operations on both groubits.

Verification. To verify correctness of the program for general groubits, substitute the definitions of **IRead** and **CTick** in terms of the basic syntax, then apply equations from Figure 5 to cancel 3 pairs of adjacent F nodes.

Discussion. It may seem surprising that dense coding is possible, since although a groubit has 2 classical bits of memory, they cannot both be directly accessed; applying the **Read** operation (see Section 2.1) reveals the logical bit, but destroys the internal bit. The program requires passing a groubit from one agent to another; to implement this, agents could use the state transfer program described in Section 3.1.

Dense coding allows agents connected by a groubit network to double their effective data transfer rate, at the expense of consuming shared entanglement. It may be possible to use this for temporal load-balancing in a groubit data center. During times of low utilization, agents in the network perform entanglement creation (Section 3.2) to generate substantial numbers of shared entangled groubits. Later, when utilization of the data centre becomes high, these entangled groubits can be consumed to double the effective rate of data transfer.

3.4. Teleportation (Figure 13). Overview. The teleportation procedure allows a groubit to be transported from one location to another, as long as those locations share an entangled groubit pair (see Section 3.2.)

Program. There are two parties, Alice and Bob. Alice starts with a groubit to be teleported, and Alice and Bob share between them an entangled pair of groubits. First, Alice performs a **Tick** operation on the groubit to be teleported. She then performs **Swap** operations on both of her groubits, then converts them into classical bits, which are transmitted to Bob

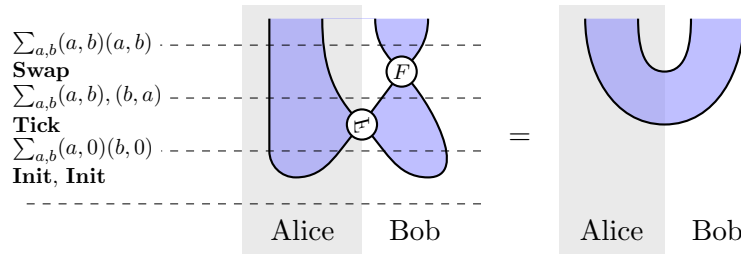


FIGURE 11. Entanglement creation.

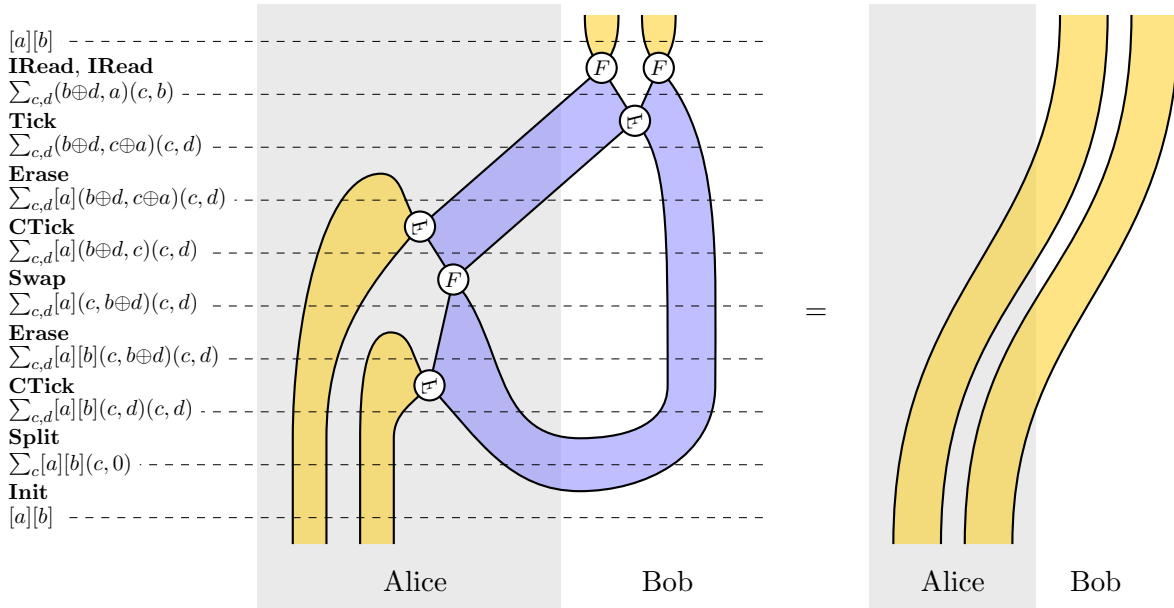


FIGURE 12. Dense coding.

by conventional means (for example, over the internet.) Bob then performs two **CTick** operations (see Figure 7), and performs **Erase** on the classical data received from Alice. The result is that Bob’s groubit is now in the same state as Alice’s was originally, both with respect to its logical and internal data.

Verification. To verify the protocol in the general case, expand the **CTick** operations using the definitions from Figure 8(a) and (b), then apply equation Figure 8(e) twice. The result is the identity, up to two yellow bubbles, which count the different classical bits that Alice could have obtained.

Discussion. Teleportation may have an application for transferring groubits between separate groubit networks, which may only be connected via the internet. Of course, these data centres would have to be furnished with a sufficient supply of entangled groubits.

3.5. Key distribution (Figures 14 and 15). Overview. Quantum key distribution (QKD) [ea14] is one of the most important protocols in quantum information. Here we describe a classical analogue which can operate on networks of groubits. The inability of the eavesdropper to read both the logical and internal state of a groubit is exploited to enable the effect. An analysis of QKD using a related graphical calculus has also been performed by Coecke and Perdrix [CP12]. We focus here on BB84-style QKD [BB85]; by dagger pivotality, the E91 variant [Eke91] has a similar analysis (see Figure 15.)

Program. The basic setup of our key distribution protocol is given in Figure 15(a), and is similar to the BB84 QKD protocol [BB85]. Alice and Bob have an authenticated public classical channel, and a groubit channel, which are both accessible by an adversary Eve. Alice begins with a classical bit, and chooses at random to encode it into a groubit using $\alpha = \mathbf{Write}$ or $\alpha = \mathbf{IWrite}$. She sends the groubit to Bob, perhaps using a state transfer

