

Pictures of Processes

Automated Graph Rewriting for Monoidal Categories and
Applications to Quantum Computing



Aleks Kissinger
St. Catherine's College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Michaelmas 2011

Contents

1	Introduction	6
2	Monoidal Categories	18
2.1	Types of Monoidal Categories	21
2.2	Free Monoidal Categories	30
2.3	Formalising Graphical Languages	32
2.3.1	Algebraic Approach: Abstract Tensor Systems	32
2.3.2	Topological Approach: Anchored Graphs	34
3	Algebraic Structures in Monoidal Categories	38
3.1	Bi-algebras and Hopf Algebras	39
3.2	Frobenius Algebras	43
3.2.1	Normal Form for Frobenius Algebras	52
3.2.2	Special and Anti-special Commutative Frobenius Algebras	55
I	Graphical Languages and Rewriting	60
4	Rewrite Systems	61
4.1	Term Rewriting	61
4.2	Graph Rewriting	65
4.3	Adhesive Categories	69
4.4	Partial Adhesive Categories	77
4.4.1	Example: The Category of Simple Graphs	79
4.4.2	Commutation of S -Pushouts and S -Pushout Complements	81
5	String Graphs and Monoidal Theories	85
5.1	String Graphs	86
5.1.1	S -pushouts of String Graphs	90
5.2	Rewriting with String Graphs	93

5.2.1	Wires and Wire-Homeomorphism	94
5.3	Cospans over the Category of String Graphs	96
5.4	Rewriting on Cospans and Rewrite Categories	99
5.5	Free Monoidal Categories	104
II Entanglement, Graphically		113
6	Quantum Information and Entanglement	114
6.1	Quantum Mechanics	114
6.2	Compound Systems and Entanglement	117
6.3	Mixed State Quantum Mechanics	118
6.4	Quantum Computation	120
6.4.1	The Circuit Model	120
6.4.2	Measurement-based Quantum Computation	122
7	Categorical Quantum Mechanics	124
7.1	Compact Categories and Teleportation	125
7.2	Complementary Observables as Frobenius Algebras	126
7.3	The Z/X Calculus and Quantum Computation	133
7.3.1	Example: Building and Rewriting Circuits	135
8	Monoidal Algebra in Quantum Entanglement Theory	137
8.1	Classifying Entanglement	137
8.1.1	Symmetric States	144
8.2	Strong SLOCC-maximality and strong symmetry	145
8.3	Frobenius States and their Induced Frobenius Algebras	149
8.3.1	Classification of Qubit Frobenius States	150
8.3.2	Classification of Frobenius States for Higher Dimensions	152
8.4	A Graphical Theory for Entanglement	153
8.4.1	Symmetric Modules of an SCFA and Distributivity	156
8.4.2	Universality	158
8.4.3	Arithmetic on the Complex Projective Line	159
III Automation		162
9	Automating String Graph Rewriting: Quantomatic	163
9.1	Conjecture Synthesis and QuantoCoSy	164

9.1.1	Conjecture Synthesis for Terms	167
9.1.2	Adapting Conjecture Synthesis to String Graphs	168
10	Conclusion	171
10.1	Future Work	172
10.1.1	Classifying Frobenius states	172
10.1.2	Super-qubits and the W state bialgebra	173
10.1.3	GW-pairs and strongly complementary observables	174
10.1.4	Knuth-Bendix completion for string graphs	175
10.1.5	Pattern graphs and graphical inductive reasoning	177

Abstract

This work is about diagrammatic languages, how they can be represented, and what they in turn can be used to represent. More specifically, it focuses on representations and applications of string diagrams. String diagrams are used to represent a collection of processes, depicted as “boxes” with multiple (typed) inputs and outputs, depicted as “wires”. If we allow plugging input and output wires together, we can intuitively represent complex compositions of processes, formalised as morphisms in a monoidal category.

While string diagrams are very intuitive, existing methods for defining them rigorously rely on topological notions that do not extend naturally to automated computation. The first major contribution of this dissertation is the introduction of a discretised version of a string diagram called a *string graph*. String graphs form a partial adhesive category, so they can be manipulated using double-pushout graph rewriting. Furthermore, we show how string graphs modulo a rewrite system can be used to construct free symmetric traced and compact closed categories on a monoidal signature.

The second contribution is in the application of graphical languages to quantum information theory. We use a mixture of diagrammatic and algebraic techniques to prove a new classification result for strongly complementary observables. Namely, maximal sets of strongly complementary observables of dimension D must be of size no larger than 2, and are in 1-to-1 correspondence with the Abelian groups of order D . We also introduce a graphical language for multipartite entanglement and illustrate a simple graphical axiom that distinguishes the two maximally-entangled tripartite qubit states: GHZ and W. Notably, we illustrate how the algebraic structures induced by these operations correspond to the (partial) arithmetic operations of addition and multiplication on the complex projective line.

The third contribution is a description of two software tools developed in part by the author to implement much of the theoretical content described here. The first tool is Quantomatic, a desktop application for building string graphs and graphical theories, as well as performing automated graph rewriting visually. The second is QuantoCoSy, which performs fully automated, model-driven theory creation using a procedure called conjecture synthesis.

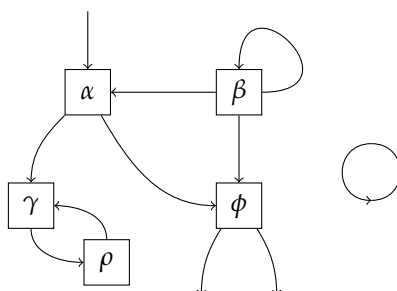
Chapter 1

Introduction

Quantum information theory is the study of how data can be encoded and manipulated using microscopic systems subject to quantum effects. Over the past two decades, it has grown into a large and diverse field, with applications in security, where quantum effects are used to design “unlistenable” data channels, foundations of physics, where fundamental principles of information are used to derive physical theories, and perhaps most notably quantum computing, where classically intractable computations such as factorisation of huge numbers can happen in the blink of an eye. Virtually all of these applications use quantum theory exactly as John von Neumann described it in 1932. However, amidst the increasing scale the problems considered, it becomes clear that this is analogous to writing complex computer programs using circuit diagrams. As in the case with software development, abstracting away from the low-level is crucial to progress.

In this dissertation, we seek out this abstraction by identifying and exploiting the behaviour of graphical representations of quantum systems. We develop a tool set for graphical reasoning by drawing a connection between categorical algebra and graph rewriting. We then show how these this tool set can be applied to the description of quantum phenomena using the language of *string diagrams*.

String diagrams consist of boxes, which represent processes (physical, logical, algebraic, ...) that have some inputs and some outputs. Some of those inputs and outputs can be connected together using wires.



The only real requirement we impose on string diagrams is that their “value” (typically as some sort of map, relation, or process) is unaffected by topological deformations. Due to the strongly physical and spatial qualities of string diagrams, it should come as no surprise that they were originally formulated by a physicist. String diagrams originated with Roger Penrose in 1971 [58] as an alternative notation for contractions of what he called *abstract tensors*, which are essentially just morphisms with some named inputs and outputs. Furthermore, the idea of representing spatially and temporally composed processes using these types of diagrams dates back at least to the 1948 advent of Feynman diagrams [36].

String diagrams make sense for any mathematical structure that has a well-behaved notion of horizontal (i.e. spatial) and vertical (i.e. temporal) composition. A very general way to formalise such structures is to use monoidal categories, which were introduced by Mac Lane [48] to describe a wide variety of categories admitting associative, product-like structures (e.g. cartesian products, direct sums, tensor products).

A connection between the notions of string diagrams and monoidal categories was inevitable. Twenty years after the introduction of string graphs, Joyal and Street [35] formalised this idea by using string diagrams (considered as topological graphs with extra structure) to build *free monoidal categories*. Intuitively, a “free X ” is an object for which the axioms of an “ X ” are true, but nothing else. So, a free monoidal category is a monoidal category where two morphisms are equal *if and only if* they are equal by the axioms of a monoidal category. In other words, string diagrams, compared up to topological deformations (of a particular kind) exactly represent morphisms compared up to the axioms of a monoidal category.

A monoidal category is a category equipped with a bifunctor $\otimes : \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V}$ that is associative, up to isomorphism and has a left and right unit $I \in \text{ob}\mathcal{V}$. There are many notions of monoidal categories with additional structure that have an extremely wide variety of applications in areas such as the study of braids and knots, linear algebra and representation theory, quantum field theory, higher-dimensional algebra, enriched and internal category theory, homotopy theory, linear logic, and programming language semantics. We introduce a few of these extended notions of monoidal category in chapter 2, namely strict and non-strict (planar) monoidal categories, braided and symmetric monoidal categories, symmetric traced categories, left- and right-autonomous categories, compact closed categories, \dagger -monoidal (pronounced dagger-monoidal) categories, and \dagger -compact closed categories. We offer a summary of what is known about the relationships between these kinds of categories, coherence results, and most importantly, graphical language theorems. A much more comprehensive collection of graphical language definitions, as well as the state of the art in what is and is not known about these languages is available in Selinger’s excellent survey paper [62].

We also review the notion of abstract tensor networks, roughly in the form it was introduced by Penrose in [58] and relate its formulation to monoidal categories and the topological graphical languages introduced by Joyal and Street.

One of the most useful aspects of a monoidal category is that it allows one to define algebraic structures *internal* to a monoidal category. That is, an algebraic structure can be defined as a collection of morphisms in some monoidal category satisfying some axioms. Since such a definition only relies on the structure of a monoidal category, it makes sense in *any* monoidal category. For instance, one can define a monoid in \mathcal{V} as a triple (A, μ, η) where A is an object and $\mu : A \otimes A \rightarrow A$ and $\eta : I \rightarrow A$ are two morphisms satisfying some equations (namely, associativity and unit laws). A monoid in the category of sets and functions is just the usual notion of a monoid, i.e. a unital semigroup. In the category of vector spaces and linear maps, it is a unital, associative algebra. In the opposite category, it is a counital, coassociative coalgebra. In the category of categories and functors, it is a (strict) monoidal category, justifying the intuition that a monoidal category is just a “categorified” monoid. In chapter 3, we define algebraic structures internal to a monoidal category and give various examples that will be used throughout this dissertation. These include monoids, commutative monoids, comonoids, Frobenius algebras, bialgebras, and Hopf algebras. We also provide many concrete examples of these algebraic structures, as they occur in familiar (and some less-familiar) categories.

Building on this background, the bulk of the thesis is organised into two roughly independent parts. The first part is about applying techniques from the theory of graph rewriting to string diagrams. The second part is about applying monoidal category theory and graphical languages to the study of quantum mechanics. In particular, diagrams are used to study quantum computing and quantum entanglement theory. A third, shorter part focuses on implementing the theoretical work from the previous two parts in a program called Quantomatic.

Part I opens with an introduction to rewrite systems in chapter 4. Rewrite systems provide a very general means of reasoning systematically about algebraic theories. In fact, this reasoning is so systematic that it can be done by a computer. Rewriting lives at the heart of most computer algebra systems (CAS), automated reasoning tools, and proof assistants. The idea behind rewriting is very simple. Rather than considering *equations* ($s = t$) between terms, as one typically does in (universal) algebra, one considers directed reductions called *rewrite rules* ($s \rightarrow t$). The application of rewrite systems from an algebraists point of view is that they can help solve *word problems*.

A word problem is a question of the form, “Is term s equivalent to term t by the axioms of an equational theory E ?” It is well known that word problems are not decidable in general. However, given a suitably nice algebraic theory and some elbow grease, it often *is* possible to solve a word problem by turning E into a rewrite system R . If we end up with a nice enough rewrite sys-

tem, we can solve the word problem by rewriting s repeatedly until no rule from R applies (called normalising s), doing the same to t , and comparing the two results to see if they are equal.

If we just randomly pick directions for each of the equations in E , this technique is very unlikely to work. However, if we can find a nice rewrite system (i.e. one that is terminating and confluent), normal forms always exist and are unique, and there is an evident solution to the word problem for all terms. Thus a large portion of the rewriting literature is about how to go about turning sets of equations into nice rewrite systems, turning ill-behaved rewrite systems into better ones, and coping with ill-behaved systems using more sophisticated strategies than “normalise and compare”.

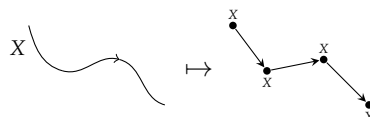
Rewrite systems are not just restricted to terms. Just as term rewriting can be thought of as replacing certain subtrees (corresponding to subterms) with other trees, we can consider replacing certain subgraphs with another graph. This is called *graph rewriting*. In 1973, Ehrig, Pfender, and Schneider introduced the double pushout (DPO) approach to graph rewriting [27]. We explain this technique in detail and with examples in section 4.2. While DPO rewriting can be formulated in many categories (including the category of sets, the category of graphs, and any topos), DPO rewriting is not well-defined in all categories with pushouts. In 1979, Ehrig and Kreowski identified certain abstract properties of a category with pushouts that allow one to do double-pushout rewriting [29]. One abstract formulation of categories in which DPO rewriting makes sense are adhesive categories, introduced by Lack and Sobociński in 2003 [43]. In these categories, pushouts involving monomorphisms behave like coverings in the sense that they form so-called *van Kampen Squares*. All toposes are adhesive categories, and in a recent result [42], Lack showed that any adhesive category embeds fully and faithfully in a topos, and that embedding preserves all the adhesive structure. So, another way to think of adhesive categories is “categories where pushouts of monomorphisms behave as they do in toposes”.

In section 4.4, we generalise adhesive categories to *partial adhesive categories*. These are categories \mathcal{C} that embed fully and faithfully in an adhesive category $S : \mathcal{C} \rightarrow \mathcal{A}$, such that S preserves monomorphisms. Intuitively, these are categories whose objects are the objects of an adhesive category (e.g. directed graphs) that satisfy certain additional axioms (e.g. simple graphs: at most one edge connecting any vertex to another). We then illustrate that the adhesive-like properties of pushouts in \mathcal{A} are inherited by the S -pushouts in \mathcal{C} (i.e. the pushouts in \mathcal{C} that exist and are preserved by S). As a result, DPO rewriting is well-defined for partial adhesive categories as long as one restricts to certain matching morphisms called S -matchings.

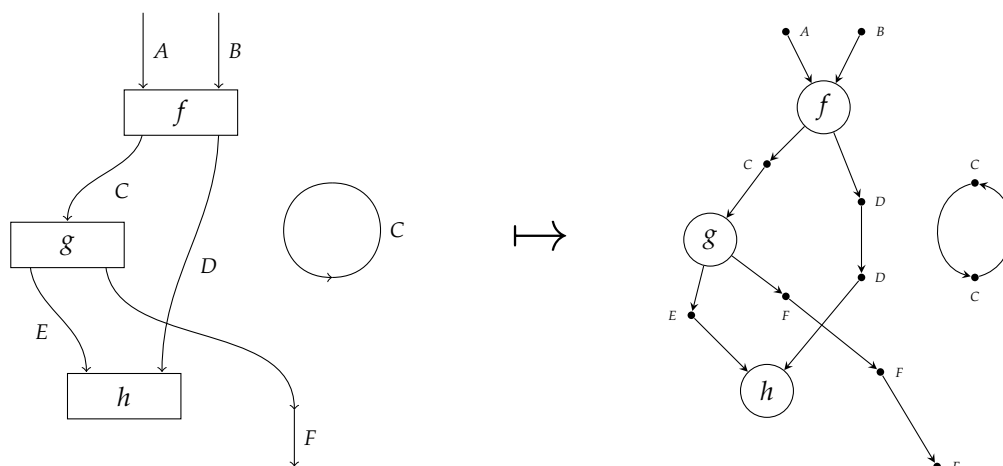
Graph rewriting can be applied to string diagrams, but not directly. This is for the simple reason that string diagrams are not graphs in a strict sense. The wires in string diagrams need not be connected to boxes at both ends. They can even be connected to themselves to form circles. Wires that are not connected to a box at their source serve as inputs for string diagrams, and wires that are not connected at their target serve as outputs. Wires that are not connected to a box at either end

are called *free wires*, and represent identity maps. A directed graph G consists of a set of vertices V_G and a set of edges E_G , as well as *total* functions $s, t : E_G \rightarrow V_G$. Therefore we cannot represent string diagrams as digraphs. Even if we relax the requirement that s and t be total functions, there is no way to distinguish circles from free wires.

We solve this problem by defining *string graphs*. These are typed graphs whose vertices fall into two categories: wire-vertices and box-vertices. The wires in string diagrams are replaced by chains of wire-vertices.

