A Classical Groupoid Model for Quantum Networks*

David Reutter[1] and Jamie Vicary[2]

1 Department of Computer Science, University of Oxford, UK
david.reutter@cs.ox.ac.uk

2 Department of Computer Science, University of Oxford, UK
jamie.vicary@cs.ox.ac.uk

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.

1998 ACM Subject Classification C.2.1 Network Architecture and Design, F.4.3 Formal Languages

Keywords and phrases groupoids, networks, quantum, semantics, key distribution

Digital Object Identifier 10.4230/LIPIcs.CALCO.2017.19

1 Introduction

Borrill and Karp have recently introduced the notion of *timeless network* [9], a new paradigm for distributed communication currently under commercial development by Earth Computing[1]. 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, in their simplest instantiations. 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:

* An extended version of this paper can be found at [28], https://arxiv.org/abs/1707.00966.

[1] See http://www.earthcomputing.io.

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

The **Read** operation destroys a groubit and creates a conventional bit, while the **Write** operation destroys a conventional bit and creates a new groubit. 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**$\big((A_L,A_I),(B_L,B_I)\big)=\big((A_L,A_I\oplus B_L),(B_L,B_I\oplus A_L)\big)$

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.
- **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 [34] 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 [16] can cause reliability issues in a data centre environment [1, 9].

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 [8]. Should this possibility be realized, it has been suggested that quantum key distribution could be used as an alternative technology to enable long-range information theoretically–secure communication [15]; we suggest that key distribution running on a large-scale groubit network may be an alternative worth investigating.

[2] For quantum protocols such as quantum key distribution, security is derivable from the laws of physics; this is not the case here [32].

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 [24].

Nonetheless, for those quantum protocols that we claim can operate on a groubit network, we mean this in a strong sense. In an extended version of this paper [28], 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 [27,33]. 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 $\mathbf{GpdAct_s}$ of finite groupoids, free profunctors and spans, and in our central technical result, show that groubits correspond precisely to biunitary connections in $\mathbf{GpdAct_s}$[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 $\mathbf{GpdAct_s}$ and its quantization functor are given in an extended version of this paper [28].

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 [9], 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 [2, 11, 13, 26, 31] 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 [31, Section I] with Figure 12 here.)

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

[3] See 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 [3, 4, 7, 23] from the combinatorial species of Joyal [20]; 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: while their work is based on groupoids, spans, and spans of spans, 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 [21] 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 [25] in 1989 and since developed by Jones, Morrison and others [18, 19, 22], 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 [17, 27, 29, 33] 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 [5].

2 Foundations

2.1 Groudits and dits

Groudits. We begin with the definition of a groudit.

▶ **Definition 1.** A *groudit* $\mathcal{G}$ is a skeletal groupoid of the form $\mathbf{G} = \coprod_i G_i$, where G_i are finite groups, equipped for each $i \in \mathrm{Ob}(\mathbf{G})$ with bijections $\sigma_i, \tau_i : G_i \to \mathrm{Ob}(\mathbf{G})$.

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

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

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

▶ **Definition 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:

$$\mathrm{Ob}(\mathbf{B}) := \mathbb{Z}_2 \qquad \mathrm{Mor}(\mathbf{B}) := \mathbb{Z}_2 \times \mathbb{Z}_2 \qquad s, t := \mathbb{Z}_2 \times \mathbb{Z}_2 \xrightarrow{\pi_1} \mathbb{Z}_2 \tag{1}$$

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 $a \to a$. 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 $\sigma_i, \tau_i : G_i = \mathbb{Z}_2 \to \mathrm{Ob}(B) = \mathbb{Z}_2$. 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 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 $i \in \mathrm{Mor}(\mathbf{D})$. 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 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 groubit. Figure 2(a)–(c) shows the three groubit-only operations: **Swap** and **Tick** are deterministic, while **Init** creates a groubit in a nondeterministic logical state. Figure 2(c) uses a rotated

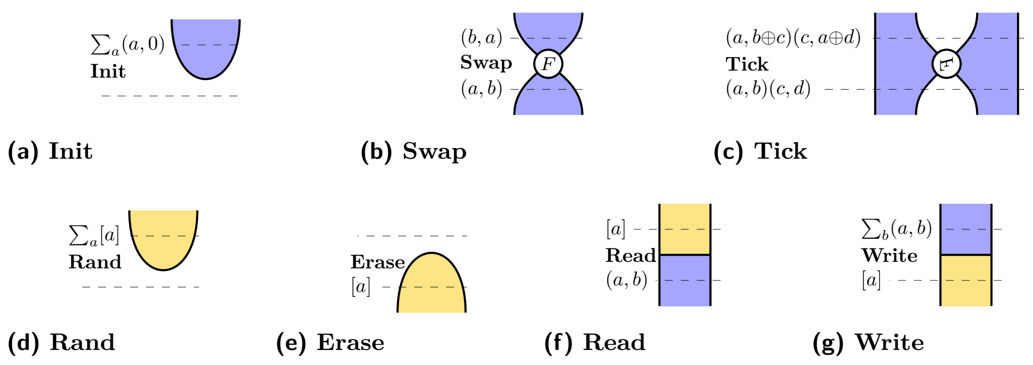


Figure 2 Atomic groubit and bit operations.