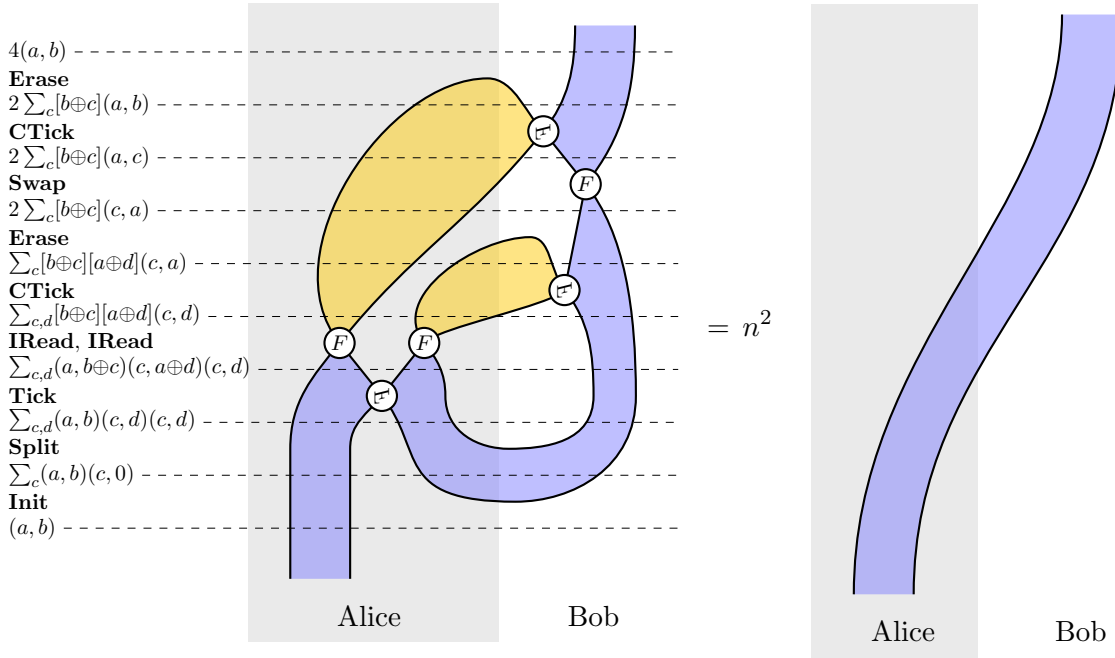


FIGURE 13. Teleportation.

algorithm (see Section 3.1), but it is intercepted by Eve, who chooses to decode the message using either $\eta = \mathbf{Read}$ or $\eta = \mathbf{IRead}$; having received a classical bit she copies it, and re-encodes a groubit using η^\dagger , which she sends to Bob. When Bob receives the groubit, he decodes it using $\beta = \mathbf{Read}$ or $\beta = \mathbf{IRead}$.

Verification. The analysis proceeds in just the same way as for the traditional BB84 procedure. If Alice, Bob and Eve all choose the same operation ($\alpha = \eta^\dagger = \beta^\dagger$), then it is as if Alice’s choice of initial bit is copied to Eve and Bob. We analyze this scenario in Figure 14(a), where we choose $\alpha = \eta^\dagger = \beta^\dagger = \mathbf{Write}$; using the equations of Figure 7, the equation can be verified. On the other hand, if any of the 3 parties do not choose the same

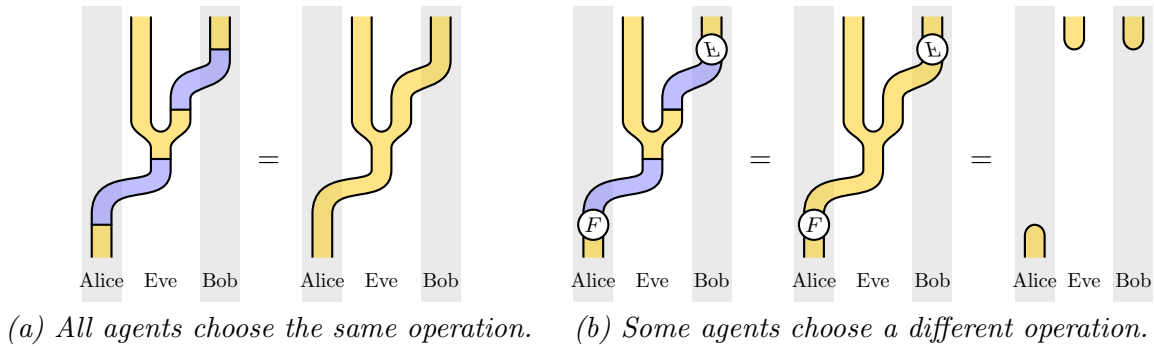


FIGURE 14. Verification of BB84 quantum key distribution.

operation, the diagram disconnects. We analyze this in Figure 14(b); using the equations of Figures 7 and 8, and in particular Figure 8(d), this chain of equalities can also be shown, leading us to conclude that all parties receive uncorrelated random bits. Although it matches the structure of the traditional quantum analysis, in this case it is incomplete, since we do not consider all possible actions by Eve; a deeper theory quantifying information flow through groubit networks would be needed to make a stronger statement.

Discussion. This protocol may have real-world relevance, either for key distribution within an insecure data centre based on groubit networks, or on a larger scale. Our analysis here cannot be considered a full security proof; just as with genuine quantum key distribution, there are many compounding details that would affect the real security of the procedure.

4. MATHEMATICAL FOUNDATIONS

4.1. Definition of \mathbf{GpdAct}_f . Here we give a precise definition of the 2-category \mathbf{GpdAct}_f . We assume some familiarity with the theory of 2-categories. For a good introduction see [Bor94, Chapter 7].

For any category \mathbf{C} , we write $\mathbb{N} : \mathbf{C} \rightarrow \mathbf{Set}$ for the constant functor that sends all morphisms to the identity on the set of natural numbers. Also, for any two functors $S, T : \mathbf{C} \rightarrow \mathbf{Set}$, we write $S \times T$ for their product in the functor category.

For a groupoid \mathbf{G} , we say that a functor $S : \mathbf{G} \rightarrow \mathbf{FinSet}$ is *free* if for any $f, f' : g \rightarrow g'$ for which there is an $x \in S(g)$ with $S(f)(x) = S(f')(x)$ it follows that $f = f'$. We say that a functor $S : \mathbf{H}^{\text{op}} \times \mathbf{G} \rightarrow \mathbf{FinSet}$ is *free* if both $S(h, -) : \mathbf{G} \rightarrow \mathbf{FinSet}$ and $S(-, g) : \mathbf{H}^{\text{op}} \rightarrow \mathbf{FinSet}$ are free for arbitrary objects $h \in \text{Ob}(\mathbf{H})$, $g \in \text{Ob}(\mathbf{G})$.

Definition 4.1. The 2-category \mathbf{GpdAct}_f is built from the following structures:

- **objects** are finite groupoids $\mathbf{G}, \mathbf{H}, \dots$;
- **morphisms** $S : \mathbf{G} \rightarrow \mathbf{H}$ are free functors $S : \mathbf{H}^{\text{op}} \times \mathbf{G} \rightarrow \mathbf{FinSet}$;
- **2-morphisms** $\sigma : S \Rightarrow T$ are natural transformations $\sigma : S \times T \Rightarrow \mathbb{N}$.

Composition of morphisms and 2-morphisms is defined below.

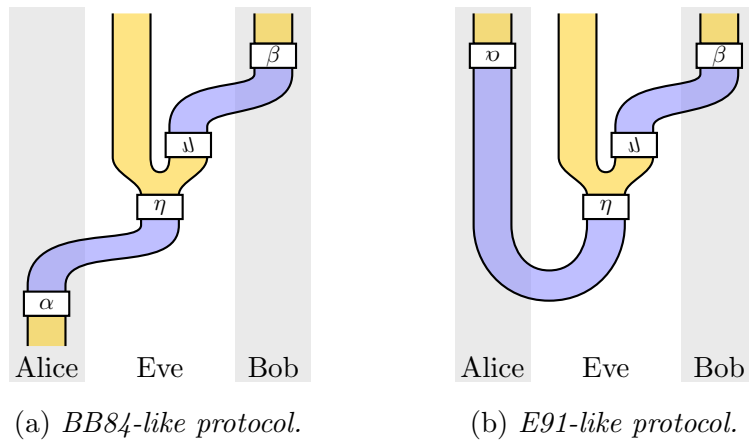


FIGURE 15. Key distribution protocols.