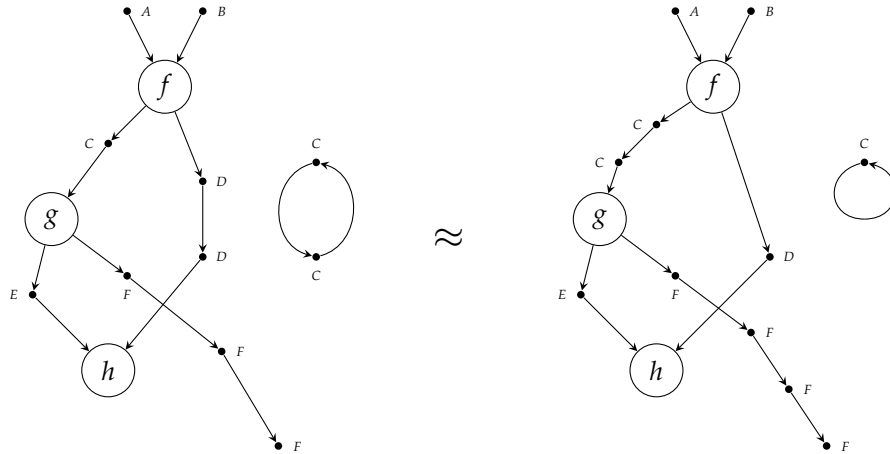


Representing boxes as box-vertices, we can translate string diagrams into string graphs.



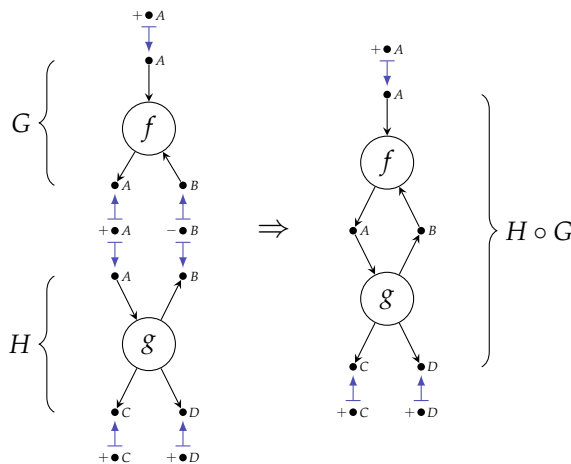
In chapter 5, we define the partial adhesive category of string graphs and string graph homomorphisms. We also define special pushouts called *pluggings*, which are used to plug the outputs of one string graph into the inputs of another string graph to form the composed graph. We also define string graph rewrite rules in such a way that any monomorphism is an S -matching. Therefore double-pushout rewriting is always well-defined.

The vigilant reader will notice that the correspondence between string diagrams and string graphs is *nearly* 1-to-1. The only obstacle is that wires in string diagrams can be converted to chains of wire-vertices of any length. To eliminate this redundancy, we consider two string graphs to be equivalent if the only difference between the two is the length of the wires.



This corresponds to the wires of the associated string diagrams being homeomorphic, when considered as subspaces of topological graphs. For that reason, this equivalence relation is called *wire-homeomorphism*. We define wire-homeomorphism using a confluent, terminating rewrite system on string graphs in section 5.2.1.

Like their topological counterparts, string graphs can be used to construct free monoidal categories. We do this by defining a framed cospan construction over the category of string graphs. Recall that for any category \mathcal{C} with pushouts, the bicategory of cospans $\text{Csp}(\mathcal{C})$ has as objects the objects of \mathcal{C} , 1-morphisms cospans $X \xrightarrow{f} F \xleftarrow{g} Y$, and 2-morphisms cospan homomorphisms. Composition is performed by pushout, and identities are cospans of identity maps. We form the category of framed cospans of string graphs by restricting the objects in the cospan construction to discrete graphs consisting of wire-vertices and the cospans $X \rightarrow G \leftarrow Y$ to maps covering the inputs and outputs of G . Composition by pushout then reduces to the intuitive notion of plugging together string graphs.



In section 5.5, we show that the free symmetric traced category and the free compact closed category on a monoidal signature can be constructed as a category of framed cospans of string graphs. By shifting from a topological representation to a combinatoric one, these morphisms can be represented straightforwardly on a computer, and they can be manipulated using automated

graph rewriting techniques. This is explored in Part III, following the introduction of graphical theories for quantum computing.

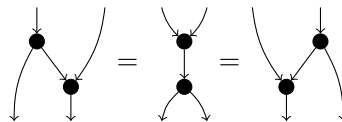
Part II describes in detail two graphical theories that are of particular interest for quantum computing. These theories were formulated in the context of Categorical Quantum Mechanics (CQM), a program initiated by Abramsky and Coecke in 2004 [2, 3] whose purpose was to study quantum phenomena from the point of view of monoidal category theory. More than any one particular result, CQM represents a set of principles and an approach to the study of quantum theory. In this approach, compositionality is at the forefront. CQM asserts that all of the interesting and important aspects of quantum theory can be witnessed by studying systems and processes and the ways in which they compose. It emphasises the role of compound systems, information flow, and diagrammatic reasoning while de-emphasising the role of Hilbert spaces as a crucial component to the understanding of quantum phenomena.

This part opens with a brief introduction for the non-physicist to quantum mechanics, quantum computing, and quantum information theory in chapter 6. Chapter 7 introduces categorical quantum mechanics and illustrates the role of monoidal categories in quantum teleportation and the study of complementary observables. The latter was explored in detail by Coecke and Duncan in [15]. In quantum mechanics, an observable comes with a basis of orthonormal eigenstates corresponding to measurement outcomes. Two non-degenerate observables O and O' are called complementary if their associated bases of eigenstates are mutually unbiased. That is, for bases $\{|u_i\rangle\}, \{|v_j\rangle\}$ and for all i, j (where D is the dimension of the space):

$$|\langle u_i | v_j \rangle|^2 = \frac{1}{D}$$

Intuitively, if we measure a quantum state in an eigenstate of O with respect to the O' observable, we are equally likely to get any outcome. That is, maximal knowledge of a state with respect to O implies minimal knowledge with respect to O' . A familiar example of complementary observables is position and momentum.

Mutually unbiased bases can be understood algebraically using particular kinds of interacting Frobenius algebras. Frobenius algebras in a monoidal category consist of a monoid $(A, \bullet, \blacktriangledown)$ and a comonoid $(A, \blacktriangledown, \bullet)$ satisfying the *Frobenius identity*.



\dagger -Frobenius algebras are Frobenius algebras whose comonoid structure is the adjoint of the monoid structure, i.e. $\blacktriangledown = (\blacktriangledown)^\dagger$ and $\bullet = (\bullet)^\dagger$.

Commutative \dagger -Frobenius algebras have attracted attention in recent years for exhibiting precisely the identities of 2-dimensional cobordisms, whose understanding is a crucial stepping stone to the formulation of topological quantum field theories. For more details, see e.g. [5, 41].

Special Frobenius algebras satisfy an additional identity on the loop map.

As the name suggests, \dagger -special commutative Frobenius algebras (\dagger -SCFAs) are commutative \dagger -Frobenius algebras that are special. Coecke, Pavlovic, and Vicary showed that orthonormal bases over finite-dimensional complex Hilbert spaces are in 1-to-1 correspondence with \dagger -SCFAs. So, rather than studying mutually unbiased bases themselves, we can study their associated \dagger -SCFAs. From this point of view, the mutually unbiased basis condition can be summed up in a simple graphical identity, where $(\downarrow, \uparrow, \downarrow, \uparrow)$ is the \dagger -SCFA induced by an orthonormal basis and $(\downarrow, \uparrow, \downarrow, \uparrow)$ is the \dagger -SCFA induced by another, mutually unbiased basis.

(1.1)

In [15], the Coecke and Duncan introduced several stronger forms of complementarity. One example is the case where the induced algebras of the two bases extend to a bialgebra. That is, the following equations are satisfied.

(1.2)

In this dissertation, we refer to a pair of observables whose bases satisfy this condition as *strongly complementary* observables. Coecke and Duncan showed that under certain additional assumptions, the equations in (1.2) imply (1.1) in an arbitrary compact closed category. In section 7.2, we simplify this result in the case of the category of finite-dimensional Hilbert spaces by providing a new proof that (1.2) *always* implies (1.1). We also provide a new classification result for pairs of strongly complementary observables.

Theorem. *Strongly complementary pairs of observables in a Hilbert space of dimension D are in 1-to-1 correspondence with the finite Abelian groups of order D .*

Furthermore, we show that it is impossible for three distinct observables to be pairwise strongly complementary. This then classifies maximal sets of strongly complementary observables for all dimensions.

In chapter 8, we turn to the application of diagrammatic techniques in the study of multipartite entanglement. The classification, computation, and manipulation of complex, many-body entangled quantum states is one of the most difficult problems facing quantum physicists and quantum

information theorists. Any naïve approach to the problem of classifying multipartite entanglement is doomed to fail, and brute-force calculations involving many entangled quantum systems are untenable on today’s computers. This suggests the need for more sophisticated techniques that capture and exploit as many symmetries and fundamental structure within a quantum system as possible. Rather than studying a multipartite state as a single, monolithic entity, we study it in terms of its components and explore how those components interact. We call this the *compositional approach* to multipartite entanglement.

By way of the Choi-Jamiołkowski isomorphism, we can consider quantum states and processes on the same footing. In that sense, a bipartite quantum state in $H \otimes H$ can be thought of as a quantum channel from H to H . Similarly, we can treat a tripartite state as a map from $H \otimes H$ to H , i.e. a *binary operation* on quantum states in H . Nearly all algebraic objects of interest (e.g. groups, rings, vector spaces) are sets equipped with one or more binary operations satisfying certain axioms. For that reason, we adopt a motto: “Just as binary operations have a special status in the study of algebra, so too should tripartite states in the study of multipartite entanglement.”

To justify this assertion, we develop a methodology for representing and studying arbitrary qubit states using tripartite states as building blocks. It is a well-known result from quantum entanglement theory that there exist two canonical, genuinely-entangled tripartite states over qubits, up to equivalence by stochastic local operations and classical communication [26]. These states are the Greenberger-Horne-Zeilinger (GHZ) state and the W state.¹

$$|GHZ\rangle = \frac{1}{\sqrt{2}} (|000\rangle + |111\rangle) \quad |W\rangle = \frac{1}{\sqrt{3}} (|100\rangle + |010\rangle + |001\rangle)$$

In section 8.2, we identify two properties shared by GHZ and W states, which we call *strong symmetry* and *strong SLOCC-maximality*. Strongly symmetric states are symmetric states that extend naturally to larger symmetric states on any number of systems. For instance, the N -partite versions of GHZ and W are defined as:

$$\begin{aligned} |GHZ_N\rangle &:= |00\dots 0\rangle + |11\dots 1\rangle \\ |W_N\rangle &:= |10\dots 0\rangle + |010\dots 0\rangle + \dots + |0\dots 01\rangle \end{aligned}$$

SLOCC-maximal states are states that are maximal with respect to conversion by stochastic local operations and classical communication. That is, $|\Psi\rangle$ is SLOCC-maximal precisely when any state $|\Psi'\rangle$ that can be converted into $|\Psi\rangle$ by way of a SLOCC protocol must already be SLOCC-equivalent to $|\Psi\rangle$. Strongly SLOCC-maximal states are multipartite states that are *inductively* SLOCC-maximal. That is, $|\Psi\rangle$ is a strongly SLOCC-maximal N partite state if it is SLOCC-maximal and it is possible to obtain a strongly SLOCC-maximal $(N - 1)$ partite state from $|\Psi\rangle$ by projecting out any of the subsystems H .

¹W states were also first introduced by Greenberger, Horn, and Zeilinger in 1991 [66], but they were not named until Dür, Vidal, and Cirac highlighted their significance in 2000 [26]. It is generally believed that the W is for Wolfgang (Dür) [13].

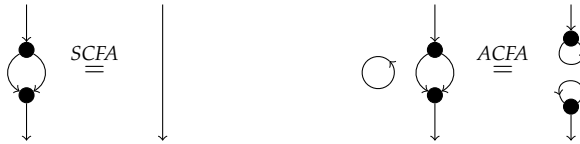
States that satisfy both of these conditions are called *Frobenius states*. In section 8.3, we show that any Frobenius state extends to commutative Frobenius algebra. Conversely, commutative Frobenius algebras can be used to construct a Frobenius state. The commutative Frobenius algebra \mathcal{G} associated with the Frobenius state $|GHZ\rangle$ is:

$$\begin{aligned} \begin{array}{c} \circlearrowleft \\ \downarrow \end{array} &= |0\rangle\langle 00| + |1\rangle\langle 11| & \begin{array}{c} \circlearrowright \\ \downarrow \end{array} &= \sqrt{2} |+\rangle = |0\rangle + |1\rangle \\ \begin{array}{c} \circlearrowright \\ \downarrow \end{array} &= |00\rangle\langle 0| + |11\rangle\langle 1| & \begin{array}{c} \circlearrowleft \\ \downarrow \end{array} &= \sqrt{2} \langle +| = \langle 0| + \langle 1| \end{aligned}$$

For the Frobenius state $|W\rangle$, the associated Frobenius algebra \mathcal{W} is:

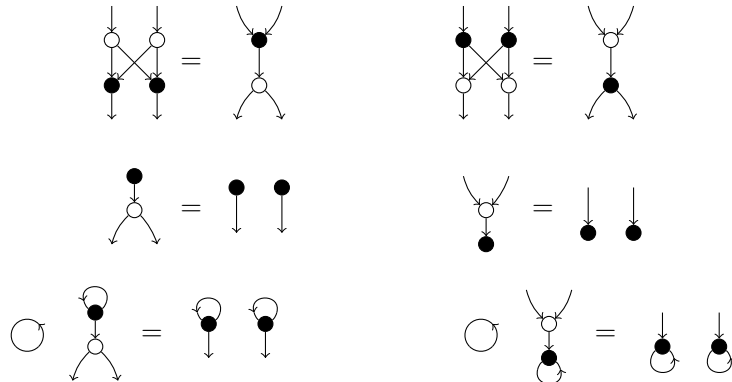
$$\begin{aligned} \begin{array}{c} \bullet \\ \downarrow \end{array} &= |1\rangle\langle 11| + |0\rangle\langle 01| + |0\rangle\langle 10| & \begin{array}{c} \bullet \\ \downarrow \end{array} &= |1\rangle \\ \begin{array}{c} \bullet \\ \downarrow \end{array} &= |00\rangle\langle 0| + |01\rangle\langle 1| + |10\rangle\langle 1| & \begin{array}{c} \bullet \\ \downarrow \end{array} &= \langle 0| \end{aligned}$$

In [19], Coecke and Kissinger produced a unique characterisation of GHZ and W states in terms of properties of their associated Frobenius algebras, which is summarised in section 8.3.1. We highlight two types of commutative Frobenius algebras based on the value of the “loop map”.



We prove that commutative Frobenius algebras are *special* if and only if their associated tripartite states are SLOCC-equivalent to GHZ. Similarly, commutative Frobenius algebras are *anti-special* if and only if their associated tripartite states are SLOCC-equivalent to W. Thus, the two canonical tripartite qubit states can be distinguished by two simple graphical identities.

Taking inspiration from the interaction properties of GHZ and W states, we define a *GW-pair* as a pair of commutative Frobenius algebras (one special, one anti-special) satisfying certain graphical identities.



We prove that the axioms of a GW-pair subsume the axiomatisation for GHZ and W states given in [19]. Again inspired by the example of the GHZ and W states, we introduce the notion of

a *distributive GW-pair*. Distributive GW-pairs behave similarly to rings, in that the “multiplication” induced by $\begin{array}{c} \diagup \quad \diagdown \\ \circ \\ \downarrow \end{array}$ distributes, up to a scalar factor, over the “addition” defined by $\begin{array}{c} \diagdown \quad \diagup \\ \bullet \\ \downarrow \end{array}$.

$$k \begin{array}{c} \triangle a \quad \triangle b \quad \triangle c \\ \downarrow \quad \downarrow \quad \downarrow \\ \downarrow \quad \bullet \\ \downarrow \end{array} = \begin{array}{c} \triangle a \quad \triangle a \quad \triangle b \quad \triangle c \\ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ \circ \quad \circ \\ \downarrow \quad \bullet \\ \downarrow \end{array} \quad (1.3)$$

Using a GW-pair, we can construct the abstract analogue of a CNOT gate, and verify graphically that it behaves as a CNOT. For the specific GW-pair defined by $|GHZ\rangle$ and $|W\rangle$, this is actually a CNOT gate. Using this fact, we prove that the generators of the pair $(\mathcal{G}, \mathcal{W})$ are universal for quantum computation. By Choi-Jamiołkowski, this means they can also be used to construct arbitrary multipartite qubit states.

The precise sense in which the GHZ algebra behaves like multiplication and the W algebra behaves like addition is explained in section 8.4.3. Qubits defined on the Bloch sphere can equivalently be considered as points on the complex projective line \mathbf{CP}^1 . We can define (partial) addition and subtraction operations on the points in \mathbf{CP}^1 , considered as the set \mathbf{C} with an additional point at infinity.

$$\begin{array}{ll} k \cdot \infty = \infty & k + \infty = \infty \\ 0 \cdot \infty = \perp & 0 + \infty = \infty \\ \infty \cdot \infty = \infty & \infty + \infty = \perp \end{array}$$

The GHZ algebra corresponds to the multiplication operation on \mathbf{CP}^1 , and the W algebra corresponds to addition. For finitary elements, multiplication distributes over addition. The failure of distributivity for ∞ in \mathbf{CP}^1 (i.e. distributivity up to a non-zero scalar) is reflected in equation (1.3) by the fact that $k = 0$ when $|a\rangle = |\infty\rangle$.