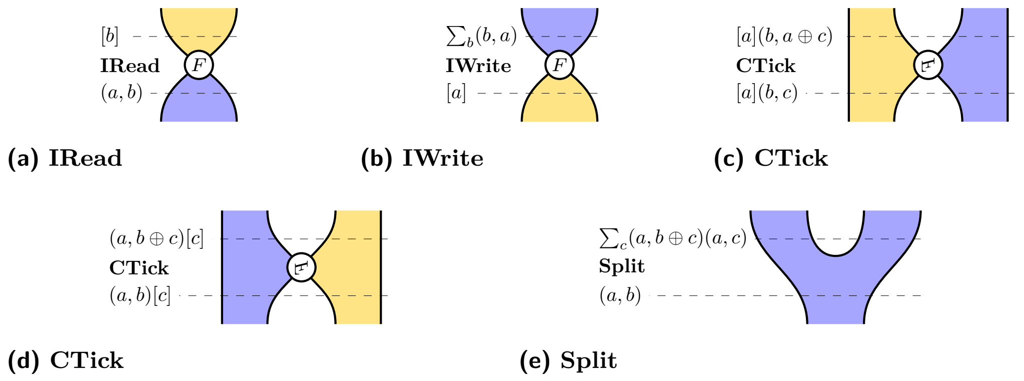


Figure 3 Derived groubit and bit operations.

letter to label the vertex, since it is represented algebraically by a rotated version of Figure 2(b) under the dagger pivotal structure (see discussion below.)

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 groubit and a bit: **Read** depicts the read-out of the logical state of a groubit, and **Write** depicts the initialization of a groubit with given logical bit and random internal bit.

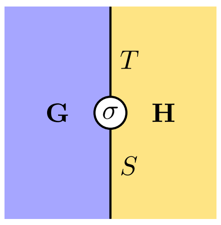
Derived operations. In Figure 3 we list the derived operations **IRead**, **IWrite**, **CTick** and **Split**. Note that **CTick** comes in both left and right versions, distinguished by their images. We will see how they are defined in terms of atomic operations later in this section.

2.2 Graphical calculus

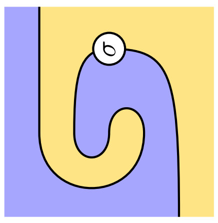
Definition. Our graphical calculus represents groupoids, free actions and spans. We begin with an informal definition of the 2-category formed by these structures. Throughout, we write '2-category' to refer to the weak structure, which is sometimes called 'bicategory'.

▶ **Definition 4.** The 2-category $\mathbf{GpdAct_s}$ is built from the following structures:

- *objects* are finite skeletal groupoids $\mathbf{G}$, $\mathbf{H}$, ...;



(a) A 2-morphism



(b) A deformation

Figure 4 Examples of the graphical calculus.

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

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 to guarantee 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 an extended version of this paper [28].

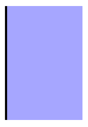
▶ **Definition 5.** For an equivariant span $\sigma : S \Rrightarrow T$, we define its *dagger* $\sigma^\dagger : T \Rrightarrow S$ as the converse: $\sigma^\dagger_{a,b}(t,s) := \sigma_{a,b}(s,t)$.

Graphical calculus. We use a 2-dimensional graphical calculus (see Figure 4(a)) to denote a 2-morphism σ in $\mathbf{GpdAct_s}$. This is the standard graphical calculus for 2-categories [30]: objects $\mathbf{G}$, $\mathbf{H}$ label the regions, morphisms $S, T : \mathbf{G} \nrightarrow \mathbf{H}$ label the wires, and 2-morphisms $\sigma : S \Rrightarrow T$ label the vertices. We often drop the labels; also, white regions will always correspond to the trivial groupoid $\mathbf{1}$ with one morphism.

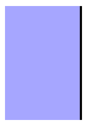
Stacking these pictures vertically performs composition of spans, stacking them horizontally performs bimodule composition, and reflecting them about a horizontal axis corresponds to the dagger operation of Definition 5. In fact, $\mathbf{GpdAct_s}$ is a *dagger pivotal 2-category* [10, Section 2.1], giving immense freedom in the graphical calculus: one may reflect, rotate and deform parts of the pictures arbitrarily (holding the boundaries fixed), preserving equality. For example, since the images of Figure 4(a) and (b) are deformations of each other with constant boundary, they represent equal 2-morphisms.

2.2.0.1 Boundaries.

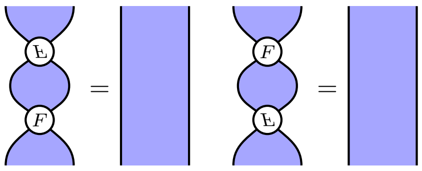
For every shaded region labelled by a skeletal groupoid $\mathbf{G}$, we have canonical boundaries drawn as follows:



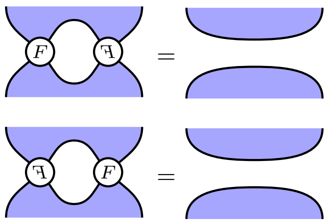
$L^{\mathbf{G}} : \mathbf{1} \nrightarrow \mathbf{G}$



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



(a) Vertical unitarity



(b) Horizontal unitarity

Figure 5 The biunitarity equations.