It is clear that this is equivalent to the restriction to skeletal groupoids, which for simplicity we use in the main body of the paper.

The morphisms of this 2-category are also known as *profunctors* or *distributors*, and there is a standard way to compose them [Bor94, Proposition 7.8.2] which we define below. The 2-morphisms can be considered as families of spans of sets⁵ which are compatible with the groupoid actions; we will refer to them as *protransformations*.

Spelled out explicitly, a profunctor $P : \mathbf{A} \rightarrow \mathbf{B}$ is a family of sets $(P_{b,a})_{b \in \text{ob}\mathbf{B}, a \in \text{ob}\mathbf{A}}$ equipped with functions

$$\begin{aligned} \text{Hom}_{\mathbf{A}}(a, a') \times P_{b,a} \times \text{Hom}_{\mathbf{B}}(b', b) &\rightarrow P_{b',a'} & (4.1) \\ (f, x, g) &\mapsto f.x.g \end{aligned}$$

for each $a, a' \in \text{ob}\mathbf{A}, b, b' \in \text{ob}\mathbf{B}$ such that for all $x \in P_{b,a}$ and all $f : a \rightarrow a', f' : a' \rightarrow a'', g : b' \rightarrow b, g' : b'' \rightarrow b'$, we have the following:

$$\text{id}_a.x.\text{id}_b = x \quad (f'f).x.(gg') = f'.(f.x.g).g'$$

A profunctor is *free* just when all groups $\text{Aut}_{\mathbf{A}}(a)$ and $\text{Aut}_{\mathbf{B}}(b)$ act freely on $P_{b,a}$. This structure induces obvious separate left and right actions on $P_{b,a}$ which we will also make use of.

A *protransformation* $\sigma : P \Rightarrow Q$ corresponds to a family of functions $P_{b,a} \times Q_{b,a} \xrightarrow{\sigma_{b,a}} \mathbb{N}$ such that the following holds, for all $p \in P_{b,a}, q \in Q_{b,a}, a \xrightarrow{f} a'$ and $b' \xrightarrow{g} b$:

$$\sigma_{b',a'}(f.p.g, f.q.g) = \sigma_{b,a}(p, q) \quad (4.2)$$

From now on we will omit the subscripts of protransformations.

Horizontal composition. The composite $Q \circ P$ of profunctors $P : \mathbf{A} \rightarrow \mathbf{B}$ and $Q : \mathbf{B} \rightarrow \mathbf{C}$ corresponds to the family of sets $(Q \circ P)_{c,a} := \coprod_{b \in \text{ob}\mathbf{B}} P_{b,a} \times Q_{c,b} / \sim$ with equivalence relation as follows, for any $x \in P_{b',a}, y \in Q_{c,b}, f : b \rightarrow b'$:

$$(x.f, y) \sim (x, f.y) \quad (4.3)$$

Given profunctors $S, S' : \mathbf{A} \rightarrow \mathbf{B}, T, T' : \mathbf{B} \rightarrow \mathbf{C}$ and protransformations $\sigma : S \Rightarrow S'$ and $\tau : T \Rightarrow T'$, the horizontal composite $\tau \circ \sigma : T \circ S \Rightarrow T' \circ S'$ corresponds to the family of functions

$$(\tau \circ \sigma)_{c,a} : \left(\coprod_b S_{b,a} \times T_{c,b} / \sim \right) \times \left(\coprod_{b'} S'_{b',a} \times T'_{c,b'} / \sim \right) \rightarrow \mathbb{N}$$

defined as follows, for $a \in \text{Ob}(\mathbf{A}), b, b' \in \text{Ob}(\mathbf{B}), c \in \text{ob}\mathbf{C}$ and $s \in S_{b,a}, t \in T_{c,b}, s' \in S'_{b',a}, t' \in T'_{c,b'}$:

$$\tau \circ \sigma([s, t], [s', t']) := \sum_{f: b \rightarrow b'} \sigma(s, s'.f) \tau(f.t, t') \quad (4.4)$$

Here $[s, t]$ and $[s', t']$ denote equivalence classes under (4.3). In the proof of Theorem 4.5 we show that this is well defined.

⁵If \mathbf{G} and \mathbf{H} are both the terminal groupoid $\mathbf{1}$, then morphisms $S, T : \mathbf{1} \rightarrow \mathbf{1}$ are finite sets and a natural transformation $\sigma : S \Rightarrow T$ is a function $\sigma : S \times T \rightarrow \mathbb{N}$, or equivalently a (bijection class of) *span of sets* $S \leftarrow \sqcup_{s \in S, t \in T} [\sigma(s, t)] \rightarrow T$, where $[\sigma(s, t)]$ is a finite set with $\sigma(s, t) \in \mathbb{N}$ elements. Analogously, a protransformation between profunctors $\sigma : S \Rightarrow T$ can be considered as a family of spans of sets $S_{g,h} \leftarrow X_{g,h} \rightarrow T_{g,h}$, indexed by $g \in \text{Ob}(\mathbf{G})$ and $h \in \text{Ob}(\mathbf{H})$, that is compatible with the actions of \mathbf{G} and \mathbf{H} .

Vertical composition. Given profunctors $P, Q, R : \mathbf{A} \rightarrow \mathbf{B}$ and protransformations $\sigma : P \Rightarrow Q$ and $\tau : Q \Rightarrow R$, the vertical composite $\tau\sigma$ corresponds to the family of functions $(\tau\sigma)_{b,a} : P_{b,a} \times R_{b,a} \rightarrow \mathbb{N}$ defined as follows, for $a \in \text{Ob}(\mathbf{A})$, $b \in \text{Ob}(\mathbf{B})$, $p \in P_{b,a}$ and $r \in R_{b,a}$:

$$\tau\sigma(p, r) := \sum_{q \in Q_{b,a}} \sigma(p, q)\tau(q, r) \quad (4.5)$$

Theorem 4.2 . \mathbf{GpdAct}_f is a 2-category.

For Theorem 4.2 to hold, the 1-morphisms of \mathbf{GpdAct}_f have to be *free* profunctors. Without the freeness condition, the horizontal and vertical composite of 2-morphisms (4.5) and (4.4) would not fulfill the interchange law. Indeed, for non-free profunctors, expression (4.4) overcounts, and would need to be normalized by a factor $\frac{1}{|\text{Stab}(s', t')|}$, where $\text{Stab}(s', t')$ denotes the stabilizer of $(s', t') \in S'_{b', a} \times T'_{c, b'}$ under the action of $\text{Hom}_{\mathbf{B}}(b, b')$. The resulting 2-category would extend \mathbf{GpdAct}_f to non-free profunctors, would still contain the 2-category of finite groupoids, profunctors and natural transformations as a subcategory, but would not admit a dagger structure anymore.