There is still much to be learned about the algebras induced by quantum states, but already the compositional approach has yielded insights about the GHZ and W states that would not have been possible otherwise. One could picture quantum algorithms or protocols that leverage the behavioural qualities identified in this dissertation. For instance, treating inputs to graphs consisting of $\begin{array}{c} \diagdown \quad \diagup \\ \bullet \\ \downarrow \end{array}$ and $\begin{array}{c} \diagup \quad \diagdown \\ \circ \\ \downarrow \end{array}$ as variables, we can think of such graphs as encoding polynomials in quantum states. Work is in progress to apply this insight to the development of quantum algorithms for hard problems such as finding the roots of diophantine polynomials.

In Part III, we introduce *Quantomatic* and *QuantoCoSy*, which are software tools for working with string graphs. *Quantomatic* allows a user to create and modify string graphs, graphical theories, and string graph rewrite systems. It also lets one selectively apply rewrite rules and normalise graphs with respect to a rewrite system. *QuantoCoSy* is a tool for synthesising *new* graphical theories from concrete models using a technique called *conjecture synthesis*. This procedure, introduced by Johansson, Dixon, and Bundy in 2010 [33], is a procedure for enumerating and checking equality

for all terms of a certain size in an algebraic theory. The thing that makes this technique so effective is it builds a rewrite system dynamically during the enumeration procedure and actively avoids checking for redundant equalities, i.e. those that are already derivable using the rules it has discovered previously. It does this by only enumerating terms that are *irreducible* with respect to a rewrite system. QuantoCoSy adapts this technique from term rewrite systems to string graph rewrite systems, and is showing potential to be a valuable tool in the generation of graphical theories from concrete, linear algebraic models.

In chapter 10, we review the major results of the dissertation and discuss future work, particularly in the area of automation.

Chapter 2

Monoidal Categories

It is often useful to reason in a very general sense about processes and how they compose. Category theory provides the tool to do this. A category consists of a collection of objects A, B, C, \dots , a collection morphisms f, g, \dots , an associative operation \circ for (vertical) composition, and for every object A an identity morphism 1_A . Objects can be thought of as types. They dictate which morphisms can be composed together. We shall primarily be interested in categories that have not only a vertical composition operation, but a horizontal composition as well. Such categories are called monoidal categories.

Definition 2.0.1. A *monoidal category* consists of a category \mathcal{V} , an object $I \in \mathcal{V}$ called the monoidal unit, a bifunctor $\otimes : \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V}$ called the monoidal product, and natural isomorphisms $\alpha_{A,B,C} : A \otimes (B \otimes C) \rightarrow (A \otimes B) \otimes C$, $\lambda_A : I \otimes A \rightarrow A$, and $\rho_A : A \otimes I \rightarrow A$, such that $\lambda_I = \rho_I$ and the following diagrams commute:

$$\begin{array}{ccccc}
 A \otimes (B \otimes (C \otimes D)) & \xrightarrow{\alpha} & (A \otimes B) \otimes (C \otimes D) & \xrightarrow{\alpha} & ((A \otimes B) \otimes C) \otimes D \\
 \downarrow A \otimes \alpha & & & & \uparrow \alpha \otimes D \\
 A \otimes ((B \otimes C) \otimes D) & \xrightarrow{\alpha} & & \xrightarrow{\alpha} & (A \otimes (B \otimes C)) \otimes D
 \end{array} \tag{2.1}$$

$$\begin{array}{ccc}
 A \otimes (I \otimes B) & \xrightarrow{\alpha} & (A \otimes I) \otimes B \\
 \searrow A \otimes \lambda & & \swarrow \rho \otimes B \\
 & A \otimes B &
 \end{array} \tag{2.2}$$

We shall refer to $(\otimes, \alpha, \lambda, \rho)$ as the *monoidal structure* of \mathcal{V} . We often drop α , λ , and ρ when they are clear from the context. Monoidal categories where all three natural isomorphisms are actually equalities are called *strict* monoidal categories.

Examples 2.0.2. The condition of being a monoidal category is very weak. Most categories of interest admit at least one monoidal structure, and many admit several. Some examples:

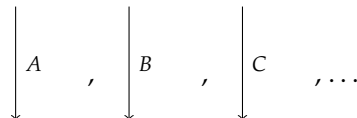
- $(\mathbf{Set}, \times, 1)$: the category of sets and total functions with the cartesian product \times and the one-element set 1 make \mathbf{Set} into a monoidal category.
- Disjoint union $+$ and the empty set form another monoidal structure on \mathbf{Set} .
- More generally, any category with finite products or coproducts is monoidal.
- $(\mathbf{Vect}_K, \otimes, K)$: The category of K -vector spaces and K -linear maps is monoidal, with monoidal product taken as tensor product of vector spaces and tensor unit K , the 1-dimensional space.
- $(\mathbf{FVect}_K, \otimes, K)$: The same as above, but restricted to finite-dimensional vector spaces.
- $(\mathbf{Mat}(K), \otimes, 1)$: The category whose objects are natural numbers and whose arrows $M : m \rightarrow n$ are $n \times m$ matrices taking values in K . Composition is matrix multiplication, the monoidal product is multiplication of natural numbers (on objects) and the Kronecker product of matrices (on arrows). This category is essentially \mathbf{FVect}_K , with a chosen basis for all of its objects.
- $(\mathbf{Rel}, \times, 1)$: the category of sets and relations. Note that the cartesian product \times is a monoidal product, but not a product in the categorical sense.
- $(\mathbf{Rel}, \oplus, \{\})$: where \oplus is the disjoint union of sets (on objects) and the disjoint union of relations (on arrows). We write the disjoint union using the \oplus symbol to highlight the fact that it is actually a *biproduct* in \mathbf{Rel} . As such, it is automatically a monoidal product.

In any monoidal category, α , λ , and ρ can be used to construct a natural isomorphism from some object to any other bracketing of that object, with or without monoidal units. E.g.

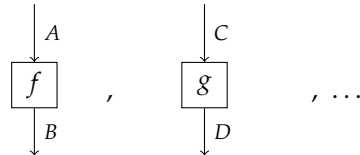
$$(A \otimes I) \otimes (B \otimes (I \otimes C)) \cong (A \otimes (B \otimes (C \otimes I)))$$

It was shown by Mac Lane that the equations in Definition 2.0.1 suffice to show that *any* such natural isomorphism is equal to any other one [48]. Such a theorem is known as a *coherence theorem*, and it was the first of many concerning monoidal categories. By a minor abuse of notation, we shall often treat monoidal categories as if they were strict. That is, we often omit brackets, α , λ , and ρ , simply assuming they are included where necessary. Coherence assures us that we can omit these details without ambiguity.

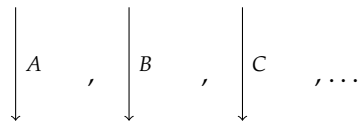
Though we shall occasionally use normal, algebraic notation for morphisms in monoidal categories, it is often vastly more convenient to use a graphical notation. In fact, the majority of this dissertation concerns formalising and exploiting graphical notation. For now, we shall treat this notation informally and fill in the details later. We represent objects as labelled wires:



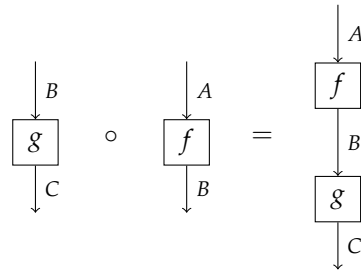
Morphisms can be thought of as processes. A morphism takes something of type A and produces something of type B . For that reason, we'll draw morphisms as *boxes* with a wire coming in labelled with a morphism's input type and a wire going out labelled with a morphism's output type.



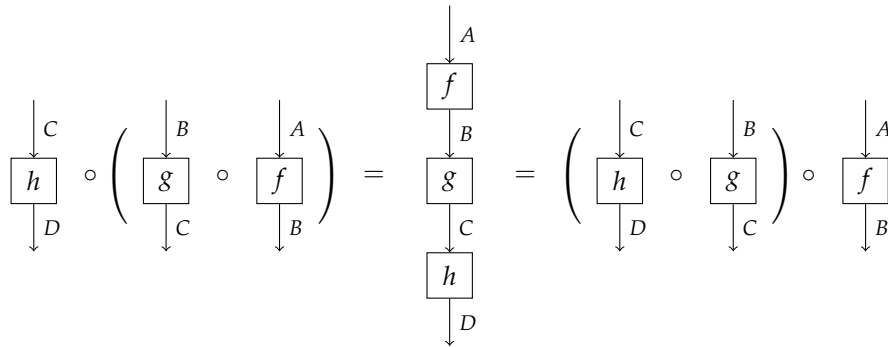
Identity morphisms are special "do nothing" processes, which take something of type A and return the thing itself. We represent these as empty wires.



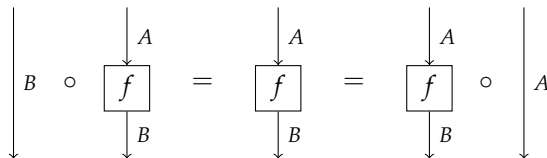
Morphisms are composed by plugging an output wire into an input wire.



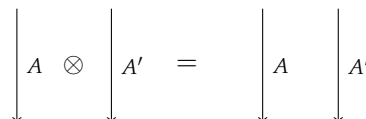
Implicit in this box-and-wire notation is the assumption that composition is associative.



...and unital.



We express the monoidal product of two objects as juxtaposition of wires.



...and the monoidal product of morphisms as the juxtaposition of boxes.

$$\begin{array}{c} \downarrow A \\ \boxed{f} \\ \downarrow B \end{array} \otimes \begin{array}{c} \downarrow A' \\ \boxed{g} \\ \downarrow B' \end{array} = \begin{array}{c} \downarrow A \\ \boxed{f} \\ \downarrow B \end{array} \begin{array}{c} \downarrow A' \\ \boxed{g} \\ \downarrow B' \end{array}$$

The monoidal product is also associative and unital, but possibly only up to isomorphism. We denote the monoidal unit I as the empty graph. Note that the bifunctionality of the tensor product is implicit in this notation.

$$(g \otimes g') \circ (f \otimes f') = (g \circ f) \otimes (g' \circ f')$$

$$\begin{array}{c} \downarrow A \quad \downarrow A' \\ \boxed{f} \quad \boxed{f'} \\ \downarrow B \quad \downarrow B' \\ \boxed{f'} \quad \boxed{g'} \\ \downarrow C \quad \downarrow C' \end{array} = \begin{array}{c} \downarrow A \quad \downarrow A' \\ \boxed{f} \quad \boxed{f'} \\ \downarrow B \quad \downarrow B' \\ \boxed{f'} \quad \boxed{g'} \\ \downarrow C \quad \downarrow C' \end{array}$$

The following proposition is a simple consequence of bifunctionality:

Proposition 2.0.3. *For any morphisms $f : A \rightarrow B$ and $g : A' \rightarrow B'$ in a monoidal category,*

$$(B \otimes g) \circ (f \otimes A') = (f \otimes B') \circ (A \otimes g)$$

Proposition 2.0.3 can be interpreted graphically by “sliding boxes” past each other:

$$\begin{array}{c} \downarrow A \\ \boxed{f} \\ \downarrow B \end{array} \begin{array}{c} \downarrow A' \\ \boxed{g} \\ \downarrow B' \end{array} = \begin{array}{c} \downarrow A \\ \boxed{f} \\ \downarrow B \end{array} \begin{array}{c} \downarrow A' \\ \boxed{g} \\ \downarrow B' \end{array} \tag{2.3}$$

It was proved by Joyal and Street that planar box-and-wire diagrams can unambiguously represent morphisms in a monoidal category. They showed furthermore that this representation is sound and complete with respect to the algebraic definition of a monoidal category. This and similar results will be discussed at length in section 2.3.2.

2.1 Types of Monoidal Categories

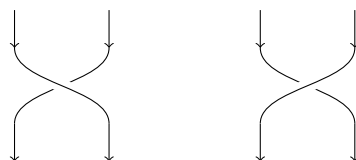
So far, we have introduced monoidal categories. These are sometimes referred to as *planar* monoidal categories, as diagrams of morphisms are always planar. However, many monoidal categories come with a notion of “crossing” wires. The weakest such category is called a braided monoidal category.

Definition 2.1.1. A *braided monoidal category* is a monoidal category $(\mathcal{V}, \otimes, I, \alpha, \lambda, \rho)$ with an additional natural isomorphism $\gamma_{A,B} : A \otimes B \rightarrow B \otimes A$ called a *braiding*, such that $\rho_A := \lambda_A \circ \gamma_{A,I}$ and the follow diagrams commute:

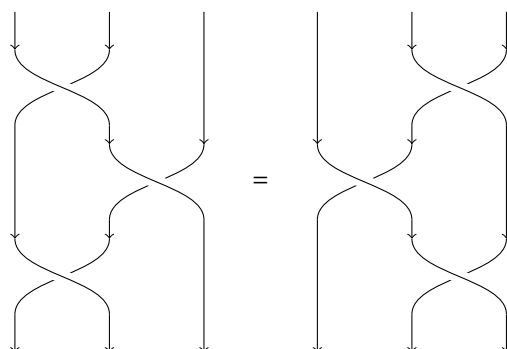
$$\begin{array}{ccccc}
 & & (B \otimes A) \otimes C & \xrightarrow{\alpha} & B \otimes (A \otimes C) \\
 & \nearrow^{\gamma \otimes C} & & & \searrow^{B \otimes \gamma} \\
 (A \otimes B) \otimes C & & & & & B \otimes (C \otimes A) \\
 & \searrow_{\alpha} & & & \nearrow_{\alpha} \\
 & & A \otimes (B \otimes C) & \xrightarrow{\gamma} & (B \otimes C) \otimes A
 \end{array} \tag{2.4}$$

$$\begin{array}{ccccc}
 & & (B \otimes A) \otimes C & \xrightarrow{\alpha} & B \otimes (A \otimes C) \\
 & \nearrow^{\gamma^{-1} \otimes C} & & & \searrow^{B \otimes \gamma^{-1}} \\
 (A \otimes B) \otimes C & & & & & B \otimes (C \otimes A) \\
 & \searrow_{\alpha} & & & \nearrow_{\alpha} \\
 & & A \otimes (B \otimes C) & \xrightarrow{\gamma^{-1}} & (B \otimes C) \otimes A
 \end{array} \tag{2.5}$$

The braiding γ and its inverse γ^{-1} are drawn as wire crossings. Note how one wire is explicitly drawn over the top of the other.



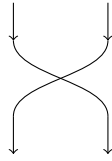
This is to emphasise that γ may not be equal to γ^{-1} . That is, we cannot simply pass wires through each other in the graphical language. However, the naturality of γ and the diagrams from Definition 2.1.1 suffice to prove any equation about morphisms in a braided monoidal category that we can prove geometrically with braid diagrams. In particular, γ satisfies the Yang-Baxter equation:



We shall primarily be interested in a special case of a braided monoidal category called a symmetric monoidal category.

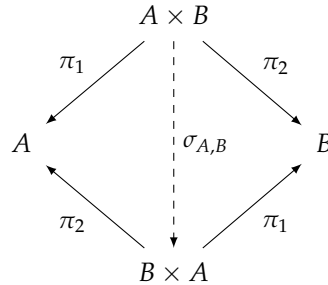
Definition 2.1.2. A *symmetric monoidal category* (SMC) is a monoidal category with a braiding σ such that $\sigma_{A,B} = \sigma_{B,A}^{-1}$.

When a braided monoidal category is symmetric, we refer to the braiding σ as the *symmetry map*. To emphasise the fact that $\sigma_{A,B} = \sigma_{B,A}^{-1}$, we do not distinguish over- and under-crossings:



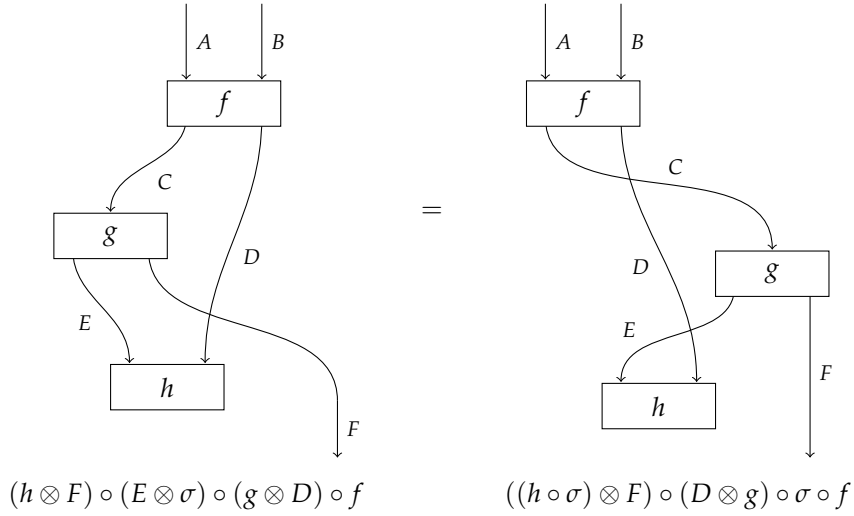
Examples 2.1.3. All of the categories from Examples 2.0.2 are symmetric monoidal categories.