We define these as the following free profunctors, for all objects $a \in \mathbf{G}$:

$$L^{\mathbf{G}}_{\bullet,a} := \mathrm{Aut}_{\mathbf{G}}(a) \qquad (\bullet, g, g') \mapsto gg' \tag{2}$$

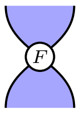
$$R^{\mathbf{G}}_{a,\bullet} := \mathrm{Aut}_{\mathbf{G}}(a) \qquad (g', g, \bullet) \mapsto g'g \tag{3}$$

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 6.** In $\mathbf{GpdAct_s}$, a biunitary on a skeletal groupoid $\mathbf{G}$ is a unitary 2-morphism



(4)

fulfilling the equations depicted in Figure 5.

The source and target of a biunitary is the set $\mathrm{Mor}(\mathbf{G})$ of morphisms of the skeletal groupoid. So concretely, a biunitary is an automorphism of $\mathrm{Mor}(\mathbf{G})$ satisfying an algebraic condition. The following theorem determines this condition precisely.

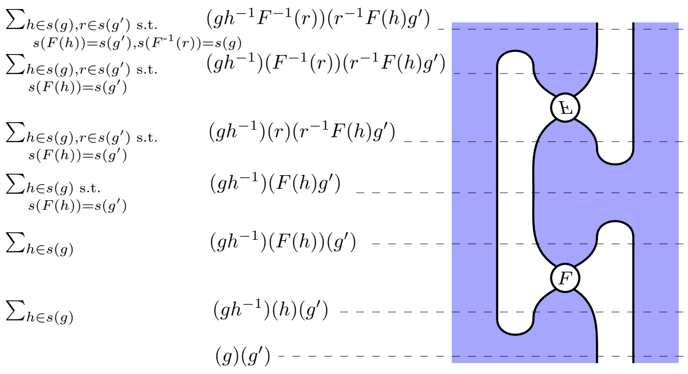
▶ **Theorem 7.** *A biunitary on a skeletal groupoid* $\mathbf{G}$ *is a bijection* $F : \mathrm{Mor}(\mathbf{G}) \to \mathrm{Mor}(\mathbf{G})$ *such that for all* $a, b \in \mathrm{Ob}(\mathbf{G})$*, we have:*

$$|F(\mathrm{Aut}_{\mathbf{G}}(a)) \cap \mathrm{Aut}_{\mathbf{G}}(b)| = 1 \tag{5}$$

Proof. The equations of Figure 5(a) say that F is unitary, which means precisely that it acts as a permutation on $\mathrm{Mor}(\mathbf{G})$. The equations of Figure 5(b) are equivalent to the composite of Figure 6 being the identity.

This holds just when, for all $a, b \in \mathrm{Ob}(\mathbf{G})$ and for all $g \in \mathrm{Aut}_{\mathbf{G}}(a)$ and $g' \in \mathrm{Aut}_{\mathbf{G}}(b)$, there are unique $h \in \mathrm{Aut}_{\mathbf{G}}(a)$, $r \in \mathrm{Aut}_{\mathbf{G}}(b)$ with $s(F(h)) = b$ and $s(F^{-1}(r)) = a$ satisfying the following conditions:

$$gh^{-1}F^{-1}(r) = g$$



$\sum_{h\in s(g), r\in s(g') \text{ s.t. } s(F(h))=s(g'), s(F^{-1}(r))=s(g)}$
$(gh^{-1}F^{-1}(r))(r^{-1}F(h)g')$
$\sum_{h\in s(g), r\in s(g') \text{ s.t. } s(F(h))=s(g')}$
$(gh^{-1})(F^{-1}(r))(r^{-1}F(h)g')$
$\sum_{h\in s(g), r\in s(g') \text{ s.t. } s(F(h))=s(g')}$
$(gh^{-1})(r)(r^{-1}F(h)g')$
$\sum_{h\in s(g) \text{ s.t. } s(F(h))=s(g')}$
$(gh^{-1})(F(h)g')$
$\sum_{h\in s(g)}$
$(gh^{-1})(F(h))(g')$
$\sum_{h\in s(g)}$
$(gh^{-1})(h)(g')$
$(g)(g')$
F
F

Figure 6 Verifying the action of a biunitary.

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

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

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

▶ **Theorem 8.** *For a skeletal groupoid* $\mathbf{G}$*, groudits on* $\mathbf{G}$ *are in bijective correspondence with biunitaries on* $\mathbf{G}$*.*

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

$$s(\epsilon_a^{-1}(b)) = a. \tag{6}$$

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:

$$\epsilon_1 : \mathrm{Mor}(\mathbf{G}) \to \mathrm{Ob}(\mathbf{G})\times\mathrm{Ob}(\mathbf{G}) \qquad \epsilon_1(g) := (sg, \epsilon_{sg}(g)) \tag{7}$$

$$\epsilon_2 : \mathrm{Ob}(\mathbf{G})\times\mathrm{Ob}(\mathbf{G}) \to \mathrm{Mor}(\mathbf{G}) \qquad \epsilon_2(a, b) := \epsilon_a^{-1}(b) \tag{8}$$

We can show that ϵ_1 and ϵ_2 are inverse:

$$\epsilon_1(\epsilon_2(a, b)) = (s(\epsilon_a^{-1}(b)), \epsilon_{s(\epsilon_a^{-1}(b))}(\epsilon_a^{-1}(b)) \overset{(6)}{=} (a, \epsilon_a(\epsilon_a^{-1}(b))) = (a, b)$$