4.2. Dagger pivotality. We defined the *dagger* of a protransformation in Definition 2.5. If σ is interpreted as a computational process, then σ^\dagger is interpreted as its time-reversal.

Definition 4.3. For any profunctor $P : \mathbf{A} \rightarrow \mathbf{B}$, its *adjoint* is the profunctor $P^* : \mathbf{B} \rightarrow \mathbf{A}$ given by

$$\mathbf{A}^{op} \times \mathbf{B} \xrightarrow{\text{inv} \times \text{inv}} \mathbf{A} \times \mathbf{B}^{op} \xrightarrow{\text{swap}} \mathbf{B}^{op} \times \mathbf{A} \xrightarrow{P} \mathbf{FinSet} \quad (4.6)$$

where $\text{inv} : \mathbf{A}^{op} \rightarrow \mathbf{A}$ is the canonical isomorphism mapping every morphism to its inverse. In terms of bimodules, given a \mathbf{A} – \mathbf{B} bimodule $(P_{b,a})_{a,b}$, its adjoint is defined to be the \mathbf{B} – \mathbf{A} bimodule $(P_{b,a})_{b,a}$ with action $(f, x, g) \mapsto g^{-1}.x.f^{-1}$.

Definition 4.4. Given a profunctor $P : \mathbf{A} \rightarrow \mathbf{B}$, we define its *cap* to be the protransformation $\epsilon : P^* \circ P \Rightarrow \mathbb{1}_{\mathbf{A}}$ defined as follows, for $a, a' \in \text{Ob}(\mathbf{A})$, $b \in \text{Ob}(\mathbf{B})$, $p \in P_{b,a}$, $p' \in P_{b,a'}$, $f \in \text{Hom}(a', a)$:

$$\epsilon([p, p'], f) := \delta_{f^{-1}.p, p'} \quad (4.7)$$

We define its *cup* to be the protransformation $\eta : \mathbb{1}_{\mathbf{B}} \Rightarrow P \circ P^*$ defined as follows, for $a, a' \in \text{ob}\mathbf{A}$, $b \in \text{ob}\mathbf{B}$, $p' \in P_{a', b}$, $p \in P_{a, b}$ and $f \in \text{Hom}(a, a')$:

$$\eta(f, [p', p]) = \delta_{p', p.f^{-1}} \quad (4.8)$$

Theorem 4.5 . *The structures defined above yield a dagger pivotal structure [CR16, Section 2.1] for the 2-category \mathbf{GpdAct}_f .*

4.3. Quantization. Here we describe a *quantization* procedure, in the form of a dagger 2-functor

$$Q : \mathbf{GpdAct}_f \rightarrow \mathbf{2Hilb},$$

where $\mathbf{2Hilb}$ is a 2-category that is suitable for the description of quantum computational processes [Bae97, Vic12b, Vic12a, RV16]. For this quantization procedure to be functorial, it is essential that the group actions arising from the 1-morphisms of \mathbf{GpdAct}_f are free.

We take the perspective on higher vector spaces arising from algebras, bimodules and intertwiners [BBFW12, BDSPV15]. The 2-functor

$$\text{Rep} : \mathbf{C}^* \mathbf{Alg} \rightarrow \mathbf{2Hilb} \quad (4.9)$$

from the 2-category $\mathbf{C}^*\mathbf{Alg}$ of finite dimensional C^* -algebras, Hilbert bimodules and intertwiners [BMZ12] to the 2-category $\mathbf{2Hilb}$ of finite dimensional 2-Hilbert spaces, linear functors and natural transformations is a dagger equivalence.

In the following, we will construct a quantization 2-functor $Q : \mathbf{GpdAct}_f \rightarrow \mathbf{C}^*\mathbf{Alg}$.

Definition 4.6. Given a finite groupoid \mathbf{G} , its groupoid algebra (or convolution algebra) $\mathbb{C}\mathbf{G}$ is the vector space freely generated by all morphisms in \mathbf{G} with algebra structure $f \star g := fg$ when f, g are composable, and 0 otherwise, and $f^* := f^{-1}$.

The groupoid algebra $\mathbb{C}\mathbf{G}$ generalizes the notion of a group algebra - for a skeletal groupoid \mathbf{G} it is $\bigoplus_{a \in \text{ob}\mathbf{G}} \text{Aut}(a)$. In general, groupoid algebras are finite direct sums of tensor products of group algebras and matrix algebras. In particular, they are finite dimensional C^* -algebras.

Definition 4.7. The *quantization* dagger 2-functor

$$Q : \mathbf{GpdAct}_f \rightarrow \mathbf{C}^*\mathbf{Alg}$$

is defined as follows:

- on objects, for a groupoid \mathbf{A} , we have $Q(\mathbf{A}) := \mathbb{C}\mathbf{A}$;
- on morphisms, for a profunctor $P : \mathbf{A} \leftrightarrow \mathbf{B}$, we define $Q(P)$ to be the bimodule $\bigoplus_{b \in \text{ob}\mathbf{B}, a \in \text{ob}\mathbf{A}} \mathbb{C}P_{b,a}$;
- on 2-morphisms, for a protransformation $\sigma : P \Rightarrow Q$, we define $Q(\sigma)$ as the intertwiner

$$\bigoplus_{b,a} \mathbb{C}P_{b,a} \rightarrow \bigoplus_{b',a'} \mathbb{C}Q_{b',a'}$$

defined at stage (b, a) as the linear map extending

$$Q(\sigma)_{b,a}(p) = \sum_{q \in Q_{b,a}} \sigma(p, q)q \quad p \in P_{b,a}.$$

In particular, composing this functor with the equivalence (4.9) we obtain a 2-functor

$$Q' : \mathbf{GpdAct}_f \rightarrow \mathbf{2Hilb}$$

mapping groupoids \mathbf{G} to their representation categories $\text{Rep}(\mathbb{C}\mathbf{G}) \cong \text{Rep}(\mathbf{G}) = [\mathbf{G}, \mathbf{Hilb}]$.

Theorem 4.8 . *Definition 4.7 defines a dagger 2-functor.*

We can use these quantization results to explain the connection of our work to *Hadamard matrices*⁶: under the image of the quantization functor Q' , every groupoid yields a Hadamard matrix. To show this, we assume some knowledge of the 2-category $\mathbf{2Hilb}$, as can be found in the references at the start of this section.

In this 2-category, a 1-morphism can be regarded as a matrix of Hilbert spaces, and we call a 1-morphism *nondegenerate* when all these Hilbert spaces are 1-dimensional. It was first shown by Jones [Jon89], and explored in more detail by the second author [Vic12b, Theorem 4.6], that when restricting to nondegenerate 1-morphisms, and where the blank region labels the 1-dimensional 2-Hilbert space, biunitaries of the form (2.7) correspond exactly to Hadamard matrices. When the groupoid \mathbf{G} is abelian (that is, a union of abelian groups), it can be checked that the boundaries (2.4) become nondegenerate 1-morphisms under the action of the quantization functor Q' . Furthermore, it is clear that dagger 2-functors preserve biunitary structures, since biunitarity is an equational property involving composition and the dagger.