- $(\mathbf{Set}, \times, \{\ast\})$, with $\sigma_{A,B} : A \times B \rightarrow B \times A$ defined as the canonical swap map $(a, b) \mapsto (b, a)$.
- $(\mathbf{Set}, +, \{\ast\})$, with $\sigma_{A,B} : A + B \rightarrow B + A$ the map that interchanges the two components of the disjoint union.
- For any category with finite products, the projection maps induce a canonical symmetry map:



- The swap map is induced similarly in any category with finite coproducts.
- $(\mathbf{Vect}_k, \otimes, k)$ is an SMC, with σ the tensor swap map. I.e. it is the linear extension of $\sigma_{V,W}(v \otimes w) = w \otimes v$.
- $(\mathbf{Rel}, \times, \{\ast\})$ is an SMC, with σ defined as the swap map of the cartesian product.

We can interpret any progressive diagram (i.e. a diagram with no feedback loops) as a morphism in a symmetric monoidal category. Like in the case of planar monoidal categories, the axioms of a symmetric monoidal category ensure that there can be no ambiguity.



The natural next question to ask would be, “Is there a meaningful way to interpret diagrams *with* feedback loops?” The answer to this question is yes. There are actually two meaningful ways to do this. The first is a *traced category*, and the second, which subsumes the first, is a *compact closed category*.

Definition 2.1.4. A symmetric traced category \mathcal{V} is a symmetric monoidal category with a function

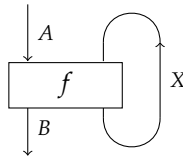
$$\text{Tr}^X : \text{hom}_{\mathcal{V}}(A \otimes X, B \otimes X) \rightarrow \text{hom}_{\mathcal{V}}(A, B)$$

defined for all objects A, B, X , satisfying the following five axioms:

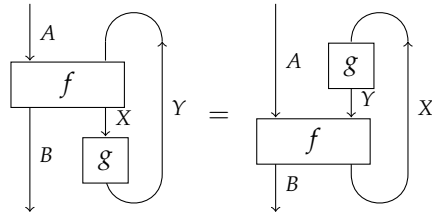
1. $\text{Tr}^X((g \otimes X) \circ f \circ (h \otimes X)) = g \circ \text{Tr}^X(f) \circ h$
2. $\text{Tr}^Y(f \circ (A \otimes g)) = \text{Tr}^X((B \otimes g) \circ f)$
3. $\text{Tr}^I(f) = f$ and $\text{Tr}^{X \otimes Y}(f) = \text{Tr}^X(\text{Tr}^Y(f))$
4. $\text{Tr}^X(g \otimes f) = g \otimes \text{Tr}^X(f)$
5. $\text{Tr}^X(\sigma_{X,X}) = 1_X$

We refer to Tr^X as the *trace operation* and X as the object being *traced out*.

We depict this graphically by connecting the X -output of a map $f : A \otimes X \rightarrow B \otimes X$ to the X -input.

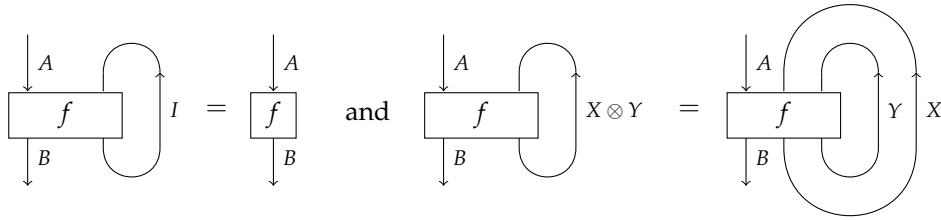


Axioms 1 and 4 are implicit in this notation, since we do not draw the bounds of the trace operation. Axiom 2 is a “box-sliding” identity:

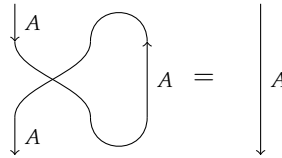


A special case ($A = B = I$) of this axiom is the familiar property of matrix traces in linear algebra: $\text{Tr}(MN) = \text{Tr}(NM)$, i.e. the value of the trace is not affected by cyclic permutations.

Axiom 3 makes the trace operation respect the monoidal product on objects:



Axiom 5 allows us to pull out loops in the diagram:



Examples 2.1.5. There are at least two ways in which the category **Rel** can be made into a symmetric traced category: one for each of the monoidal products defined from Examples 2.0.2.

- **(Rel, \otimes)**: For a relation $R : A \times X \rightarrow B \times X$, we define a new relation $\text{Tr}^X(R) : A \rightarrow B$ as follows:

$$a \left[\text{Tr}^X(R) \right] b \Leftrightarrow \forall x \in X. (a, x)R(b, x)$$

Thinking of a relation as a matrix over the booleans, this is analogous to the usual partial trace of a matrix.

- The trace for **(Rel, \oplus)** was defined in [34]. Let $R : A \oplus X \rightarrow B \oplus X$ be a relation. We define the trace as:

$$a \left[\text{Tr}^X(R) \right] b \Leftrightarrow (aRb) \vee (\exists x, x' \in X. aRx \wedge x'Rx \wedge x'Rb)$$

The new relation incorporates “feedback” from the X -output of R to the X -input of R via term “ $x'Rx$ ” on the RHS.

Example 2.1.6. The category **FVect $_k$** of finite-dimensional vector spaces and linear maps is a traced monoidal category, with Tr given by the partial trace operation on a linear map. Suppose $f :$

$A \otimes X \rightarrow B \otimes X$ is a linear map. Then, by fixing bases $x^i \in X$, $a^i \in A$ and $b^i \in B$, then f is uniquely determined by an indexed collection $f_{k,l}^{i,j} \in K$ called a *tensor*.

$$f(a^i \otimes x^j) = \sum_{k,l} f_{k,l}^{i,j} b^k \otimes x^l$$

We can then define a new tensor by summing together the lower X -index with the upper X -index.

$$\tilde{f} = \sum_k f_{j,k}^{i,k}$$

This new tensor defines a linear map from A to B :

$$\text{Tr}^X(f)(a^i) = \sum_j \tilde{f}_j^i b^j$$

Equivalently, for $x_i \in X^*$ the corresponding basis of the dual space of X , we can define the partial trace as:

$$\text{Tr}^X(f) := \sum_i (B \otimes x_i) \circ f \circ (A \otimes x^i)$$

When $A = B = K$, this is just the usual trace of a matrix. With this in mind, we can see why \mathbf{FVect}_K is an example of a traced monoidal category, but \mathbf{Vect}_K is not. For an infinite-dimensional vector space V , $\text{Tr}^V(1_V)$ is also infinite. In particular, it is not an element of $\text{hom}(K, K) \cong K$.

Note how the dual space plays a role in the definition of the trace. The dual space is actually a special case of a general categorical notion called a *dual*.

Definition 2.1.7. Let A and A^* objects in a monoidal category. A^* is called the *right dual* of A (equivalently, A is called the *left dual* of A^*) if there exist maps $d_A : A \otimes A^* \rightarrow I$, $e_A : I \rightarrow A^* \otimes A$ satisfying the “line-yank” identities:

$$\begin{array}{ccc} A & & A^* \xrightarrow{e_A \otimes A^*} A^* \otimes A \otimes A^* \\ \downarrow A \otimes e & \searrow 1_A & \downarrow A^* \otimes d \\ A \otimes A^* \otimes A & \xrightarrow{d_A \otimes A} & A \end{array}$$

e_A is called the *cap*, and d_A is called the *cup*, of the compact structure.

In the graphical notation, the object A^* is represented as a wire labelled A , but directed upward instead of downward.

$$A := \begin{array}{c} | \\ \downarrow \\ A \end{array} \qquad A^* := \begin{array}{c} \uparrow \\ | \\ A \end{array}$$

We represent e_A and d_A as half-turn of wire, forming a cup or a cap.

$$d_A := \text{cup} \quad e_A := \text{cap}$$

The diagrams from Definition 2.1.7 are called the “line-yank” identities, because their graphical representations literally look like pulling a wire straight.

Definition 2.1.8. A monoidal category where every object has a right (resp. left) dual is called a *right (resp. left) autonomous category*.

Definition 2.1.9. A *compact closed category* is a category that is right autonomous and symmetric. For an object A in a compact closed category, the dual maps d_A and e_A are called a *compact structure* for A .

Note that compact closed categories are automatically left autonomous. Any right dual A^* of A can be made into a left dual by choosing maps $d' := d \circ \sigma$ and $e' := \sigma \circ e$. The left line-yank identity can then be derived from symmetry and the right line-yank.

Also note that in a compact category, and map $f : A \rightarrow B$ can also be considered as a map $f^* : A^* \rightarrow B^*$ by using caps and cups to “bend the wires” around.

Example 2.1.10. The category \mathbf{FVect}_K is compact closed. For any finite-dimensional vector space A , fix bases $a^i \in A$ and $a_i \in A^*$ such that $a_j \circ a^i = \delta_i^j$. Then, define a compact structure for A as follows:

$$d_A :: a_i \otimes a^j \mapsto (a_i \circ a^j) \quad e_A :: 1 \mapsto \sum_i a_i \otimes a^i$$

Compact closed categories are automatically symmetric traced categories. A cap and a cup can be used to construct a “feedback loop” that acts as a trace operation.

$$\text{Tr}^X(f) := (B \otimes d_X) \circ (f \otimes X^*) \circ (A \otimes e_X') \tag{2.6}$$

The axioms of a compact structure then suffice to prove the five trace axioms given in Definition 2.1.4. Therefore, compact closed categories subsume symmetric traced categories. Often these categories are easier to work with than their traced counterparts, especially using the graphical language. It would be convenient to use the compact structure axioms when proving identities in symmetric traced categories. This is possible due to a result by Joyal, Street, and Verity.

Theorem 2.1.11 ([34]). *Any symmetric traced category can be fully and faithfully embedded in a compact category.*

They prove this result by defining the free compact closure of a symmetric traced category, using a technique called the “Int construction”. For a symmetric traced category \mathcal{V} , they build a compact closed category $\text{Int}(\mathcal{V})$ into which \mathcal{V} embeds fully and faithfully. Thus $f = g$ in \mathcal{V} if and only if $f = g$ in $\text{Int}(\mathcal{V})$. This construction is also free over \mathcal{V} , i.e. for a compact closed category \mathcal{V}' , a traced symmetric functor $F : \mathcal{V} \rightarrow \mathcal{V}'$ extends uniquely to a compact closed functor $\tilde{F} : \text{Int}(\mathcal{V}) \rightarrow \mathcal{V}'$.

We shall introduce one more type of monoidal category, introduced by Abramsky and Coecke, for the sake of reasoning about quantum information [3].

Definition 2.1.12. A category \mathcal{C} is called a \dagger -category if there exists an identity-on-objects functor $(-)^{\dagger} : \mathcal{C} \rightarrow \mathcal{C}^{\text{op}}$ such that $((-)^{\dagger})^{\dagger} = 1_{\mathcal{C}}$.

In particular, \dagger -categories are always isomorphic with their opposite category. As the notation might suggest, the \dagger functor is an abstract version of the conjugate-transpose of a complex linear map. We can use it to define an abstract notion of unitarity.

Definition 2.1.13. A morphism U in a \dagger -category is called *unitary* if it is an isomorphism and $U^{\dagger} = U^{-1}$.

\dagger -monoidal categories are simply monoidal \dagger -categories, where all of the structural natural isomorphisms are actually unitary isomorphisms.

Definition 2.1.14. A \dagger -symmetric monoidal category is a symmetric monoidal category where all of the components of α , λ , and σ are unitary.

\dagger -compact closed categories have the additional property that the \dagger reflects caps and cups vertically.

Definition 2.1.15. A \dagger -compact closed category is a compact closed category where all of the components of α , λ , and σ are unitary and for all A , $d_A^{\dagger} = e'_A$ and $e_A^{\dagger} = d'_A$.

Examples 2.1.16. Several examples of \dagger -compact closed categories are:

- (\mathbf{Rel}, \times) , with \dagger given by relational converse: $aR^{\dagger}b \Leftrightarrow bRa$

- $(\mathbf{Mat}(K), \otimes)$, with \dagger given by transposition of matrices
- $(\mathbf{FHilb}, \otimes)$, the category of finite-dimensional complex Hilbert spaces. The compact structure is the same as for $\mathbf{FVect}_{\mathbb{C}}$, and \dagger is given by the adjoint of linear operators with respect to $\langle - | - \rangle$.

The category \mathbf{Hilb} is a \dagger -symmetric monoidal category, but it is not traced (and hence not compact closed).

For each of the categories we have introduced, we can form the category of (symmetric, traced, compact, ...) monoidal categories and suitably structure-preserving functors.

Definition 2.1.17. For monoidal categories \mathcal{C}, \mathcal{D} , a strong monoidal functor consists of functor $F : \mathcal{C} \rightarrow \mathcal{D}$ an isomorphism $\phi : I \rightarrow F(I)$ and a natural isomorphism $\psi_{A,B} : FA \otimes FB \rightarrow F(A \otimes B)$ such that following diagrams commute:

$$\begin{array}{ccc}
(FA \otimes FB) \otimes FC & \xrightarrow{\psi \otimes FC} & F(A \otimes B) \otimes FC \xrightarrow{\psi} F((A \otimes B) \otimes C) \\
\downarrow \alpha & & \downarrow F(\alpha) \\
FA \otimes (FB \otimes FC) & \xrightarrow{FA \otimes \psi} & FA \otimes F(B \otimes C) \xrightarrow{\psi} F(A \otimes (B \otimes C)) \\
\\
FA \otimes I & \xrightarrow{\rho} & FA & I \otimes FA & \xrightarrow{\lambda} & FA \\
\downarrow FA \otimes \phi & & \downarrow F(\rho^{-1}) & \downarrow \phi \otimes FA & & \downarrow F(\lambda^{-1}) \\
FA \otimes FI & \xrightarrow{\psi} & F(A \otimes I) & FI \otimes FA & \xrightarrow{\psi} & F(I \otimes A)
\end{array}$$

The adjective *strong* is used to distinguish from a *lax monoidal functor*, where the isomorphisms are replaced by arbitrary maps. In the case of strict monoidal categories, this definition simplifies, as coherence diagrams need not be considered.

Definition 2.1.18. For strict monoidal categories \mathcal{C}, \mathcal{D} , a *strict monoidal functor* is a functor $F : \mathcal{C} \rightarrow \mathcal{D}$ where $I = F(I)$ and $F(A \otimes B) = FA \otimes FB$.

For symmetric, traced, compact closed, and \dagger -compact closed categories, we can make similar definitions, where the functor application must commute with all of the structure in sight. There are also strict versions of all of these categories, where the associativity and unit natural isomorphisms are all identities.

Definitions 2.1.19. The following are all categories of (small) monoidal categories:

- **MonCat**: the category of monoidal categories and monoidal functors
- **SymMonCat**: the category of symmetric monoidal categories and symmetric monoidal functors

- **SymTraceCat**: the category of symmetric traced categories and symmetric traced functors
- **CCCat**: the category of compact closed categories and compact functors
- **MonCat_s, SymMonCat_s, SymTraceCat_s, CCCat_s**: the strict versions

2.2 Free Monoidal Categories

An important question for monoidal categories is as follows.

Given a suitable description for the generators of a (symmetric, traced, compact closed, ...) monoidal category, can we generate the *free* such category?

This question is important because two arrows f and g are equal in a free monoidal category *if and only if* their equality can be established only using the axioms of that category. We can make this precise by expounding on the usual universal property satisfied by free objects, but first, we define the notion of a *monoidal signature*, which defines the generators of a monoidal category.

Notation 2.2.1. For a set O , let $w(O)$ be the free monoid over O , i.e. the set of lists with elements taken from O where multiplication is concatenation and the unit is the empty list. For a function $f : O \rightarrow O'$, let $w(f) : w(O) \rightarrow w(O')$ be the lifting of f to lists:

$$w(f)([A, B, C, \dots]) = [f(A), f(B), f(C), \dots]$$

Definition 2.2.2. A (small, strict) monoidal signature $T = (O, M, \text{dom}, \text{cod})$ consists of a set of objects O , a set of morphisms M , and a pair of functions $\text{dom} : M \rightarrow w(O)$ and $\text{cod} : M \rightarrow w(O)$.