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

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

$$\mathrm{Mor}(\mathbf{G}) \xrightarrow{\epsilon_1} \mathrm{Ob}(\mathbf{G})\times\mathrm{Ob}(\mathbf{G}) \xrightarrow{\gamma} \mathrm{Ob}(\mathbf{G})\times\mathrm{Ob}(\mathbf{G}) \xrightarrow{\tau_2} \mathrm{Mor}(\mathbf{G})$$

Then a simple calculation shows the following:

$$F_{\epsilon,\tau}(g) = \tau^{-1}_{\epsilon_{s(g)}(g)}(s(g)) \in \mathrm{Aut}_{\mathbf{G}}(\epsilon_{s(g)}(g)) \tag{9}$$

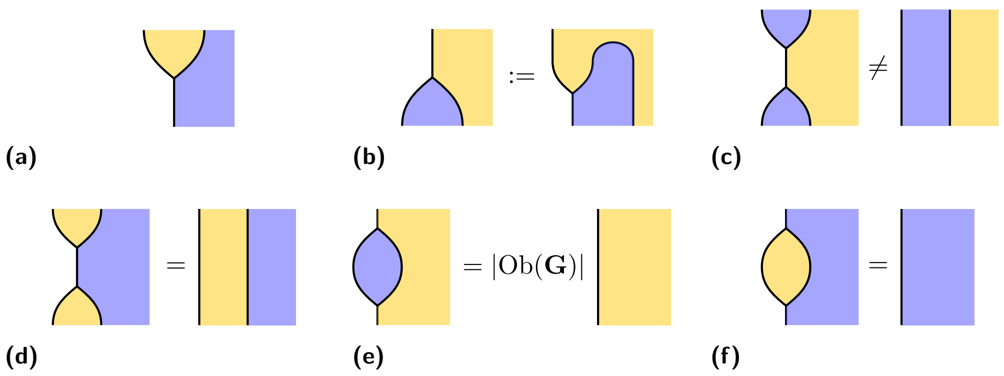


Figure 7 Building blocks for the measurement calculus.

By construction, $F_{\epsilon,\tau}$ is unitary, since it is a composite of bijections. To show it is biunitary, suppose now that $g, g' \in \mathrm{Aut}_{\mathbf{G}}(a)$ such that $s(F_{\epsilon,\tau}(g)) = s(F_{\epsilon,\tau}(g'))$. Then by equation (6), 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 $F : \mathrm{Mor}(\mathbf{G}) \to \mathrm{Mor}(\mathbf{G})$, we define balancers ϵ^F, τ^F for all $a \in \mathrm{Ob}(\mathbf{G})$ and $g \in \mathrm{Mor}(\mathbf{G})$ as

$$\epsilon^F_a(g) := s(F(g)) \qquad\qquad \tau^F_a(g) := s(F^{-1}(g)) \tag{10}$$

We must show that for all $a \in \mathrm{Ob}(\mathbf{G})$, $\epsilon^F_a, \tau^F_a : \mathrm{Aut}_{\mathbf{G}}(a) \to \mathrm{Ob}(\mathbf{G})$ are bijections. First, surjectivity. For any $b \in \mathrm{Ob}(\mathbf{G})$, using the biunitarity property (5), pick the unique morphism $g \in F(\mathrm{Aut}_{\mathbf{G}}(a)) \cap \mathrm{Aut}_{\mathbf{G}}(b)$. Then $F^{-1}(g) \in \mathrm{Aut}_{\mathbf{G}}(a)$ and $b = s(g) = s(F(F^{-1}(g))) = \epsilon^F_a(F^{-1}g)$. A similar proof shows surjectivity of τ^F_G. Next, injectivity. Suppose that $g, h \in \mathrm{Aut}_{\mathbf{G}}(a)$ with $\epsilon^F_a(g) = \epsilon^F_a(h)$; then $s(F(g)) = s(F(h))$. Then $F(g), F(h) \in F(\mathrm{Aut}_{\mathbf{G}}(a)) \cap \mathrm{Aut}_{\mathbf{G}}(s(F(g)))$. Then by the biunitarity property (5), 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 $g \in \mathrm{Aut}_{\mathbf{G}}(a)$, then by (6) we have $\epsilon^{F_{\epsilon,\tau}}_a(g) = s(F_{\epsilon,\tau}(g)) = \epsilon_a(g)$ and similarly $\tau^{F_{\epsilon,\tau}}_a(g) = s(F^{-1}_{\epsilon,\tau}(g)) = \tau^F_a(g)$. In the other direction, given a biunitary F and $g \in \mathrm{Aut}_{\mathbf{G}}(a)$, we observe that $F_{\epsilon^F,\tau^F}(g) = (\tau^F_{\epsilon^F_a(g)})^{-1}(a)$. To show that this equals $F(g)$, we have to show that $\tau^F_{\epsilon^F_a(g)}(F(g)) = a$. And indeed, we have $\tau^F_{\epsilon^F_G(g)}(F(g)) = s(F^{-1}(F(g))) = s(g) = a$. ◀

2.4 Measurements

Syntax. In Section 2.1 we describe 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 are equipped with a biunitary of the form (4), 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. These interact via the 2-morphisms given 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 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.