⁶A Hadamard matrix is a unitary matrix for which every coefficient has the same absolute value; such a matrix contains the same information as a pair of mutually unbiased bases.

With these pieces in place, the result follows: for any groubit defined over an abelian groupoid \mathbf{G} , its image in $\mathbf{2Hilb}$ under the quantization functor Q' yields a Hadamard matrix, acting on a Hilbert space given by the groupoid algebra of \mathbf{G} . For example, for the groubit described in Definition 2.2, this yields the following Hadamard matrix:

$$Q'(\mathbf{Swap}) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \quad (4.10)$$

This operates on a 4-dimensional Hilbert space, the groupoid algebra of $\mathbb{Z}_2 + \mathbb{Z}_2$, and the matrix is written in the character basis.

APPENDIX A. OMITTED PROOFS

Theorem 4.2 . \mathbf{GpdAct}_f is a 2-category.

Proof. We first observe that a natural transformation $\alpha : P \Rightarrow Q$ corresponds to a family of functions $P_{b,a} \xrightarrow{\alpha_{b,a}} Q_{b,a}$ such that

$$\alpha_{b',a'}(f.p.g) = f.\alpha_{b,a}(p).g$$

for all $p \in P_{b,a}$ and $a \xrightarrow{f} a', b' \xrightarrow{g} b$. Every natural transformation α induces a protransformation $\alpha_*(p, q) = \delta_{\alpha(p), q}$.

Vertical composition. Vertical composition (4.5) is clearly associative with unit $1_P(x, y) = \delta_{x,y}$, and it extends the vertical composition of natural transformations.

Horizontal composition. *Claim.* The composite of free profunctors is free.

Proof. The composite of free profunctors $P : \mathbf{A} \leftrightarrow \mathbf{B}$ and $Q : \mathbf{B} \leftrightarrow \mathbf{C}$ is given by the bimodule $\coprod_{b \in \text{ob} \mathbf{B}} P_{b,a} \times Q_{c,b} / \sim$ quotiented by (4.3) with the obvious left and right action. Let $[p, q]$ be an equivalence class in this set and let $f \in \text{Aut}(a)$ with $f.[p, q] = [p, q]$ or equivalently $(f.p, q) \sim (p, q)$. Thus, there is a $g \in \text{Aut}(b)$ such that $(f.p, q) = (p.g, g^{-1}.q)$. Since $\text{Aut}(b)$ acts freely on $Q_{c,b}$ it follows that $g = 1_b$ and thus that $f.p = p$. Since $\text{Aut}(a)$ acts freely on $P_{b,a}$ it follows that $f = 1_a$. In particular, $\text{Aut}(a)$ acts freely on $(Q \circ P)_{c,a}$. The proof that $\text{Aut}(c)$ acts freely is analogous.

Claim. Expression (4.4) is well defined.

Proof. We show that (4.4) descends to a map on the quotient. Let $\tilde{b} \xrightarrow{g} b$ and $\tilde{b}' \xrightarrow{g'} b'$ be arbitrary morphisms. Then

$$\begin{aligned} & \tau \circ \sigma([s.g, g^{-1}.t], [s'.g', g'^{-1}.t']) \\ & \stackrel{(4.4)}{=} \sum_{\tilde{b} \xrightarrow{\tilde{f}} \tilde{b}'} \sigma(s.g, s'.g' \tilde{f}) \tau(\tilde{f} g^{-1}.t, g'^{-1}.t') \\ & \stackrel{(4.2)}{=} \sum_{\tilde{b} \xrightarrow{\tilde{f}} \tilde{b}'} \sigma(s, s'.g' \tilde{f} g^{-1}) \tau(g' \tilde{f} g^{-1}.t, t') \\ & = \sum_{b \xrightarrow{f} b'} \sigma(s, s'.f) \tau(f.t, t') = \tau \circ \sigma([s, t], [s', t']). \end{aligned}$$

Horizontal composition of protransformations extends the horizontal composition of natural transformations.

The interchange law. We show that composition of morphisms is functorial. Clearly, $1_T \circ 1_S = 1_{T \circ S}$. *Claim.* For protransformations

$$\begin{array}{ccccc}
 & S & & T & \\
 & \downarrow & & \downarrow & \\
 \text{A} & \xrightarrow{S'} & \text{B} & \xrightarrow{T'} & \text{C} \\
 & \downarrow & & \downarrow & \\
 & S'' & & T'' &
 \end{array}$$

it holds that $(\nu \circ \tau)(\mu \circ \sigma) = (\nu\mu) \circ (\tau\sigma)$.

Proof. For a class $[s', t'] \in (T' \circ S')_{c,a}$ we define

$$\begin{aligned}
 \langle s', t' \rangle_b &= \{(\tilde{s}, \tilde{t}) \in S'_{b,a} \times T'_{c,b} \mid (\tilde{s}, \tilde{t}) \sim (s', t')\} \\
 &= \{(\tilde{s}, \tilde{t}) \mid \exists b' \xrightarrow{f} b : (\tilde{s}.f, f^{-1}.\tilde{t}) = (s', t')\} \\
 &= [s', t'] \cap (S'_{b,a} \times T'_{c,b})
 \end{aligned} \tag{A.1}$$

By the orbit-stabilizer theorem and the fact that our profunctors act freely we can express horizontal composition (4.4) as follows:

$$\tau \circ \sigma([s, t], [s', t']) := \sum_{(\tilde{s}, \tilde{t}) \in \langle s', t' \rangle_b} \sigma(s, \tilde{s}) \tau(t, \tilde{t}). \tag{A.2}$$

Then, for $s \in S_{ba}, t \in T_{c,b}, s'' \in S''_{b',a}, t'' \in T''_{c,b'}$, the following holds:

$$\begin{aligned}
 &(\nu \circ \tau)(\mu \circ \sigma)([s, t], [s'', t'']) \\
 &\stackrel{(4.5)}{=} \sum_{\substack{\text{classes} \\ [s', t'] \in (T' \circ S')_{c,a}}} (\mu \circ \sigma)([s, t], [s', t']) (\nu \circ \tau)([s', t'], [s'', t'']) \\
 &\stackrel{(A.2)}{=} \sum_{\substack{\text{classes} \\ [s', t'] \in (T' \circ S')_{c,a} \\ (s, t) \in \langle s', t' \rangle_b}} \sigma(s, \tilde{s}) \mu(t, \tilde{t}) (\nu \circ \tau)([s', t'], [s'', t'']) \\
 &= \sum_{\substack{\tilde{s} \in S_{b,a}, \tilde{t} \in T_{c,b}}} \sigma(s, \tilde{s}) \mu(t, \tilde{t}) (\nu \circ \tau)([\tilde{s}, \tilde{t}], [s'', t'']) \\
 &\stackrel{(A.2)}{=} \sum_{\substack{\tilde{s} \in S_{b,a}, \tilde{t} \in T_{c,b}}} \sigma(s, \tilde{s}) \mu(t, \tilde{t}) \sum_{(\tilde{s}', \tilde{t}') \in \langle s'', t'' \rangle_b} \tau(\tilde{s}, \tilde{s}') \nu(\tilde{t}, \tilde{t}') \\
 &\stackrel{(4.5)}{=} \sum_{(\tilde{s}, \tilde{t}) \in \langle s'', t'' \rangle_b} \tau \sigma(s, \tilde{s}) \nu \mu(t, \tilde{t}) \\
 &\stackrel{(A.2)}{=} ((\nu\mu) \circ (\tau\sigma))([s, t], [s'', t''])
 \end{aligned}$$