The maps dom and cod should be interpreted as giving input and output types to a morphism $m \in M$. For instance, if $\text{dom}(m) = [A, B, C]$ and $\text{cod}(m) = [D]$, then m represents a morphism from $A \otimes B \otimes C$ to D . The empty list is interpreted as the tensor unit I .

Example 2.2.3. Define a monoidal signature $T = (O, M, \text{dom}, \text{cod})$ where

$$\begin{aligned} O &= \{A, B, C\} \\ M &= \{f, g\} \\ \text{dom} &= \{f \mapsto [A, B], g \mapsto [C]\} \\ \text{cod} &= \{f \mapsto [C], g \mapsto [C]\} \end{aligned}$$

This signature defines three (primitive) objects A, B, C and two morphisms $f : A \otimes B \rightarrow C$ and $g : C \rightarrow C$. We will often write a signature as a set of boxes, representing the diagram generators:

$$T := \left\{ \begin{array}{c} A \quad B \\ \downarrow \quad \downarrow \\ \boxed{f} \\ \downarrow \\ C \end{array} , \begin{array}{c} C \\ \downarrow \\ \boxed{g} \\ \downarrow \\ C \end{array} \right\}$$

There is also a notion of a non-strict monoidal signature. In that case, $w(O)$ is replaced with the free (\otimes, I) -algebra over O . We treat the strict case for simplicity, but many of the results translate immediately, replacing equality with coherent natural isomorphism.

Definition 2.2.4. For monoidal signatures S, T , a monoidal signature homomorphism f consists of functions $f_O : O_S \rightarrow O_T$ and $f_M : M_S \rightarrow M_T$ such that the following diagrams commute.

$$\begin{array}{ccc}
 M_S & \xrightarrow{\text{dom}_S} & w(O_S) \\
 f_M \downarrow & & \downarrow w(f_O) \\
 M_T & \xrightarrow{\text{dom}_T} & w(O_T)
 \end{array}
 \qquad
 \begin{array}{ccc}
 M_S & \xrightarrow{\text{cod}_S} & w(O_S) \\
 f_M \downarrow & & \downarrow w(f_O) \\
 M_T & \xrightarrow{\text{cod}_T} & w(O_T)
 \end{array}$$

MonSig is the category of monoidal signatures and monoidal signature homomorphisms.

A monoidal signature is essentially a strict monoidal category without composition or identity maps. A monoidal signature homomorphism is thus a monoidal functor, minus the condition that it respect composition and identity maps.

Definition 2.2.5. A monoidal signature is called *simple* if the images of dom and cod are restricted to single-element lists.

There are evident forgetful functors from **MonCat_s**, **SymMonCat_s**, **SymTraceCat_s**, and **CCCat_s** into **MonSig**. If this forgetful functor has a left adjoint F , the image of a signature T under F is called the *free monoidal category* over T .

To get a better feel for these objects, we unroll the universal property of the free category. Fix a monoidal category \mathcal{V} . Then, for a monoidal signature T , a monoidal signature homomorphism from T to $U(\mathcal{V})$ is called a *valuation*. This homomorphism gives a value in \mathcal{V} for each of the generators in T . The universal property of the adjunction then guarantees there is a unique strong monoidal functor $\bar{v} : FT \rightarrow \mathcal{V}$ such that:

$$\begin{array}{ccc}
 T & \hookrightarrow & UF(T) \\
 & \searrow v & \downarrow U(\bar{v}) \\
 & & U\mathcal{V}
 \end{array}$$

In the non-strict case, \bar{v} is only unique up to unique, coherent natural isomorphism. In [37], Kelly and Laplaza gave a prescription for constructing the free category on any “algebraically-defined” additional structure on a category. They went on to describe concretely the free compact closed category on a category (or equivalently, a simple signature). Shum proved a similar result in 1994 [63] for tortile categories, i.e. braided monoidal categories with coherently-defined left and right duals. In the next section, we will discuss work to define the free symmetric, traced, and compact categories on an *arbitrary* signature, using graphical language.

2.3 Formalising Graphical Languages

There are three ways in which one can formalise graphical languages for monoidal categories. The first formalisation is **algebraic**, where string diagrams are used as an equivalent representation of a tensor network defined using the *abstract index* notation introduced by Penrose in his 1971 paper [58]. The second formalisation is **topological**. In this approach, topological graphs (i.e. realisations of 1D simplicial complexes) with added structure are used to represent morphisms [35]. The third formalisation is **combinatoric**. A special kind of typed graph called a *string graph* is used to represent morphisms. This approach was developed by Dixon, Duncan, and Kissinger [22, 24]. In this section, we'll discuss the first two approaches. We'll discuss *string graphs* in detail in Part I.

2.3.1 Algebraic Approach: Abstract Tensor Systems

In this section, we will look at the original formulation of string diagrams, due to Penrose in 1971 [58].

Recall that a tensor is a set of real or complex numbers, indexed by one or more natural numbers. For example, the following is an $(n_1 \cdot n_2 \cdot n_3)$ -dimensional tensor:

$$\{\chi_{ij}^k : i = 1..n_1; j = 1..n_2; k = 1..n_3\}$$

Subscripts should be thought of as inputs and superscripts as outputs. Familiar examples of tensors are vectors, v^i and matrices, M_i^j . We can compose tensors by *contraction*, i.e. "summing together" a lower index and an upper index of the same dimension:

$$\zeta_j^i = \sum_{kl} \chi_{kl}^i \beta_j^k \rho^l$$

As expected, when we focus on vectors and matrices, we recover the usual notions of composition and application of linear maps. In order to simplify such expressions, we can use the Einstein summation convention, where any repeated indices are assumed to be summed over.

$$\zeta_j^i = \chi_{kl}^i \beta_j^k \rho^l$$

Penrose introduced *abstract tensor systems* to express generalised tensors and contractions. In many ways, this formalism resembles that of monoidal categories. Natural number indices are replaced with formal *labels*, which are simply names that can be used to identify inputs and outputs. These are taken from a *labelling set*.

$$\mathcal{L} = \{a, b, c, \dots, a_0, b_0, \dots, a_1, b_1, \dots\}$$

Vector spaces are replaced by sets of formal tensors. For two lists of labels U, L the set \mathcal{T}_L^U has as elements *formal tensors*. For $L = \{a_0, \dots, a_m\}$ and $U = \{b_0, \dots, b_n\}$, we write a formal tensor

using the usual tensor notation:¹

$$\chi_{a_0, \dots, a_m}^{b_0, \dots, b_n} \in \mathcal{T}_L^U$$

It's useful to think of the sets \mathcal{T}_L^U as something akin to a hom-set, whose elements ζ are morphisms from $X^{\otimes |L|}$ (i.e. $|L|$ copies of X) to $X^{\otimes |U|}$. Penrose defines four operations over sets of abstract tensors.

Relabelling: $R : \mathcal{T}_L^U \rightarrow \mathcal{T}_{L'}^{U'}$ for $L \cong L'$ and $U \cong U'$

Addition: $+$: $\mathcal{T}_L^U \times \mathcal{T}_L^U \rightarrow \mathcal{T}_L^U$

Outer product: \otimes : $\mathcal{T}_L^U \times \mathcal{T}_{L'}^{U'} \rightarrow \mathcal{T}_{L+L'}^{U+U'}$

Contraction: $C_p^q : \mathcal{T}_L^U \rightarrow \mathcal{T}_{L-\{p\}}^{U-\{q\}}$

These satisfy certain compatibility axioms (e.g. associativity and identity laws), mirroring those of normal tensor contraction.

Remark 2.3.1. The usual, categorical composition can be defined in terms of outer product and contraction.

$$\beta_{y'_0, \dots, y'_m}^{z_0, \dots, z_n} \circ \alpha_{x_0, \dots, x_l}^{y_0, \dots, y_m} := C_{y'_0}^{y_0} C_{y'_1}^{y_1} \dots C_{y'_m}^{y_m} (\alpha_{x_0, \dots, x_l}^{y_0, \dots, y_m} \otimes \beta_{y'_0, \dots, y'_m}^{z_0, \dots, z_n})$$

If we then include for every pair of labels a Dirac delta tensor δ_a^b (i.e. an identity map), with suitable axioms, the data from an abstract tensor system defines a symmetric traced category. In chapter 5.5 we make use of (essentially) the converse construction to prove the main theorem.

With the inclusion of raising and lowering tensors $g_{a,b}$, $g^{a,b}$, it becomes a compact closed category with $X = X^*$. For this reason, abstract tensor systems are widely regarded as the prototype for compact closed categories, in their modern formulation.

As in the concrete case, we represent outer (i.e. tensor) product as juxtaposition, contraction by repeating an index, and let relabeling be implicit.

Example 2.3.2. The following is an abstract tensor contraction, followed by its explicit form, in terms of the functions above.

$$\alpha_{ab}^c \beta_c^d + \gamma_{ab}^d := C_c^c (\alpha_{a,b}^c \otimes \beta_{c'}^d) + \gamma_{a,b}^d$$

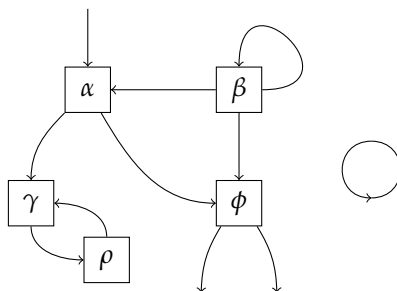
However, even with this convention, contraction expressions can get quite complex. Consider this expression, involving six abstract tensors:

$$\alpha_{abc}^{de} \beta_f^{bf} \gamma_{dh}^i \rho_i^h \phi_{eg}^{jk} \delta_l^l \quad (2.7)$$

In order to work with this expression, one has to keep track of 11 indices, which makes computations time-consuming and error-prone. To address this issue, Penrose introduced a second,

¹Note that the data associated with the tensor includes a total ordering on the sets L and U . This is implicit in the use of *lists* of index names in the tensor.

graphical notation. Tensors are drawn as boxes, and contractions over pairs of indices as wires. The “identity” tensor (i.e. the Dirac delta δ_i^j) is also drawn as a wire. The non-contracted, or “free” indices are left as dangling wires, and contractions δ_i^i are represented as circles. In the graphical notion, expression (2.7) becomes the following diagram:



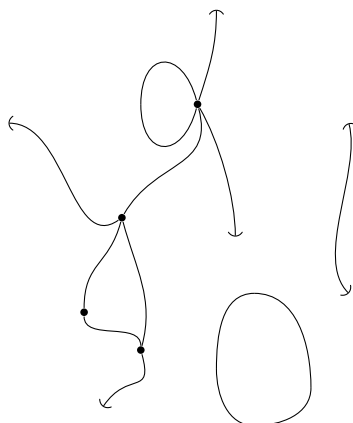
Such a diagram can always be interpreted, unambiguously as an abstract tensor network, up to a relabelling of indices. It is also clear how the data above forms a symmetric traced category.

2.3.2 Topological Approach: Anchored Graphs

In 1991, Joyal and Street formalised the graphical Penrose notation as a generalised topological graph, with some added structure [35]. They went on to prove that variations of these graphs could be used to construct the free planar, symmetric, and braided categories on a monoidal signature. The usual notion of a finite topological graph is a Hausdorff space that forms the geometric realisation of a one-dimensional, finite simplicial (or equivalently, CW) complex. A generalised finite topological graph is a finite topological graph that is allowed to have some “open ends”. That is, some edges in the graph look like the half-open interval or the open interval. We’ll drop the adjective finite for the rest of this section, and assume that all of the graphs are finite.

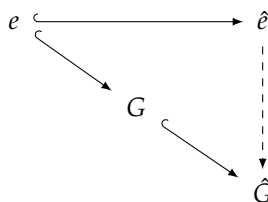
Definition 2.3.3 (Generalised Topological Graph). A *generalised topological graph* is a pair (G, G_0) , where G is a Hausdorff space and G_0 is a finite subset of points in G such that $G - G_0$ is isomorphic to a sum of open intervals $I_o := (0, 1) \subseteq \mathbb{R}$ and copies of S_1 . The points in G_0 are called *vertices*. The compactification of an open interval $I_o \cong e \subseteq G - G_0$ is called an *edge* \hat{e} . A copy of $S_1 \subseteq G - G_0$ is called a *circle* \hat{c} . If a subgraph of \hat{G} is an edge or a circle, we shall call it a *wire* in G . Let $W(G)$ be the set of all wires in G .

Example 2.3.4. A generalised topological graph. The points in G_0 are marked as dots.



This gives a bit more information than just the graph itself. The set G_0 is used to identify all of the “logical” vertices in the graph, not just the “topological” vertices (i.e. those points lacking a neighbourhood of $[0, 1]$). In particular, a vertex can occur along an edge. Also note that G need not be compact. Let \hat{G} be the compactification of G obtained by freely adding endpoints to any open ends. Since $(\hat{-})$ is left adjoint to the forgetful functor $U : \mathbf{CHaus} \rightarrow \mathbf{Haus}$, the embedding of e in G uniquely fixes an embedding from \hat{e} into \hat{G} .

Note that all edges naturally embed in the compactification $\hat{G} \supseteq G$ obtained by adding endpoints to open edges.



String diagrams are not just graphs, but directed graphs. Thus, more data needs to be added to impose directedness to the wires in the graph. These wires can either be edges ($\cong [0, 1]$) or circles ($\cong S^1$). In both cases, we have orientable manifolds, so we impose directedness by giving each of these manifolds an orientation.

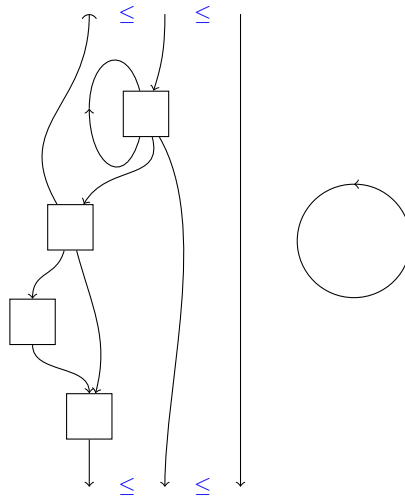
Definition 2.3.5 (Polarised Graph). A *polarised graph* is a tuple $(G, G_0, (o_w), (p_{v,i}))$, where o_w assigns each wire $w \in W(G)$ an orientation. We can therefore define an input $\hat{e}(0)$ and an output $\hat{e}(1)$ for each edge. For each vertex $v \in G_0$, $\text{in}(v)$ is the set of edges such that $\hat{e}(1) = v$ and $\text{out}(v)$ is the set of edges such that $\hat{e}(0) = v$. For all $v \in G_0$, $p_{v,0}$ is a total order on $\text{in}(v)$ and $p_{v,1}$ is a total order on $\text{out}(v)$, called a *polarisation*. A polarised graph that contains no directed cycles is called *progressive*.

Polarised graphs come with a notion of boundary. We can furthermore put an ordering on this boundary.

Definition 2.3.6 (Boundary of a polarised graph). For a polarised graph $(G, G_0, (o_w), (p_{v,i}))$, $\partial G := \hat{G} - G$ is a discrete space called the *boundary* of G . Points in ∂G that are the input of some edge are

called *inputs* of G , and outputs of edges in ∂G are called *outputs* of G . A polarised graph with a pair of total orders β_0, β_1 on its inputs and its outputs respectively is called an *anchored graph*.

Example 2.3.7. We can make the topological graph from Example 2.3.4 into an anchored graph $\Gamma = (G, G_0, (o_w), (p_{v,i}), (\beta_i))$. We depict Γ diagrammatically as follows. G is drawn as before, but with the elements of G_0 shown as squares. The orientations provided by ω are depicted as arrow heads on the wires. The polarisation $p_{v,i}$ is shown by ordering from left to right the inputs and outputs to each square. The orderings β_i are shown by ordering the inputs and outputs of the graph from left to right.



This is very close to representing the diagrammatic language given at the beginning of this chapter. The only thing missing is labels on the boxes and wires.

Definition 2.3.8 (Valuation). For an anchored graph $\Gamma = (G, G_0, (o_w), (p_{v,i}), (\beta_i))$ and a monoidal signature $T = (O, M, \text{dom}, \text{cod})$, a *valuation* ν of Γ is a function ν_0 that assigns an element of O to every edge or circle in Γ and a function ν_1 that assigns an element m of M to every point in G_0 in such a way that respects the domain on codomain of m .

An isomorphism of anchored graphs with valuations $(\Gamma, \nu) \rightarrow (\Gamma', \nu')$ is an isomorphism of generalised topological graphs that respects orientations, polarisation, and input and output ordering, and is compatible with the valuations ν and ν' .