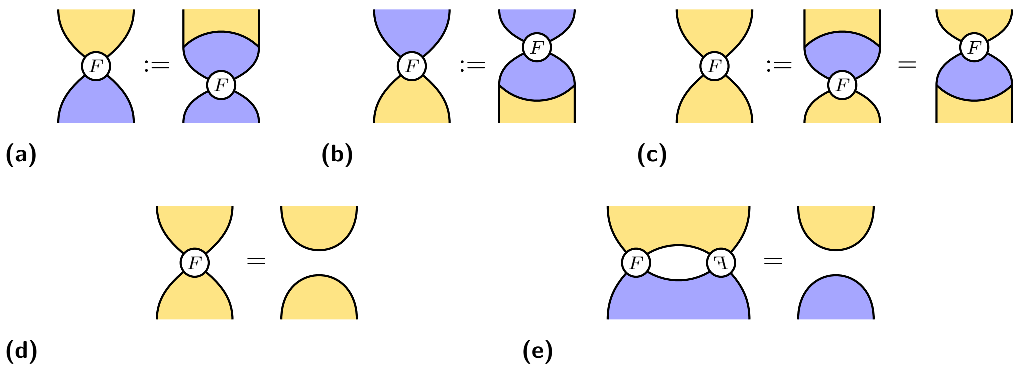


Figure 8 Yellow-blue and yellow-yellow versions of the biunitary.

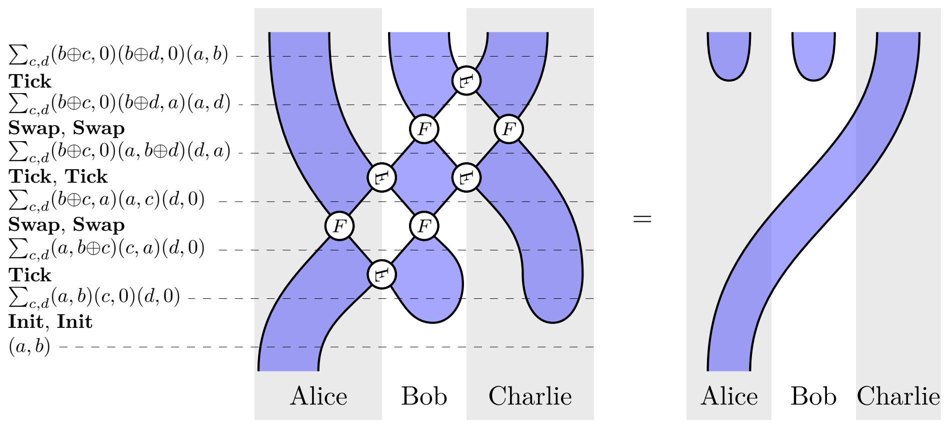
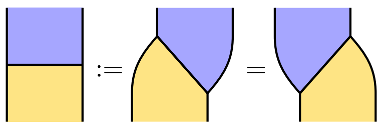

$\sum_{c,d}(b\oplus c,0)(b\oplus d,0)(a,b)$
Tick
$\sum_{c,d}(b\oplus c,0)(b\oplus d,a)(a,d)$
Swap, **Swap**
$\sum_{c,d}(b\oplus c,0)(a,b\oplus d)(d,a)$
Tick, **Tick**
$\sum_{c,d}(b\oplus c,a)(a,c)(d,0)$
Swap, **Swap**
$\sum_{c,d}(a,b\oplus c)(c,a)(d,0)$
Tick
$\sum_{c,d}(a,b)(c,0)(d,0)$
Init, **Init**
(a,b)
F
Alice
Bob
Charlie
=
Alice
Bob
Charlie

Figure 9 State transfer.

These building blocks are required to satisfy the axioms Figure 7(d), (e) and (f). 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:



(11)

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), which we require to satisfy equations Figure 8(d) and (e). These structures yield our composite operations **IRead**, **IWrite**, **LRead**, **RRead** and **CTick**.

Semantics. We suppose that 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 $S : \mathbf{B} \nrightarrow \mathbf{G}$ as follows, where $\emptyset$ is the empty set:

$$S_{b,g} := \begin{cases} \mathrm{Aut}_{\mathbf{G}}(g) & \text{if } b = g \\ \emptyset & \text{otherwise} \end{cases} \tag{12}$$

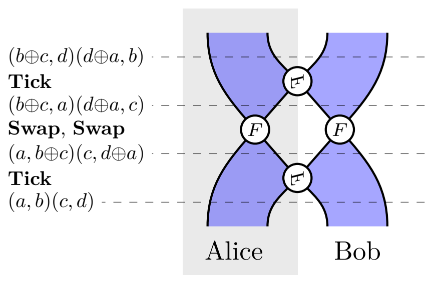

$(b\oplus c, d)(d\oplus a, b)$
Tick
$(b\oplus c, a)(d\oplus a, c)$
Swap, **Swap**
$(a, b\oplus c)(c, d\oplus a)$
Tick
$(a, b)(c, d)$
F
F
F
F
Alice
Bob

Figure 10 The basic state transfer repeating block.

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

Figure 7(a) $$g \mapsto (a, g) \tag{13}$$

The span (13) is unitary, which explains equations Figure 7(d) and (f). These definitions satisfy the required equations.