Associator. Let $P : \mathbf{A} \rightarrow \mathbf{B}$, $Q : \mathbf{B} \rightarrow \mathbf{C}$, $R : \mathbf{C} \rightarrow \mathbf{D}$ be profunctors. It is well known that there are natural isomorphisms (and hence proisomorphisms)

$$\alpha_{P,Q,R} : R \circ (Q \circ P) \Rightarrow (R \circ Q) \circ P$$

fulfilling the pentagon equation. In our notation,

$$\alpha_{P,Q,R}([p, q], r) = [p, [q, r]] \quad (\text{A.3})$$

for $p \in P_{b,a}$, $q \in Q_{c,b}$ and $r \in R_{d,c}$. It remains to show that α remains natural in P, Q, R after extending to \mathbf{GpdAct}_f .

Claim. $\alpha_{P,Q,R}$ is natural in P, Q, R .

Proof. Let $\sigma : P \Rightarrow P'$, $\tau : Q \Rightarrow Q'$ and $\mu : R \Rightarrow R'$ be protransformations. Then, the following holds for $p \in P_{b,a}$, $q \in Q_{c,b}$, $r \in R_{d,c}$, $p' \in P'_{b,a}$, $q' \in Q'_{c,b}$, $r' \in R'_{d,c}$:

$$\begin{aligned} & ((\mu \circ \tau) \circ \sigma) \alpha_{P,Q,R}([p, q], r), [p', [q', r']] \\ & \stackrel{(\text{A.3})}{=} ((\mu \circ \tau) \circ \sigma)([p, [q, r]], [p', [q', r']]) \\ & \stackrel{(4.4)}{=} \sum_{b' \xrightarrow{f} b} \sigma(p.f, p') (\mu \circ \tau)([q, r], f.[q', r']) \\ & \stackrel{(4.4)}{=} \sum_{\substack{b' \xrightarrow{f} b \\ c' \xrightarrow{g} c}} \sigma(p.f, p') \tau(q.g, f.q') \mu(r, g.r') \end{aligned}$$

while

$$\begin{aligned} & \alpha_{P',Q',R'}(\mu \circ (\tau \circ \sigma))([p, q], r), [p', [q', r']] \\ & \stackrel{(\text{A.3})}{=} (\mu \circ (\tau \circ \sigma))([p, [q, r]], [p', [q', r']]) \\ & \stackrel{(4.4)}{=} \sum_{c' \xrightarrow{g} c} (\tau \circ \sigma)([p, q].g, [p', q']) \mu(r, g.r') \\ & \stackrel{(4.4)}{=} \sum_{\substack{b' \xrightarrow{f} b \\ c' \xrightarrow{g} c}} \sigma(p.f, p') \tau(q.g, f.q') \mu(r, g.r') \end{aligned}$$

Unitors. For every groupoid \mathbf{A} , we define the identity profunctor

$$\mathbb{1}_{\mathbf{A}} : \mathbf{A}^{\text{op}} \times \mathbf{A} \rightarrow \mathbf{Set} \quad a, a' \mapsto \text{Hom}(a, a').$$

Let $P : \mathbf{A} \rightarrow \mathbf{B}$ be a profunctor. It is well known that there are natural isomorphisms (and thus proisomorphisms) $\lambda_P : P \circ \mathbb{1}_{\mathbf{A}} \Rightarrow P$ and $\rho_P : \mathbb{1}_{\mathbf{B}} \circ P \Rightarrow P$ fulfilling the triangle equations. In our notation

$$\lambda_P([f, p]) := f.p \quad \rho_P([p, g]) = p.g \quad (\text{A.4})$$

for $f \in \text{Hom}(a, a')$, $p \in P_{b,a}$, $g \in \text{Hom}(b', b)$.

Claim. ρ_P and λ_P are natural in P .

Proof. Let $\sigma : P \Rightarrow Q$ be a protransformation. For $f \in \text{Hom}(a, a')$, $p \in P_{b,a}$ and $q \in Q_{b,a'}$ we have

$$\sigma \lambda_P([f, p], q) = \sigma(f.p, q)$$

and

$$\begin{aligned}
& \lambda_Q(\sigma \circ 1_{\mathbb{1}_A})([f, p], q) \\
& \stackrel{(4.5)}{=} \sum_{\substack{\text{classes} \\ [g, q'] \in (Q \circ \mathbb{1}_A)_{b, a'}}} (\sigma \circ 1_{\mathbb{1}_A})([f, p], [g, q']) \lambda_Q([g, q'], q) \\
& \stackrel{(A.2)}{=} \sum_{\substack{\text{classes} \\ [g, q'] \in (Q \circ \mathbb{1}_A)_{b, a'} \\ (\tilde{g}, \tilde{q}) \in (g, q')_a}} 1_{\mathbb{1}_A}(f, \tilde{g}) \sigma(p, \tilde{q}) \lambda_Q([\tilde{g}, \tilde{q}], q) \\
& \stackrel{(A.4)}{=} \sum_{\substack{\tilde{g} \in \text{Hom}(a, a') \\ \tilde{q} \in Q_{b, a}}} \delta_{f, \tilde{g}} \sigma(p, \tilde{g}) \delta_{\tilde{g}, \tilde{q}, q} \\
& = \sigma(p, f^{-1}.q) \stackrel{(4.2)}{=} \sigma(f.p, q) = \sigma \lambda_P([f, p], q)
\end{aligned}$$

A similar proof shows that ρ_P is natural in P . \square

Theorem 4.5 . *The structures defined above yield a dagger pivotal structure [CR16, Section 2.1] for the 2-category \mathbf{GpdAct}_f .*

Proof. We first consider the dagger structure, then the pivotal structure.

Dagger. From the symmetry of the expressions (4.5) and (4.4) we conclude that $(\sigma\tau)^\dagger = \tau^\dagger\sigma^\dagger$ and $(\sigma \circ \tau)^\dagger = \sigma^\dagger \circ \tau^\dagger$. All invertible natural transformations and in particular all coherence isomorphisms are unitary.