Since an anchored graph gives a total order to inputs and outputs, we can associate input and output words to a pair (Γ, ν) . Let $T = (O, M, \text{dom}, \text{cod})$ be a monoidal signature. $\mathbb{F}_S(T)$ is the category whose objects are words in $w(O)$. For words v and w , arrows are isomorphism classes of progressive anchored graphs with valuations into T that have input word v and output word w . Composition $g \circ f$ is defined by plugging the outputs of some representative of the isomorphism class of f into the inputs of some representative of g , then taking the isomorphism class of the resultant graph.

Theorem 2.3.9. [35] $\mathbb{F}_S(T)$ is the free symmetric monoidal category over T .

In [35], Joyal and Street alluded to a sequel paper “*The Geometry of Tensor Calculus, II*” (GTC-II) in which they would prove analogous results for traced and compact closed categories. For various reasons, this paper was never completed. In chapter 5.5 we show that categories of string graphs can be used to construct the free symmetric traced and compact closed categories over a monoidal signature. This, along with the fact that the geometric realisation functor lifts to an equivalence of categories between the string graph-based free categories and the topologically-defined free categories [24] suffices to prove two of the missing GTC-II theorems: namely that suitably defined categories of topological graphs can be used to define (i) the free symmetric traced category and (ii) the free compact closed category over a monoidal signature.

Chapter 3

Algebraic Structures in Monoidal Categories

In a monoidal category \mathcal{V} , one can define various algebraic structures *internal* to \mathcal{V} . A standard example is that of a monoid in \mathcal{V} .

Definition 3.0.10. For a monoidal category \mathcal{V} , a monoid in \mathcal{V} is a triple $(X, \mu : X \otimes X \rightarrow X, \eta : I \rightarrow X)$ such that the following diagrams commute:

$$\begin{array}{ccc}
 X \otimes X \otimes X & \xrightarrow{X \otimes \mu} & X \otimes X \\
 \mu \otimes X \downarrow & & \downarrow \mu \\
 X \otimes X & \xrightarrow{\mu} & X
 \end{array}
 \quad
 \begin{array}{ccc}
 X & & \\
 \eta \otimes X \downarrow & \searrow 1_X & \\
 X \otimes X & \xrightarrow{\mu} & X
 \end{array}
 \quad
 \begin{array}{ccc}
 X & \xrightarrow{X \otimes \eta} & X \otimes X \\
 \downarrow & \searrow 1_X & \downarrow \mu \\
 & & X
 \end{array}
 \quad (3.1)$$

The diagrams in this definition establish associativity, left unit, and right unit.

Example 3.0.11. A monoid in the category $(\mathbf{Set}, \times, \{*\})$ is the monoid in the usual sense. That is, a group without inverses. X is the carrier set, $\mu : X \times X \rightarrow X$ is the associative multiplication operation, and $\nu : \{*\} \rightarrow X$ is the map that picks out the unit, i.e. $\eta(*) = e$. If we write $\mu(x, y)$ as $x \cdot y$, then the commutative diagrams from Definition 3.0.10 are equivalent to these equations:

- $(x \cdot y) \cdot z = x \cdot (y \cdot z)$,
- $e \cdot x = x$, and $x \cdot e = x$.

Examples 3.0.12. A monoid in $(\mathbf{Vect}_K, \otimes, K)$ is an associative, unital K -algebra. Since $\mu : V \otimes V \rightarrow V$ is a linear map, then due to the universal property of the tensor product, μ uniquely determines a *bilinear* map $(- \cdot -) : V \times V \rightarrow V$. Again, associativity and unitality come from Diagrams (3.1). Similarly, a monoid in $(\mathbf{Ab}, \otimes, \mathbb{N})$, the category of Abelian groups and group homomorphisms, is a ring.

We can also express a monoid graphically, as a triple $(X, \bullet : X \otimes X \rightarrow X, \circ : I \rightarrow X)$. Associativity, left unit, and right unit become the following equations.

$$\begin{array}{c} \diagup \\ \bullet \\ \diagdown \end{array} = \begin{array}{c} \diagdown \\ \bullet \\ \diagup \end{array} \quad \begin{array}{c} \bullet \\ \diagup \\ \bullet \\ \diagdown \end{array} = \begin{array}{c} \bullet \\ \diagdown \\ \bullet \\ \diagup \end{array} \quad \begin{array}{c} \bullet \\ \diagup \\ \bullet \\ \diagdown \end{array} = \begin{array}{c} \bullet \\ \diagdown \\ \bullet \\ \diagup \end{array} \quad (3.2)$$

By simply turning everything upside-down, we define a comonoid in a monoidal category.

Definition 3.0.13. For a monoidal category \mathcal{V} , a comonoid in \mathcal{V} is a triple $(X, \delta : X \rightarrow X \otimes X, \epsilon : X \rightarrow I)$ such that the following diagrams commute:

$$\begin{array}{ccc}
 \begin{array}{ccc} X & \xrightarrow{\delta} & X \otimes X \\ \delta \downarrow & & \downarrow \delta \otimes X \\ X \otimes X & \xrightarrow{X \otimes \delta} & X \otimes X \otimes X \end{array} & \begin{array}{ccc} X & & \\ \delta \downarrow & \searrow 1_X & \\ X \otimes X & \xrightarrow{\epsilon \otimes X} & X \end{array} & \begin{array}{ccc} X & \xrightarrow{\delta} & X \otimes X \\ & \searrow 1_X & \downarrow X \otimes \epsilon \\ & & X \end{array} \quad (3.3)
 \end{array}$$

Graphically, these axioms are:

$$\begin{array}{c} \downarrow \\ \bullet \\ \diagdown \end{array} = \begin{array}{c} \downarrow \\ \bullet \\ \diagup \end{array} \quad \begin{array}{c} \bullet \\ \diagup \\ \bullet \\ \diagdown \end{array} = \begin{array}{c} \bullet \\ \diagdown \\ \bullet \\ \diagup \end{array} \quad \begin{array}{c} \bullet \\ \diagup \\ \bullet \\ \diagdown \end{array} = \begin{array}{c} \bullet \\ \diagdown \\ \bullet \\ \diagup \end{array}$$

A *commutative monoid* is a monoid satisfying the following equation:

$$\begin{array}{c} \diagup \\ \bullet \\ \diagdown \end{array} = \begin{array}{c} \diagdown \\ \bullet \\ \diagup \end{array}$$

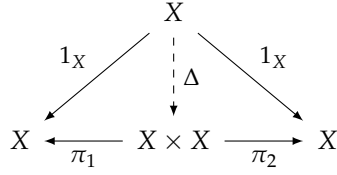
Similarly, a *cocommutative comonoid* is a comonoid satisfying the same equation, but upside-down.

$$\begin{array}{c} \downarrow \\ \bullet \\ \diagup \end{array} = \begin{array}{c} \downarrow \\ \bullet \\ \diagdown \end{array}$$

3.1 Bi-algebras and Hopf Algebras

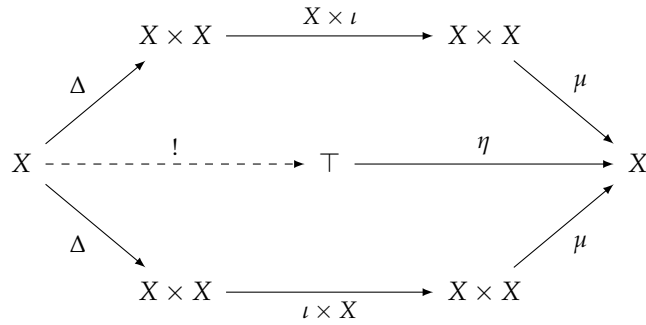
Suppose we wished to define a *group* in a monoidal category. We begin by looking at how groups are defined in **Set**. A group is a monoid, with an additional unary operation $(-)^{-1}$ such that $x \cdot x^{-1} = e = x^{-1} \cdot x$. How can we express these equations as commutative diagrams? This poses a challenge, because unlike in the monoid axioms, the free variable x in the inverse axiom features *twice*. Luckily, **Set** is not just a monoidal category, but a cartesian monoidal category. A cartesian monoidal category is a category with finite products, where the monoidal product is defined using the categorical product and the monoidal unit is the terminal object. In the case of **Set**, these are the cartesian product on the one-element set.

In particular, cartesian monoidal categories always have a diagonal map for every object induced by the universal property of the product.



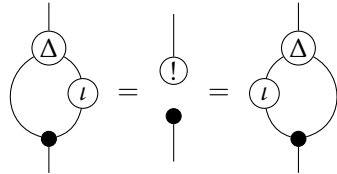
The diagonal map can be used to “copy” the free variable $x \in X$.

Definition 3.1.1. A *group object* in a cartesian monoidal category $(\mathcal{C}, \times, \top)$, is a tuple (X, μ, η, ι) such that (X, μ, η) is a monoid and the following diagram commutes:



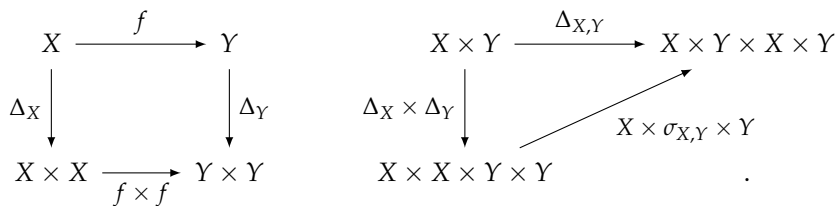
where $\Delta : X \rightarrow X \times X$ is the diagonal map and $! : X \rightarrow \top$ is the unique map from X to \top , the terminal object.

The picture becomes clearer by looking at the graphical versions of these identities. Note how the terminal map is used to “delete” the free variable, just as the Δ map is used to copy it.



We have defined a group object in a cartesian monoidal category, but we have not quite answered the original question. Is there a notion of a group in a (non-cartesian) monoidal category? The maps Δ and $!$ in a monoidal category do not come for free. Instead, we require that they be part of the *structure* of the the “group” object. To justify what this structure should be, we highlight some properties of $(\Delta, !)$.

Proposition 3.1.2. For any object X in a cartesian monoidal category, the triple $(X, \Delta, !)$ forms a comonoid. Furthermore, the following diagrams commute for all X, Y , and f .



Proof. We write $\langle g, h \rangle : A \rightarrow B \times C$ for the unique induced map to the product of the pair of maps $g : A \rightarrow B, h : A \rightarrow C$. We can prove associativity using the definition of Δ and some well-known properties of $\langle -, - \rangle$.

$$\begin{aligned} (X \times \Delta) \circ \Delta &= \langle \pi_1, \Delta \circ \pi_2 \rangle \circ \Delta = \langle \pi_1 \circ \Delta, \Delta \circ \pi_2 \circ \Delta \rangle = \langle 1_X, \Delta \circ 1_X \rangle \\ &= \langle 1_X, \langle 1_X, 1_X \rangle \rangle = \langle \langle 1_X, 1_X \rangle, 1_X \rangle = (\Delta \times X) \circ \Delta \end{aligned}$$

Right unit (and similarly, left unit) follows from terminality of $!$.

$$\begin{aligned} (X \times !_X) \circ \Delta &= \langle \pi_1, !_X \circ \pi_2 \rangle \circ \Delta = \langle \pi_1, !_X \times X \rangle \circ \Delta \\ &= \langle \pi_1 \circ \Delta, !_X \times X \circ \Delta \rangle = \langle 1_X, !_X \rangle = 1_X \end{aligned}$$

The copying property is proved as follows:

$$\begin{aligned} \Delta \circ f &= \langle 1_X, 1_X \rangle \circ f = \langle f, f \rangle = \langle f \circ \pi_1 \circ \Delta, f \circ \pi_2 \circ \Delta \rangle \\ &= \langle f \circ \pi_1, f \circ \pi_2 \rangle \circ \Delta = (f \times f) \circ \Delta \end{aligned}$$

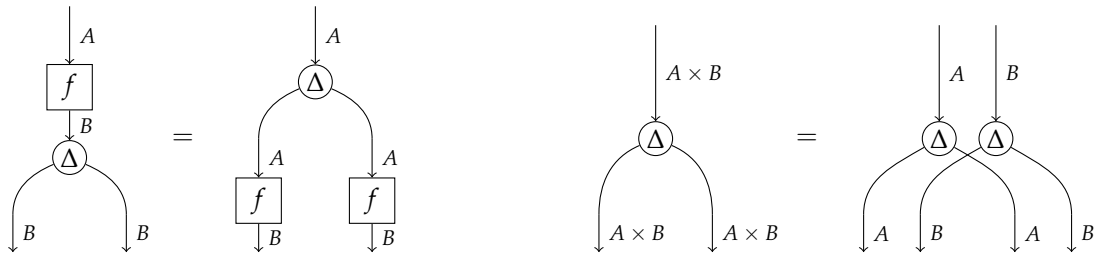
For the final property, we establish that $(\Delta_X \times \Delta_Y) \circ (X \times \sigma_{X,Y} \times Y) = \langle 1_{X,Y}, 1_{X,Y} \rangle$. For simplicity, let π_1, π_2 be projections of the two-part products $(\cdot \times \cdot) \times (\cdot \times \cdot)$, and let $\pi'_1, \pi'_2, \pi'_3, \pi'_4$ be projections of the four-part products $(\cdot \times \cdot \times \cdot \times \cdot)$.

$$\begin{array}{ccccc} & & X \times Y & \xrightarrow{1_{X,Y}} & X \times Y \\ & \nearrow^{1_{X,Y}} & \uparrow & & \uparrow \\ X \times Y & \xrightarrow{\Delta_X \times \Delta_Y} & X \times X \times Y \times Y & \xrightarrow{X \otimes \sigma \otimes Y} & X \times Y \times X \times Y \\ & \searrow_{1_{X,Y}} & \downarrow & & \downarrow \\ & & X \times Y & \xrightarrow{1_{X,Y}} & X \times Y \end{array}$$

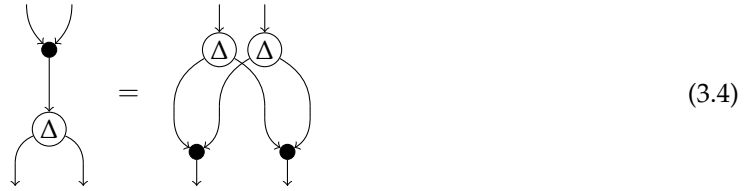
$\begin{array}{l} \pi_1 \times \pi_1 = \langle \pi'_1, \pi'_3 \rangle \\ \pi_2 \times \pi_2 = \langle \pi'_2, \pi'_4 \rangle \end{array}$
 $\begin{array}{l} \langle \pi'_1, \pi'_2 \rangle = \pi_1 \\ \langle \pi'_3, \pi'_4 \rangle = \pi_2 \end{array}$

□

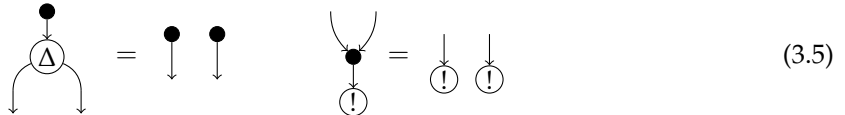
We already saw graphical versions of the comonoid axioms after 3.0.13. The remaining two diagrams from Proposition 3.1.2 are:



In particular, Proposition 3.1.2 implies that Δ copies μ .



Δ also copies the unit η . By terminality, μ also “co-copies” $!$.

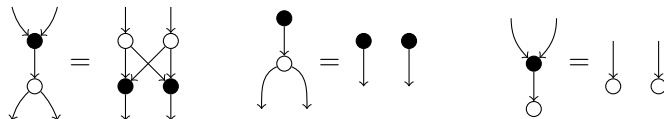


In a monoidal category, a monoid paired with a comonoid that behaves like copy and delete operations is called a *bialgebra*.

Definition 3.1.3. In a monoidal category \mathcal{V} , a *bialgebra* is a tuple $(X, \mu, \eta, \delta, \epsilon)$ where (X, μ, η) is a monoid, (X, δ, ϵ) is a comonoid and the following diagrams commute.

$$\begin{array}{ccccc}
 X \otimes X & \xrightarrow{\mu} & X & \xrightarrow{\delta} & X \otimes X \\
 \delta \otimes \delta \downarrow & & & & \uparrow \mu \otimes \mu \\
 X \otimes X \otimes X \otimes X & \xrightarrow{X \otimes \sigma \otimes X} & X \otimes X \otimes X \otimes X & & \\
 \\
 I & \xrightarrow{\eta} & X & & X \otimes X \\
 & \searrow \eta \otimes \eta & \downarrow \delta & & \swarrow \epsilon \otimes \epsilon \\
 & & X \otimes X & & X \xrightarrow{\epsilon} I
 \end{array}$$

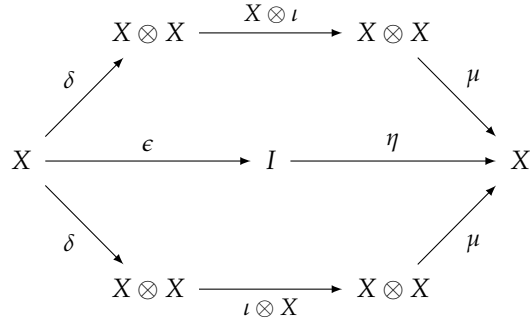
These conditions are the same as equations (3.4) and (3.5), but with $(X, \Delta, !)$ replaced by an *arbitrary* comonoid.