▶ **Proposition 9.** *Using definitions* (12) *and* (13)*, the equations Figure 7(d)–(f) and Figure 8(d) and (e) all hold.*

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 [29]. 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 Alice, 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

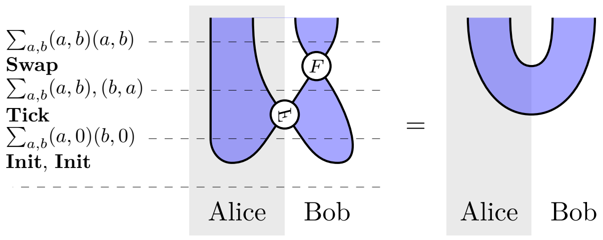


Figure 11 Entanglement creation.

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, due to message atomicity properties that we now explore. This is desirable, since timeouts are a basic feature of the dominant TCP protocol for internet communication [16] which are the source of reliability issues in data centre environments [1, 9]. If any of the 4 operations of the scheme given in Figure 10 fail, Alice assumes that she retains ownership of the message, and is free to send it along another route of the network, or to return a failure message to the sender. Bob assumes ownership of the message if the final **Tick** occurs successfully.

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.

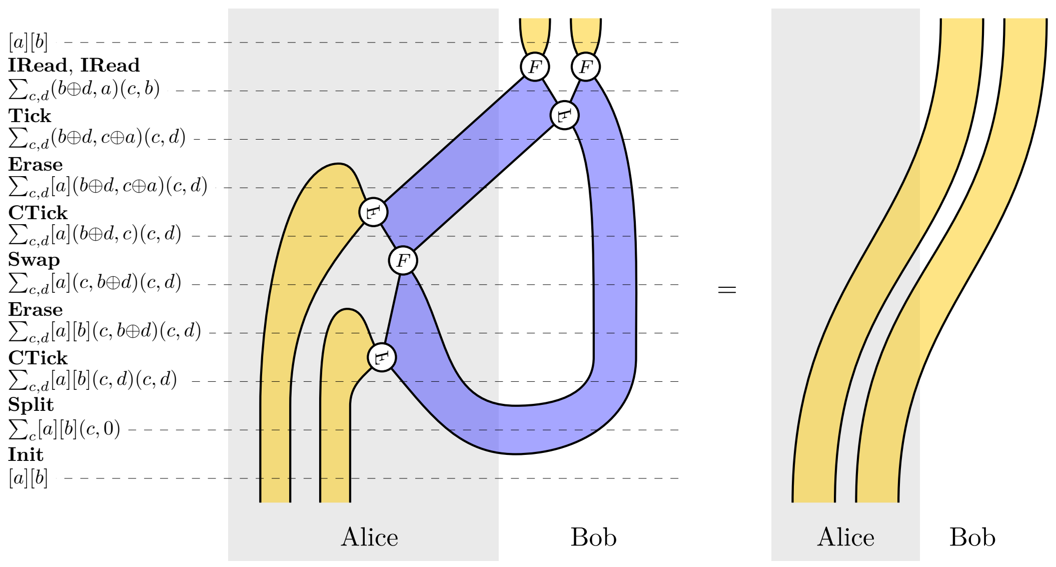

$[a][b]$
IRead, **IRead**
$\sum_{c,d}(b\oplus d, a)(c,b)$
Tick
$\sum_{c,d}(b\oplus d, c\oplus a)(c,d)$
Erase
$\sum_{c,d}[a](b\oplus d, c\oplus a)(c,d)$
CTick
$\sum_{c,d}[a](b\oplus d, c)(c,d)$
Swap
$\sum_{c,d}[a](c, b\oplus d)(c,d)$
Erase
$\sum_{c,d}[a][b](c, b\oplus d)(c,d)$
CTick
$\sum_{c,d}[a][b](c,d)(c,d)$
Split
$\sum_{c}[a][b](c,0)$
Init
$[a][b]$
F
F
F
=
Alice
Bob
Alice
Bob

Figure 12 Dense coding.

Verification. To verify correctness of the program for general groudits, 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 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.