Pivotality. We show that for every profunctor $P : \mathbf{A} \leftrightarrow \mathbf{B}$, the cup (4.8) and cap (4.7) are well defined. We first show that ϵ descends to a map on the quotient. Let $p \in P_{b, a}, p' \in P_{b, a'}, q \in P_{b', a}, q' \in P_{b', a'}$ and $f \in \text{Hom}(a', a)$ and suppose that $(p, p') \sim (q, q')$. Then, there is a $b \xrightarrow{r} b'$ s.t. $(p.r, r^{-1} \star p') = (q, q')$, where $r^{-1} \star p' = p'.r$ is the dual action. Therefore

$$\epsilon([q, q'], f) = \delta_{f^{-1}.q, q'} = \delta_{f^{-1}.p.r, p'.r} = \delta_{f^{-1}.p, p'}.$$

To show that ϵ is an intertwiner we let p, p' and f be as above and let $a \xrightarrow{g} \tilde{a}$ and $\tilde{a}' \xrightarrow{h} a'$. Therefore,

$$\epsilon(g.[p, p'].h, gfh) = \delta_{h^{-1}f^{-1}g^{-1}g.p, h^{-1}.p'} = \delta_{f^{-1}.p, p'}.$$

Using unitors and associators, we can show that the left transpose of any morphism $\mu : P \rightrightarrows S$, given by the composite

$$(\epsilon \circ 1_{P_*})(1_{S^*} \circ \mu \circ 1_{P_*})(1_{S^*} \circ \eta),$$

is equal to the protransformation $\mu^* : S^* \rightrightarrows P^*$ defined by the functions

$$S_{a, b} \times P_{a, b} \xrightarrow{\text{swap}} P_{a, b} \times S_{a, b} \xrightarrow{\mu} \mathbb{N}.$$

For $\mu = 1_P : P \rightrightarrows P$ this implies the snake equation. A similar argument shows that the right tranpose is given by the same equation. This proves that our 2-category is a dagger pivotal 2-category. \square

Theorem 4.8 . *Definition 4.7 defines a dagger 2-functor.*

Proof (sketch). Here, we use the description of $\mathbf{C}^*\mathbf{Alg}$ in terms of symmetric separable dagger Frobenius algebras, Frobenius bimodules and intertwiners [Vic10, HVW14, CR16].

We first show that the data given in Definition 4.7 is well defined: Given a profunctor $P : \mathbf{A} \rightarrow \mathbf{B}$, the vector space $\bigoplus_{b,a} \mathbb{C}P_{b,a}$ is a $\mathbb{C}\mathbf{A}$ – $\mathbb{C}\mathbf{B}$ bimodule with action

$$\mathbb{C}\mathbf{A} \otimes \bigoplus_{b,a} \mathbb{C}P_{b,a} \otimes \mathbb{C}\mathbf{B} \rightarrow \bigoplus_{b,a} \mathbb{C}P_{b,a}$$

induced from the action (4.1) of the morphisms on $P_{b,a}$ (again defined to be 0 on non-compatible components).

Given a protransformation $\sigma : P \Rightarrow T$, the linear map $Q(\sigma)$ is indeed an intertwiner: For $p \in P_{b,a}$, $a \xrightarrow{f} a'$, $b' \xrightarrow{g} b$:

$$\begin{aligned} Q(\sigma)(f.p.g) &= \sum_{t \in T_{b,a}} \sigma(f.p.g, t)t = \sum_{t \in T_{b,a}} \sigma(p, f^{-1}.t.g^{-1})t \\ &= \sum_{\tilde{t} \in T_{b,a}} \sigma(p, \tilde{t})f.\tilde{t}.g = f.Q(\sigma).g \end{aligned}$$

By definition, Q preserves the dagger structure.

The critical step in the proof of functoriality is showing that the horizontal composition of 2-morphisms is preserved. Formally, we defined horizontal composition of profunctors $P : \mathbf{A} \rightarrow \mathbf{B}$ and $S : \mathbf{B} \rightarrow \mathbf{C}$ and protransformations via the coequalizer of the maps

$$\begin{aligned} \coprod_{b,b'} P_{b',a} \times \text{Hom}(b, b') \times S_{c,b} &\rightarrow \coprod_b P_{b,a} \times S_{c,b} \\ (p, f, s) &\mapsto (p.f, s) \\ (p, f, s) &\mapsto (p, f.s) \end{aligned}$$

leading to the equivalence relation $(p, s) \sim (p.f, f^{-1}.s)$. We denoted this coequalizer as follows:

$$\epsilon : \coprod_{b \in \text{ob}\mathbf{B}} P_{b,a} \times S_{c,b} \rightarrow (S \circ P)_{c,a} \quad (p, s) \mapsto [p, s]$$

In $\mathbf{C}^*\mathbf{Alg}$, the horizontal composition of $Q(P)$ and $Q(S)$ are defined via the splitting of the dagger idempotent [HVW14] $X \in \text{End}\left(\bigoplus_{b,a} \mathbb{C}P_{b,a} \otimes \bigoplus_{c,\tilde{b}} \mathbb{C}S_{c,\tilde{b}}\right)$ given at stage $((b, a, c, \tilde{b}), (b', a', c', \tilde{b}'))$ as

$$\begin{aligned} \mathbb{C}P_{b,a} \otimes \mathbb{C}S_{c,\tilde{b}} &\rightarrow \mathbb{C}P_{b',a'} \otimes \mathbb{C}S_{c',\tilde{b}'} \\ (p, s) &\mapsto \frac{\delta_{b,\tilde{b}}\delta_{b',\tilde{b}'}\delta_{a,a'}\delta_{c,c'}}{|\text{Hom}(b', b)|} \sum_{b' \xrightarrow{f} b} (p.f, f^{-1}.s) \end{aligned}$$

To show that our functor preserves horizontal composition we thus have to show that the quantization of the coequalizer $Q(\epsilon) : (p, s) \mapsto [p, s]$ splits X (up to a constant). And indeed,

$$Q(\epsilon)^\dagger Q(\epsilon) \in \text{End}\left(\bigoplus_{a,b,c} \mathbb{C}P_{b,a} \otimes \mathbb{C}S_{c,b}\right)$$

is given at stage $((a, b, c), (a', b', c'))$ as

$$\begin{aligned} \mathbb{C}P_{b,a} \otimes \mathbb{C}S_{c,b} &\rightarrow \mathbb{C}P_{b',a'} \otimes \mathbb{C}S_{c',b'} \\ (p, s) &\mapsto \delta_{a,a'}\delta_{c,c'} \sum_{(\tilde{p}, \tilde{s}) \in (p,s)_{b'}} (\tilde{p}, \tilde{s}) \end{aligned}$$

$$= \delta_{a,a'} \delta_{c,c'} \frac{1}{|\text{St}(p, s)|} \sum_{b' \xrightarrow{f} b} (p.f, f^{-1}.s)$$

where $\text{St}(p, s) = \left\{ b \xrightarrow{g} b \mid (p.g, g^{-1}.s) = (p, s) \right\}$ is the stabilizer of the joint action of $\text{Aut}_{\mathbf{B}}(b)$ on $P_{b,a} \times S_{c,b}$ and in $\langle p, s \rangle_{b'}$ is defined in (A.1). Since all groupoids in $\mathbf{GpdAct}_{\mathbf{f}}$ act freely, this stabilizer vanishes and $Q(\epsilon)^{\dagger}Q(\epsilon)$ is indeed proportional to X . Similarly $Q(\epsilon)Q(\epsilon)^{\dagger}$ is proportional to $\mathbb{1}_{Q(S \circ P)}$. \square