In the general case, it is worth noting that the comonoid within a bialgebra acts like a copy and deletion operation *relative to the associated monoid*, rather than globally. This distinction applied to the category of Hilbert spaces has deep connections to the *no-cloning theorem* for quantum mechanics. See, for example [1]. In a cartesian category, *any* monoid can be made into a bialgebra using the comonoid $(X, \Delta, !)$.

In the general case of monoidal categories, the inverse map is replaced with an arbitrary map, called an *antipode*. The analogue to a group object is a *Hopf algebra*.

Definition 3.1.4. A *Hopf algebra* is a tuple $(X, \mu, \eta, \delta, \epsilon, \iota)$ such that $(X, \mu, \eta, \delta, \epsilon)$ forms a bialgebra and the following diagram commutes:



Example 3.1.5. For any group G , we can form the group algebra $(K[G], \mu, \eta)$ as follows. Let $K[G]$ be a vector space in \mathbf{Vect}_K spanned by the elements $g \in G$. μ is then defined on basis vectors by the group multiplication: $\mu(g \otimes h) = gh$. $\eta(1) = e$, the group identity. This clearly forms a monoid in \mathbf{Vect}_K . We can turn this monoid into a bialgebra by adding maps that “copy” and “delete” the basis of group elements. We can make it into a Hopf algebra by adding a linear map that sends elements of the group basis to their inverse.

$$\begin{aligned}
\delta &:: g \mapsto g \otimes g \\
\epsilon &:: g \mapsto 1 \\
\iota &:: g \mapsto g^{-1}
\end{aligned}$$

The Hopf algebra $(K[G], \mu, \eta, \delta, \epsilon, \iota)$ captures all of the structure of the group algebra $K[G]$ without relying on “global” copy and deletion operations, which are not present in the (non-cartesian) monoidal category \mathbf{Vect}_K .

Examples 3.1.6. Here are a few more examples of Hopf algebras.

- Any group object in a cartesian monoidal category is automatically a Hopf algebra, with $\delta = \Delta$ and $\epsilon = !$.
- The universal enveloping algebra of a Lie algebra $U(\mathfrak{g})$ has a natural Hopf algebra structure, given by the unique extension of the following maps on \mathfrak{g} to $U(\mathfrak{g})$:

$$\delta(x) = (x \otimes 1) + (1 \otimes x) \quad \epsilon(x) = 0 \quad \iota(x) = -x$$

- More generally, quantum groups of the form $U_q(\mathfrak{g})$ for $q \in \mathbb{C}$ carry a Hopf algebra structure.

3.2 Frobenius Algebras

Associative algebras (A, μ, η) in a compact closed category of vector spaces always come with two canonical right representations: one over A and one over the dual space A^* . Frobenius algebras are associative algebras that are *self-dual*, i.e. these two representations are isomorphic. Frobenius algebras can be defined in any monoidal category. In order to talk about representations abstractly, it is convenient to use the (equivalent) language of modules.

Definition 3.2.1. For a monoid $\mathcal{A} = (A, \mu, \eta)$, a right \mathcal{A} -module (M, m) is a morphism $m : A \otimes M \rightarrow M$ such that the following diagrams commute.

$$\begin{array}{ccc}
 A \otimes A \otimes M & \xrightarrow{A \otimes m} & A \otimes M \\
 \mu \otimes M \downarrow & & \downarrow m \\
 A \otimes M & \xrightarrow{m} & M
 \end{array}
 \qquad
 \begin{array}{ccc}
 M & \xrightarrow{\eta \otimes M} & A \otimes M \\
 \searrow 1 & & \downarrow m \\
 & & M
 \end{array}$$

Left \mathcal{A} -modules are defined analogously, and for commutative monoids, the two concepts are equivalent. Graphically, the right module equations look like asymmetric versions of associativity and unit laws.

Definition 3.2.2. For a monoid $\mathcal{A} = (A, \mu, \eta)$, and right \mathcal{A} -modules (M, m) and (N, n) , an \mathcal{A} -module homomorphism is a morphism $\phi : M \rightarrow N$ such that:

$$\begin{array}{ccc}
 A \otimes M & \xrightarrow{A \otimes \phi} & A \otimes N \\
 m \downarrow & & \downarrow n \\
 M & \xrightarrow{\phi} & N
 \end{array}$$

Module homomorphisms pass through the module map.

(3.6)

Monoids in a monoidal category have a canonical right \mathcal{A} -module, namely the monoid itself.

This is called the *regular right \mathcal{A} -module*. In a compact closed category, a monoid comes with a canonical right module over its dual object $\tilde{\mu} : A \otimes A^* \rightarrow A^*$.

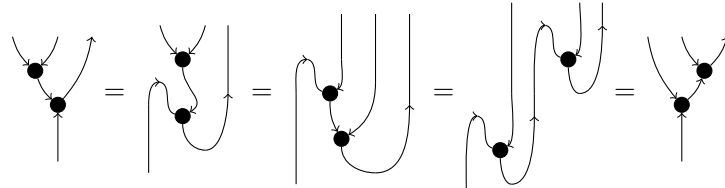
$$\tilde{\mu} := (A^* \otimes d) \circ (A^* \otimes \mu \otimes A^*) \circ (e \otimes A \otimes A^*)$$

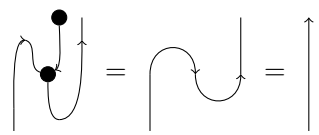
We shall call this the *dual right \mathcal{A} -module*. Graphically:

$$\begin{array}{c} \curvearrowright \\ \bullet \\ \uparrow \end{array} := \begin{array}{c} \curvearrowright \\ \bullet \\ \uparrow \end{array}$$

Proposition 3.2.3. $(A^*, \tilde{\mu})$ is a right \mathcal{A} -module.

Proof. We can show the two module conditions are satisfied graphically.

Associativity: 

Unit: 

□

Frobenius is generally credited with being the first to study the class of algebras whose regular and dual modules are isomorphic. This is a particularly important aspect of many associative algebras, including algebras of the form $K[G]$ for some finite group G . The formal definition is due to Brauer and Nesbitt [11]. An abstract version is provided here.

Definition 3.2.4. A Frobenius algebra in a compact closed category is a monoid (A, μ, η) equipped with a module isomorphism $\chi : (A, \mu) \rightarrow (A^*, \tilde{\mu})$.

The existence of a Frobenius algebra on an object A implies that A is *self-dual*, and furthermore, the monoid structure μ, η is compatible with that self-duality. Graphically, we represent χ and its inverse χ^{-1} as follows:

$$\chi := \begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array} \quad \chi^{-1} := \begin{array}{c} \uparrow \\ \bullet \\ \uparrow \end{array}$$

Graphically, the axioms of a Frobenius algebra are the monoid axioms given by (3.2) and three additional identities:

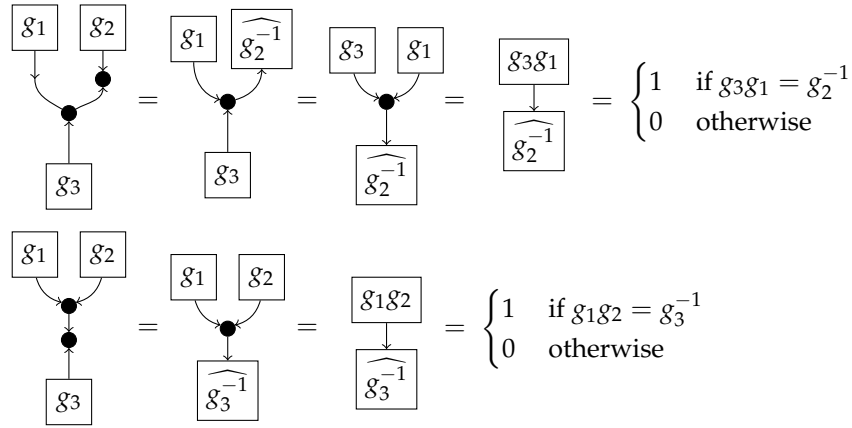
$$\begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array} = \downarrow \quad \begin{array}{c} \uparrow \\ \bullet \\ \uparrow \end{array} = \uparrow \quad \begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array} = \begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array} \quad (3.7)$$

Proposition 3.2.5. Let G be a finite group. Then, the associative algebra $K[G]$ is Frobenius.

Proof. Let $(K[G], \downarrow, \uparrow)$ be the associative algebra of the finite group G . For $g \in G$, let \hat{g} be the unique map defined on $h \in G$ as:

$$\hat{g}(h) = \begin{cases} 1 & \text{if } h = g \\ 0 & \text{otherwise} \end{cases}$$

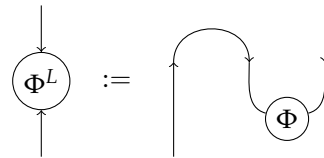
For χ , let $\chi(g) = \widehat{g^{-1}}$. Since $K[G]$ is finite-dimensional, this extends to an isomorphism $K[G] \cong K[G]^*$. It only remains to show that χ is a module homomorphism. It suffices to show that the two sides of equation (3.6) agree on all basis elements $g_i \in G$.



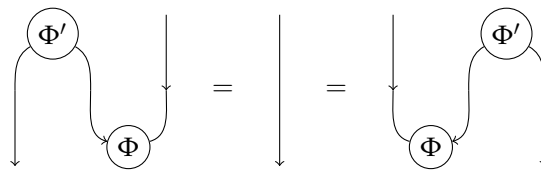
The proof is completed by noting that $g_3 g_1 = g_2^{-1}$ if and only if $g_1 g_2 = g_3^{-1}$. □

There are actually many equivalent definitions for a Frobenius algebra. Two additional definitions replace the isomorphism condition given above with a non-degeneracy condition.

Definition 3.2.6. A map of the form $\Phi : A \otimes A \rightarrow I$ is said to be non-degenerate if the following map is an isomorphism:

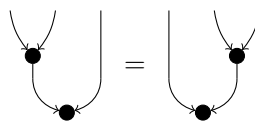


Equivalently, there exists a unique map $\Phi' : I \rightarrow A \otimes A$ such that:



Theorem 3.2.7. The following are equivalent.

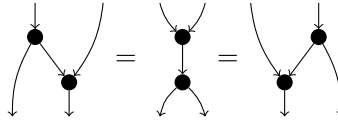
1. (A, μ, η, χ) is a Frobenius algebra,
2. (A, μ, η) is a monoid with non-degenerate map $\Phi : A \otimes A \rightarrow I$ that is associative with respect to the monoid:



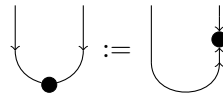
3. (A, μ, η) is a monoid with a map $\epsilon : A \rightarrow I$ such that the following is non-degenerate:



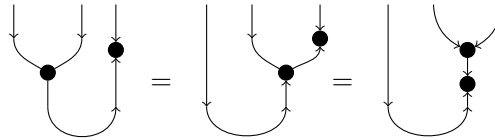
4. and (A, μ, η) is a monoid, (A, δ, ϵ) is a comonoid satisfying the Frobenius identity:



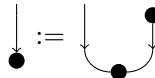
Proof. (1 \Rightarrow 2) Define Φ in terms of χ as:



The associativity of Φ follows from the definition of dual module and the module homomorphism identity in equation (3.7).

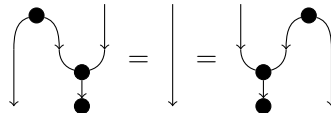


(2 \Rightarrow 3) Let ϵ be defined as:

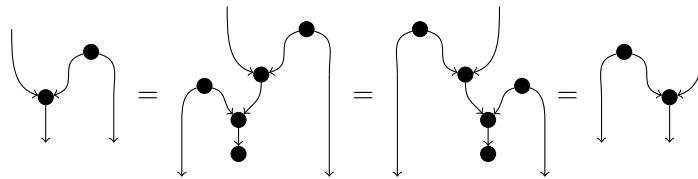


It can then easily be shown that $\epsilon \circ \mu = \Phi$, which is non-degenerate.

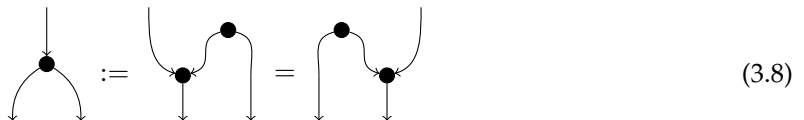
(3 \Rightarrow 4) Let $\Phi' : I \rightarrow A \otimes A$ be the unique map such that:



We can show this induced cap commutes horizontally with the multiplication:

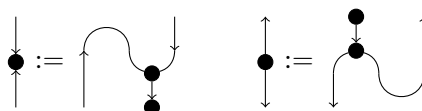


Define δ as follows:

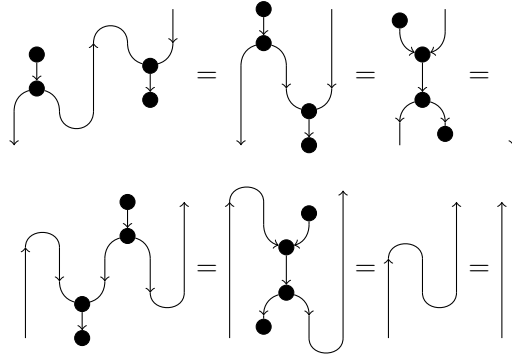


Then it follows from the monoid identities on (A, μ, η) that (A, δ, ϵ) is a comonoid and that the Frobenius identity holds.

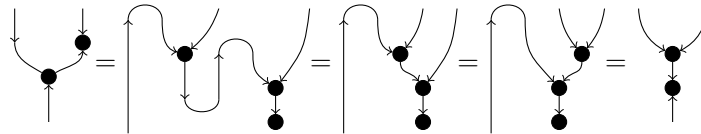
(4 \Rightarrow 1) Finally, let the isomorphism χ and its inverse be defined as follows:



It follows from the Frobenius identity and unit laws that this is indeed an isomorphism.



The only thing that remains to be checked is that this is an isomorphism of *modules*:

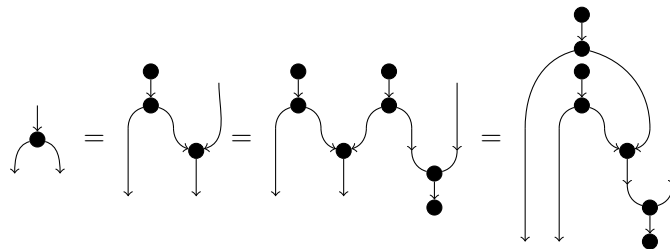


□

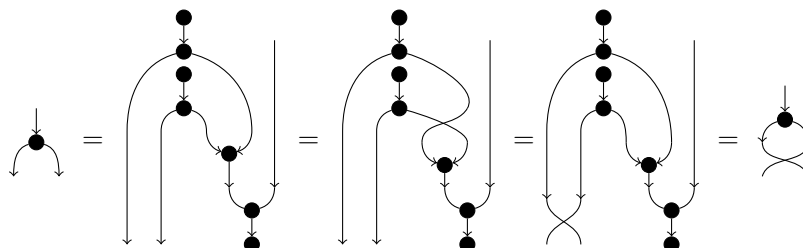
The equation from definition 4 first appeared in a paper by Carboni and Walters [14] in the context of categories of relations, though it was only later discovered to be a characterising equation for Frobenius algebras. This presentation is particularly interesting because, like in the case of bialgebras, it consists of (1) a monoid, (2) a comonoid, and (3) a rule for how they interact, called a *distributive law*. In definition 4, a Frobenius algebra consists of a monoid and a comonoid. Therefore a Frobenius algebra could be *commutative* or *cocommutative*. While these might seem like distinct conditions, it turns out they are the same.

Theorem 3.2.8. *For a Frobenius algebra $(A, \mu, \eta, \delta, \epsilon)$, (A, μ, η) is a commutative monoid if and only if (A, δ, ϵ) is a cocommutative comonoid.*

Proof. First, we use Equation (3.8) to derive another form for δ in terms of μ .



Cocommutative then trivially follows from commutativity:



The opposite implication is the same proof, upside-down. □

It is worth noting that this is *not* the case for bialgebras or Hopf algebras. As a simple counter-example, consider any non-commutative monoid in a cartesian category. The cocommutative monoid $(\Delta, !)$ automatically makes this into a bialgebra.