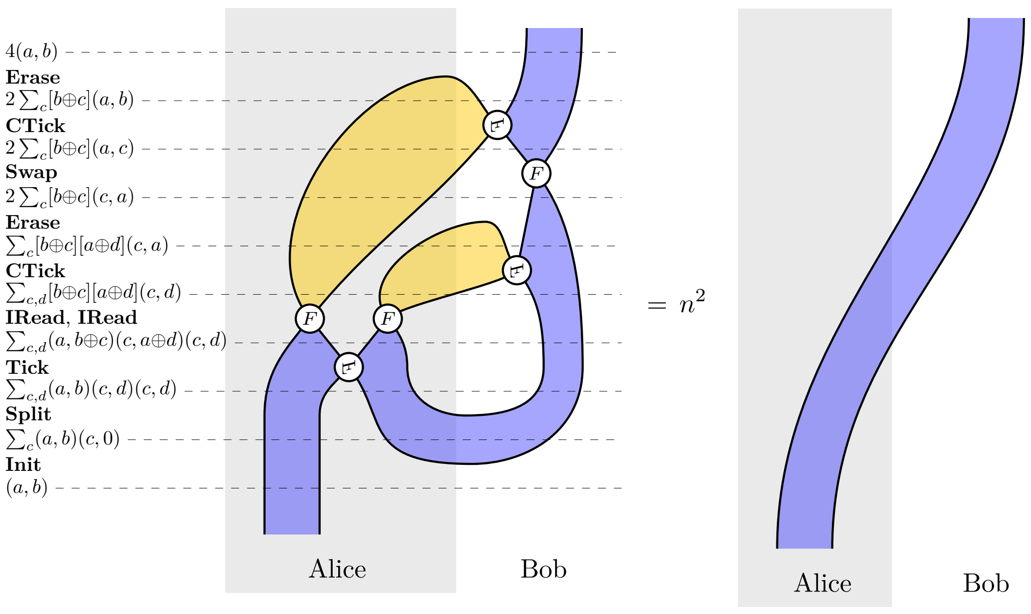

$4(a,b)$
Erase
$2\sum_c[b\oplus c](a,b)$
CTick
$2\sum_c[b\oplus c](a,c)$
Swap
$2\sum_c[b\oplus c](c,a)$
Erase
$\sum_c[b\oplus c][a\oplus d](c,a)$
CTick
$\sum_{c,d}[b\oplus c][a\oplus d](c,d)$
IRead, **IRead**
$\sum_{c,d}(a,b\oplus c)(c,a\oplus d)(c,d)$
Tick
$\sum_{c,d}(a,b)(c,d)(c,d)$
Split
$\sum_c(a,b)(c,0)$
Init
(a,b)
F
$= n^2$
Alice
Bob
Alice
Bob

Figure 13 Teleportation.

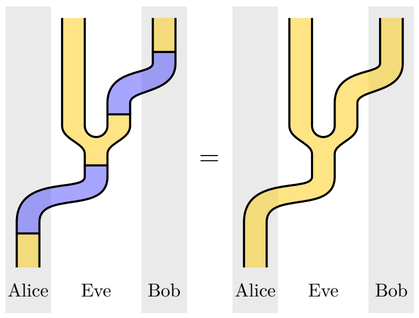
Verification. To verify the protocol in the general case, expand the **CTick** operations using the definitions from Figure 8(a) and (b), then apply equation Figure 8(e) twice. The result is the identity, up to two yellow bubbles, which count the different classical bits that Alice could have obtained.

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 groudits.

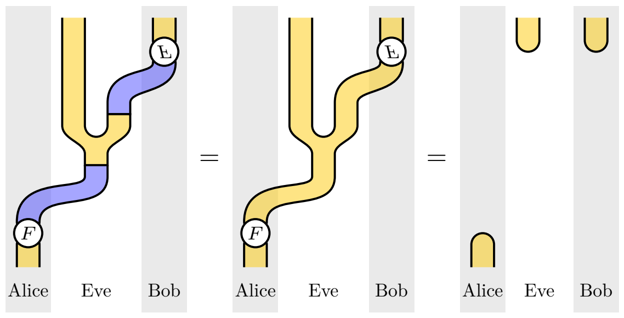
3.5 Key distribution (Figures 14 and 15)

Overview. Quantum key distribution (QKD) [15] is one of the most important protocols in quantum information. Here we describe a classical analogue which can operate on networks of groudits. 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 [12]. We focus here on BB84-style QKD [6]; by dagger pivotality, the E91 variant [14] 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 [6]. 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 $\alpha =$ **Write** or $\alpha =$ **IWrite**. She sends the groubit to Bob, perhaps using a state transfer algorithm (see Section 3.1), but it is intercepted by Eve, who chooses to decode the message using either $\eta =$ **Read** or $\eta =$ **IRead**; having received a classical bit she copies it, and re-encodes a groubit using $\eta^\dagger$, which she sends to Bob. When Bob receives the groubit, he decodes it using $\beta =$ **Read** or $\beta =$ **IRead**.

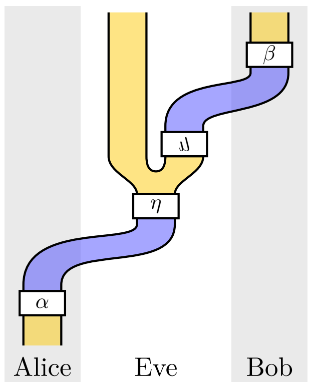


(a) All agents choose the same operation.

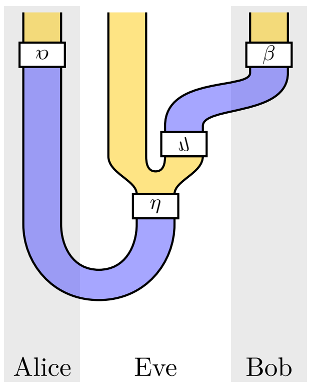


(b) Some agents choose a different operation.

Figure 14 Verification of BB84 quantum key distribution.



(a) BB84-like protocol.



(b) E91-like protocol.

Figure 15 Key distribution protocols.

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 $\alpha = \eta^\dagger = \beta^\dagger =$ **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 operation, the diagram disconnects. We analyze this in Figure 14(b); using the equations of Figures fig:measurement 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.

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.

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.

References

1 Akintomide Adesanmi and Lotfi Mhamdi. Controlling TCP Incast congestion in data centre networks. In *ICCW 2015*. IEEE, 2015. `doi:10.1109/iccw.2015.7247446`.

2 Miriam Backens and Ali Nabi Duman. A complete graphical calculus for Spekkens' toy bit theory. *Foundations of Physics*, 46(1):70–103, 2015. `doi:10.1007/s10701-015-9957-7`.