REFERENCES

- [AM15] Akintomide Adesanmi and Lotfi Mhamdi. Controlling TCP Incast congestion in data centre networks. In *ICCW 2015*. IEEE, 2015.
- [Bae97] John C. Baez. Higher-dimensional algebra II. 2-Hilbert spaces. *Advances in Mathematics*, 127(2):125–189, 1997. [arXiv:9609018](#).
- [BB85] Charles H. Bennett and Gilles Brassard. Quantum public key distribution. *IBM Tech. Disc. Bul.*, 28:3153–3163, 1985.
- [BBFW12] John Baez, Aristide Baratin, Laurent Freidel, and Derek Wise. Infinite-dimensional representations of 2-groups. *Memoirs of the American Mathematical Society*, 219(1032), 2012.
- [BD01] John Baez and James Dolan. From finite sets to Feynman diagrams. In *Mathematics Unlimited*, pages 29–50. 2001.
- [BD15] Miriam Backens and Ali Nabi Duman. A complete graphical calculus for Spekkens’ toy bit theory. *Foundations of Physics*, 46(1):70–103, 2015.
- [BDSPV15] Bruce Bartlett, Christopher L. Douglas, Christopher Schommer-Pries, and Jamie Vicary. Modular categories as representations of the 3-dimensional bordism 2-category. 2015. [arXiv:1509.06811](#).
- [Ber] Daniel J. Bernstein. Introduction to post-quantum cryptography. In *Post-Quantum Cryptography*, pages 1–14. Springer Nature.
- [BHW10] John Baez, Alexander Hoffnung, and Christopher Walker. HDA VII: Groupoidification. *TAC*, 24(18):489–553, 2010.
- [BLLR97] Francois Bergeron, Gilbert Labelle, Pierre Leroux, and Margaret Readdy. *Combinatorial Species and Tree-like Structures*. Cambridge University Press (CUP), 1997.
- [BMZ12] Alcides Buss, Ralf Meyer, and Chenchang Zhu. A higher category approach to twisted actions on C^* -algebras. *Proceedings of the Edinburgh Mathematical Society*, 56(02):387–426, Aug 2012.
- [Bor94] Francis Borceux. *Handbook of Categorical Algebra*. CUP, 1994.
- [Bor16] Paul Borrill. The timeless datacentre. *Stanford Colloquium on Computer Systems*, 2016. [YouTube:IPTITmH-YvQ](#).
- [BV14] Krzysztof Bar and Jamie Vicary. Groupoid semantics for thermal computing. [arXiv:1401.3280](#), 2014.
- [CES11] Bob Coecke, Bill Edwards, and Robert W. Spekkens. Phase groups and the origin of non-locality for qubits. *ENTCS*, 270(2):15–36, 2011.
- [CP12] Bob Coecke and Simon Perdrix. Environment and classical channels in categorical quantum mechanics. *LMCS*, 8(4), 2012.

- [CR16] Nils Carqueville and Ingo Runkel. Orbifold completion of defect bicategories. *Quantum Topol.*, 7(2):203–279, 2016. arXiv:1210.6363.
- [DM16] Leonardo Disilvestro and Damian Markham. Quantum protocols within Spekkens’ toy model. arXiv:1608.09012, 2016.
- [ea14] Romain Alléaume et al. Using quantum key distribution for cryptographic purposes: a survey. *TCS*, 560(1):62–81, 2014.
- [Eke91] Artur Ekert. Quantum cryptography based on Bell’s theorem. *Physical Review Letters*, 67:661, 1991.
- [HVW14] Chris Heunen, Jamie Vicary, and Linde Wester. Mixed quantum states in higher categories. *Electronic Proceedings in Theoretical Computer Science*, 172:304–315, Dec 2014.
- [IAC99] Sami Iren, Paul D. Amer, and Phillip T. Conrad. The transport layer: tutorial and survey. *ACM Computing Surveys*, 31(4):360–404, 1999.
- [JLW16] Arthur Jaffe, Zhengwei Liu, and Alex Wozniakowski. Holographic software for quantum networks. 2016. arXiv:1605.00127.
- [JMS13] Vaughan F. R. Jones, Scott Morrison, and Noah Snyder. The classification of subfactors of index at most 5. *Bull. Amer. Math. Soc.*, 51(2):277–327, 2013. arXiv:1304.6141.
- [Jon89] Vaughan F.R. Jones. On knot invariants related to some statistical mechanical models. *Pac. J. Math.*, 137:311–334, 1989.
- [Jon99] Vaughan F. R. Jones. Planar algebras, I. 1999. arXiv:math/9909027.
- [Joy81] André Joyal. Une théorie combinatoire des séries formelles. *Advances in Mathematics*, 42(1):1–82, 1981.
- [Mau93] Ueli M. Maurer. Protocols for secret key agreement by public discussion based on common information. *IEEE Transactions on Information Theory*, 39(3):733–742, 1993.
- [Mor06] *Theory Appl. Categ.*, 16:No. 29, 785–854, 2006. arXiv:math/0601458.
- [MP14] Scott Morrison and Emily Peters. The little desert? Some subfactors with index in the interval $(5, 3 + \sqrt{5})$. *International Journal of Mathematics*, 25(08):1450080, 2014. arXiv:1205.2742.
- [NC09] Michael Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information*. CUP, 2009.
- [Ocn89] Adrian Ocneanu. Quantized groups, string algebras, and Galois theory for algebras. In *Operator Algebras and Applications*, pages 119–172. CUP, 1989.
- [Pus12] Matthew F. Pusey. Stabilizer notation for spekkens’ toy theory. *Foundations of Physics*, 42(5):688–708, 2012.
- [RV16] David Reutter and Jamie Vicary. Biunitary constructions in quantum information. 2016. arXiv:1609.07775.
- [RV18] David Reutter and Jamie Vicary. Shaded tangles for the design and verification of quantum programs. 2018. arXiv:1805.01540.
- [Sel10] Peter Selinger. A survey of graphical languages for monoidal categories. In *N. Struct. Phys.*, pages 289–355. Springer, 2010. arXiv:0908.3347.
- [Spe07] Robert W. Spekkens. Evidence for the epistemic view of quantum states: A toy theory. *PRA*, 75(3), 2007.
- [Vic10] Jamie Vicary. Categorical formulation of finite-dimensional quantum algebras. *Communications in Mathematical Physics*, 304(3):765–796, Nov 2010.
- [Vic12a] Jamie Vicary. Higher quantum theory. 2012. arXiv:1207.4563.

- [Vic12b] Jamie Vicary. Higher semantics of quantum protocols. In *Proceedings of LICS 2012*, 2012.
- [VV14] Umesh Vazirani and Thomas Vidick. Robust device independent quantum key distribution. In *ITCS 2014*, 2014.
- [Zha15] Yongwang Zhao. A survey on formal specification and verification of separation kernels. 2015.