3 John Baez and James Dolan. From finite sets to Feynman diagrams. In Björn Engquist and Wilfried Schmid, editors, *Mathematics Unlimited*, pages 29–50. Springer, 2001. `arXiv:math/0004133`.

4 John Baez, Alexander Hoffnung, and Christopher Walker. HDA VII: Groupoidification. *Theory and Applications of Categories*, 24(18):489–553, 2010. `arXiv:0908.4305`.

5 Krzysztof Bar and Jamie Vicary. Groupoid semantics for thermal computing, 2014. `arXiv:1401.3280`.

6 Charles H. Bennett and Gilles Brassard. Quantum public key distribution. *IBM Tech. Disc. Bul.*, 28:3153–3163, 1985.

7 Francois Bergeron, Gilbert Labelle, Pierre Leroux, and Margaret Readdy. *Combinatorial Species and Tree-like Structures*. Cambridge University Press (CUP), 1997. `doi:10.1017/cbo9781107325913`.

8 Daniel J. Bernstein. Introduction to post-quantum cryptography. In *Post-Quantum Cryptography*, pages 1–14. Springer Nature, 2009. `doi:10.1007/978-3-540-88702-7_1`.

9 Paul Borrill. The timeless datacentre. *Stanford Colloquium on Computer Systems*, 2016. YouTube:IPTlTmH-YvQ.

10 Nils Carqueville and Ingo Runkel. Orbifold completion of defect bicategories. *Quantum Topology*, 7(2):203–279, 2016. `doi:10.4171/qt/76`.

11 Bob Coecke, Bill Edwards, and Robert W. Spekkens. Phase groups and the origin of non-locality for qubits. *ENTCS*, 270(2):15–36, 2011. `doi:10.1016/j.entcs.2011.01.021`.

12 Bob Coecke and Simon Perdrix. Environment and classical channels in categorical quantum mechanics. *LMCS*, 8(4), 2012. `doi:10.2168/lmcs-8(4:14)2012`.

13 Leonardo Disilvestro and Damian Markham. Quantum protocols within Spekkens' toy model. *Physical Review A*, 95(5), 2017. `doi:10.1103/physreva.95.052324`.

14 Artur Ekert. Quantum cryptography based on Bell's theorem. *Physical Review Letters*, 67:661, 1991. `doi:10.1103/PhysRevLett.67.661`.

15 Romain Alléaume et al. Using quantum key distribution for cryptographic purposes: a survey. *TCS*, 560(1):62–81, 2014. `arXiv:quant-ph/0701168`.

16 Sami Iren, Paul D. Amer, and Phillip T. Conrad. The transport layer: tutorial and survey. *ACM Computing Surveys*, 31(4):360–404, 1999. `doi:10.1145/344588.344609`.

17 Arthur Jaffe, Zhengwei Liu, and Alex Wozniakowski. Holographic software for quantum networks, 2016. URL: `https://arxiv.org/abs/1605.00127`.

18 Vaughan F. R. Jones. Planar algebras, I, 1999. `arXiv:math/9909027`.

19 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. doi:10.1090/s0273-0979-2013-01442-3.

20 André Joyal. Une théorie combinatoire des séries formelles. *Advances in Mathematics*, 42(1):1–82, 1981. doi:10.1016/0001-8708(81)90052-9.

21 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. doi:10.1007/3-540-48071-4_32.

22 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. doi:10.1142/s0129167x14500803.

23 Jeffrey Morton. Categorified algebra and quantum mechanics. *Theory and Applications of Categories*, 16(29):785–854, 2006. arXiv:math/0601458.

24 Michael Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information.* CUP, 2009. doi:10.1017/cbo9780511976667.

25 Adrian Ocneanu. Quantized groups, string algebras, and Galois theory for algebras. In *Operator Algebras and Applications*, pages 119–172. CUP, 1989. doi:10.1017/cbo9780511662287.008.

26 Matthew F. Pusey. Stabilizer notation for Spekkens' toy theory. *Foundations of Physics*, 42(5):688–708, 2012. doi:10.1007/s10701-012-9639-7.

27 David Reutter and Jamie Vicary. Biunitary constructions in quantum information, 2016. arXiv:1609.07775.

28 David Reutter and Jamie Vicary. A classical groupoid model for quantum networks, 2017. arXiv:1707.00966.

29 David Reutter and Jamie Vicary. Shaded tangles for the design and verification of quantum programs, 2017. arXiv:1701.03309.

30 Peter Selinger. A survey of graphical languages for monoidal categories. In *New Structures for Physics*, pages 289–355. Springer, 2010. doi:10.1007/978-3-642-12821-9_4.

31 Robert W. Spekkens. Evidence for the epistemic view of quantum states: A toy theory. *Physical Review A*, 75(3), 2007. doi:10.1103/physreva.75.032110.

32 Umesh Vazirani and Thomas Vidick. Robust device independent quantum key distribution. In *ITCS*, 2014.

33 Jamie Vicary. Higher semantics of quantum protocols. In *Proceedings of LICS*, 2012. doi:10.1109/lics.2012.70.

34 Yongwang Zhao, Zhibin Yang, and Dianfu Ma. A survey on formal specification and verification of separation kernels. *Frontiers of Computer Science*, 2017. doi:10.1007/s11704-016-4226-2.