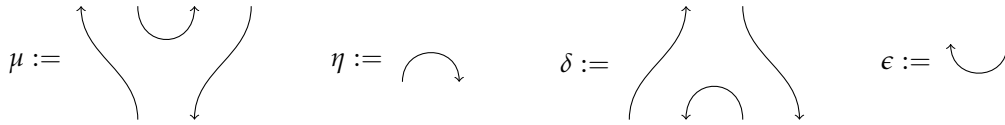
Example 3.2.9. Revisiting the group algebra example, we can define the rest of the Frobenius algebra structure as follows.

$$\delta :: g \mapsto \sum_{g_1, g_2 \in G, g_1 g_2 = g} g_1 \otimes g_2$$

$$\epsilon = \hat{\epsilon}$$

The comultiplication can be thought of an averaging operation. It takes a group element to a sum over all of its possible factorisations.

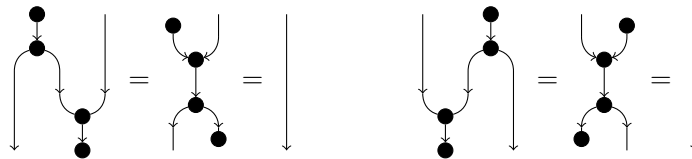
Example 3.2.10. Any compact closed category automatically has a Frobenius algebra on $A^* \otimes A$ given by:



In a \dagger -monoidal category, we can introduce the notion of a \dagger -Frobenius algebra, where the monoid structure is just the dagger of the comonoid structure.

Definition 3.2.11. A Frobenius algebra $(A, \mu, \eta, \delta, \epsilon)$ is called a \dagger -Frobenius algebra if $\mu = \delta^\dagger$ and $\eta = \epsilon^\dagger$.

As we have already seen in the proof of Theorem 3.2.7, the Frobenius identity implies that the Frobenius “cap” and “cup” maps form a compact structure.

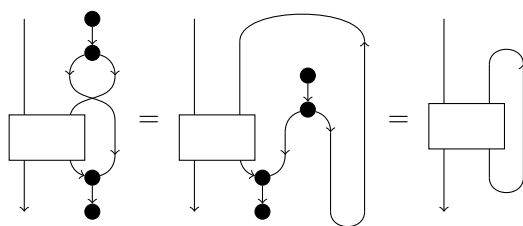


In fact, the Frobenius algebra induces a *self-dual* compact structure, i.e. $A^* = A$. Using this compact structure, we can define a transposition operation $(-)^{\bullet T}$ relative to a particular Frobenius algebra.

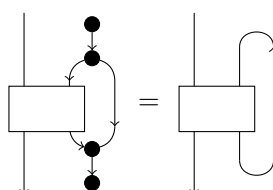
(3.9)

The situation here is a bit delicate. Whereas in most categories there is a canonical choice of a compact structure when A and A^* are distinct objects (e.g. a vector space and its dual space), defining a compact structure when A and A^* are the same object often involves a choice. Different Frobenius algebras defined on a single object A will often define *different* compact structures. It is a well-known fact that for finite-dimensional vector spaces, there is no canonical isomorphism connecting a space to its dual space. Picking a self-dual compact structure corresponds to choosing a *particular* isomorphism $A \cong A^*$.

The distinction between compact structures, where duals are defined up to isomorphism, and (non-canonical) self-dual compact structures is worth bearing in mind, particularly with regards to the Frobenius algebras defined in Part II. However, the situation is simpler when it comes to the trace operation. In Equation 2.6, we showed that any compact structure can be used to construct a trace operation. It turns out that the compact structures generated by Frobenius algebras (and more generally, *any* compact structures) always generate *the same* trace.

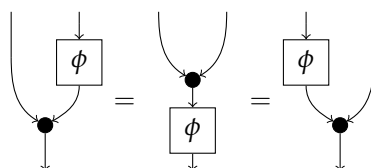


In the case of commutative Frobenius algebras, we usually represent the trace without a twist on the cap:



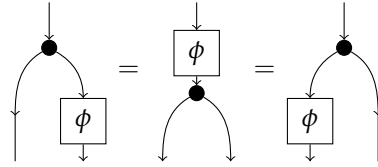
Often it is useful to consider maps that can pass freely through the Frobenius algebra structure. That is, we consider maps that are module endomorphisms of the regular left and right \mathcal{A} modules. By analogy to phase gates in quantum circuits, we call such maps *phases*.

Definition 3.2.12. For a Frobenius algebra $\mathcal{A} = (A, \mu, \eta, \delta, \epsilon)$, a map $\phi : A \rightarrow A$ is called a *phase* for \mathcal{A} if:

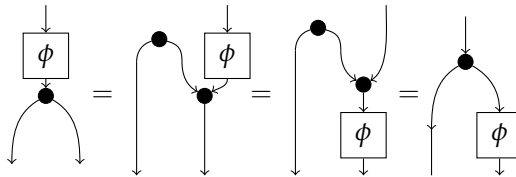


We shall see the connection between abstract phases and quantum phase gates in sections 7.2 and 7.3.

Proposition 3.2.13. *A phase for a Frobenius algebra \mathcal{A} is also a comodule endomorphism for δ , considered as a left and right comodule.*



Proof. The proof follows straightforwardly from the Frobenius identities.

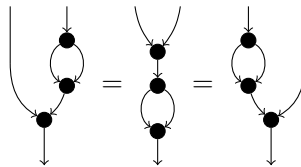


The left comodule identity follows similarly. □

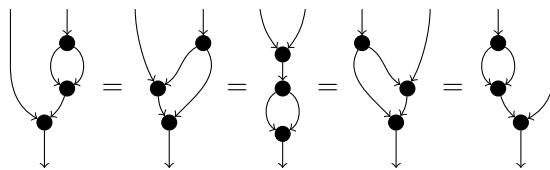
There is a canonical phase associated with any Frobenius algebra called its *loop map*.



Proposition 3.2.14. *For a Frobenius algebra $\mathcal{A} = (A, \mu, \eta, \delta, \epsilon)$, the loop map is a phase for \mathcal{A} . I.e.:*



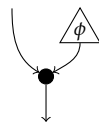
Proof. The proof follows from the Frobenius identity.



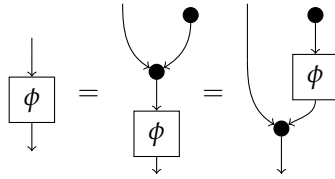
□

For commutative Frobenius algebras, phases can always be expressed as a left (or right) multiplication by a point.

Proposition 3.2.15. *All phases for a given commutative Frobenius algebra are of the following form:*



Proof. A map of the above form is clearly a phase. Let $\varphi : A \rightarrow A$ be an arbitrary phase map. Then:

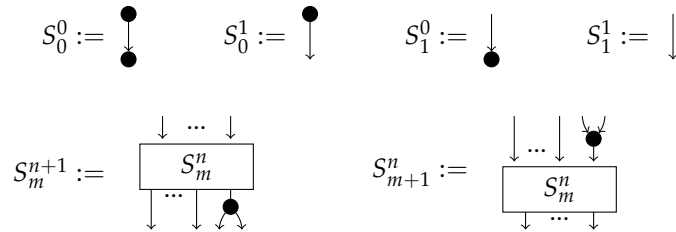


□

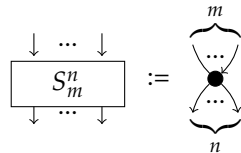
3.2.1 Normal Form for Frobenius Algebras

We shall primarily be interested in *commutative Frobenius algebras* (CFAs). The primary purpose of this section is to show that CFAs have particularly nice normal forms. To do that, we will state and prove the so-called *spider theorem*. First, we introduce the notion of a *spider*.

Definition 3.2.16. For a commutative Frobenius algebra \mathcal{A} , a *spider* is a map S_m^n defined as follows.

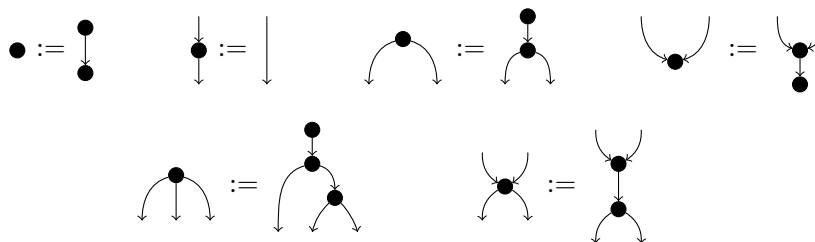


We represent the maps S_m^n as single dots, with m in-edges and n out-edges.



Since a CFA is a commutative, associative, cocommutative, and coassociative, we can freely interchange edges.

Examples 3.2.17. Some spiders of various arities:



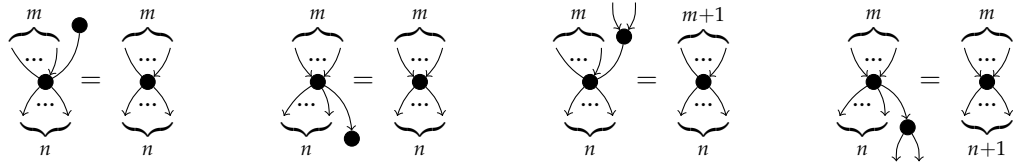
As a minor technical point, it is occasionally necessary to distinguish morphisms in a monoidal category from their actual representations as string diagrams. For this section, we shall use D, D', \dots to represent formal string diagrams and $|D|, |D'|, \dots$ to represent the associated morphisms in a monoidal category.

Definition 3.2.18. For a commutative Frobenius algebra $\mathcal{A} = (A, \mu, \eta, \delta, \epsilon)$, an \mathcal{A} -diagram D is a string diagram whose vertices are all labeled $\mu, \eta, \delta, \epsilon$. An \mathcal{A} -tree is an \mathcal{A} -diagram that contains

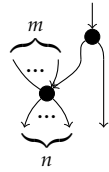
no cycles. An \mathcal{A} -tree that is formally equal to the diagram of S_m^n for some m, n is said to be in *spider-normal form*.

Lemma 3.2.19. *Suppose D is a connected \mathcal{A} -tree with m in-edges and n out-edges. Then $|D| = S_m^n$. I.e. the value of an \mathcal{A} -tree is uniquely determined by its input and output arities.*

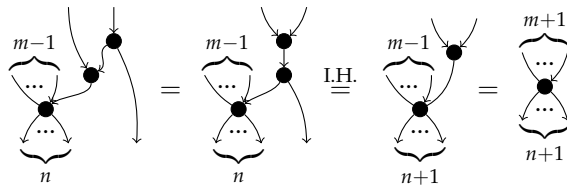
Proof. We proceed by induction on the number of vertices in D . Note that any one-vertex \mathcal{A} -tree is trivially equal to a spider S_1^0, S_0^1, S_2^1 , or S_1^2 . Thus, assume for an \mathcal{A} -tree containing N vertices, $|D| = S_m^n$. We show that for an \mathcal{A} -tree D' containing $N + 1$ vertices, $|D'| = S_{m'}^{n'}$. Since a spider is commutative and cocommutative, we can assume without loss of generality that the additional vertex is composed on the rightmost leg above or below the spider. These cases follow trivially from associativity, unitality, and the definition of spiders:



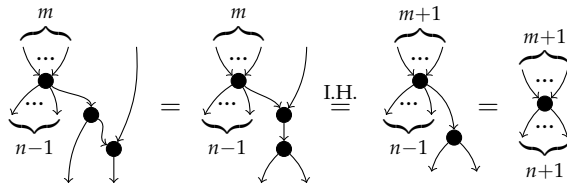
The only cases remaining are post-composition by μ and pre-composition by δ . First, consider pre-composition with δ :



If $m = 0$, this is already in spider-normal form. So, consider the case where $m \geq 1$. By definition of S_m^n , we can pull out a multiplication. Applying the Frobenius identity and the induction hypothesis:



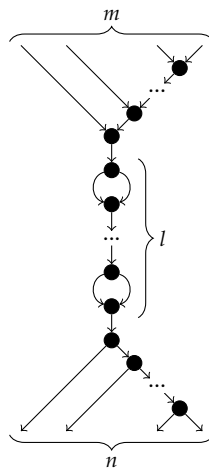
We complete the proof by applying the same method upside-down for post-composition with μ .



□

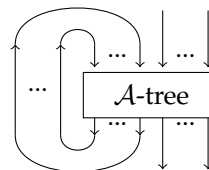
Because of this lemma, we could have equivalently defined S_m^n in Definition 3.2.16 as $S_m^n = |D|$ for any \mathcal{A} -tree D with m inputs and n outputs. With the help of a normal form result for \mathcal{A} -trees and Proposition 3.2.14, we are ready to state a general normal form result for commutative Frobenius algebras.

Theorem 3.2.20. *For a commutative Frobenius algebra \mathcal{A} , any connected \mathcal{A} -diagram is equivalent to one of this form:*

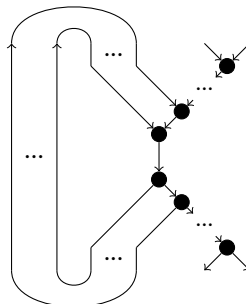


Proof. We first prove a small identity relating to traces:

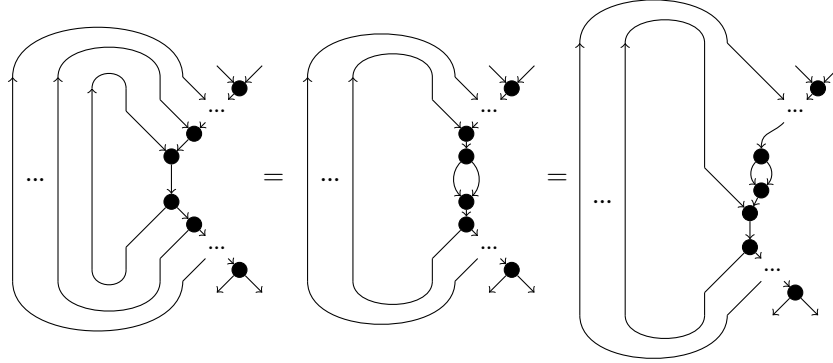
Using the axioms of a compact closed category, we can deform any \mathcal{A} -diagram into an \mathcal{A} -tree with traces.



Applying Lemma 3.2.19, we have:



We can then turn each of the traces into a loop map, using equation (3.10), then push the loops up using Proposition 3.2.14.



Repeating the process for each of the traces, we obtain a diagram in normal form. □

3.2.2 Special and Anti-special Commutative Frobenius Algebras

For any monoid in a traced category, we can define a map $\kappa : A \otimes A \rightarrow I$:

$$\kappa := \begin{array}{c} \text{---} \\ \diagdown \quad \diagup \\ \bullet \\ \diagup \quad \diagdown \\ \text{---} \\ \bullet \\ \diagdown \quad \diagup \\ \text{---} \end{array} \quad (3.11)$$

In \mathbf{FVect}_K , any map from $A \otimes A$ to the base field K is called a *bilinear form*. The map given by (3.11) is a particularly important bilinear form called the *Killing form*¹. This form plays a particularly important role in the representation theory of an algebra. For an algebraically closed field K , the Killing form of a finite-dimensional associative K -algebra is non-degenerate if and only if that algebra is semisimple. Since the Killing form automatically associates with the multiplication, any algebra with a non-degenerate Killing form (i.e. any finite-dimensional semisimple algebra) is automatically Frobenius. However, the converse is not true. There are many interesting Frobenius algebras that have degenerate Killing forms. First, note that we can relate the rank of the Killing form of a commutative Frobenius algebra with that of the loop map defined in the previous section.

$$\text{rank} \left(\begin{array}{c} \text{---} \\ \diagdown \quad \diagup \\ \bullet \\ \diagup \quad \diagdown \\ \text{---} \\ \bullet \\ \diagdown \quad \diagup \\ \text{---} \end{array} \right) = \text{rank} \left(\begin{array}{c} \text{---} \\ \diagdown \quad \diagup \\ \bullet \\ \diagup \quad \diagdown \\ \text{---} \\ \bullet \\ \diagdown \quad \diagup \\ \text{---} \end{array} \right) = \text{rank} \left(\begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array} \right)$$

We shall primarily focus on the minimal and maximal cases of this rank: the cases where the loop map is full-rank or rank one. It is never rank zero, except in the case of the zero-dimensional space. We wish to abstract the notions of full-rank and rank one to an arbitrary category. Clearly full-rank maps are just isomorphisms. A linear map is rank-one if and only if it can be factored through the base field:

¹The Killing form is much more commonly defined for Lie algebras than for associative algebras. The Killing form of an associative algebra is sometimes called simply its *trace form*.

$$\begin{array}{ccc}
 A & \xrightarrow{\zeta} & K \\
 & \searrow f & \downarrow v \\
 & & B
 \end{array}$$

We therefore define disconnected morphisms as an abstraction of rank-one linear maps.

Definition 3.2.21. A morphism $f : A \rightarrow B$ in a monoidal category is called *disconnected* if it factors through the tensor unit.

$$\begin{array}{c}
 \downarrow \\
 \boxed{f} \\
 \downarrow
 \end{array}
 =
 \begin{array}{c}
 \downarrow \\
 \nabla \\
 \triangle \\
 \downarrow
 \end{array}$$

For a Frobenius algebra, having an invertible loop is the abstract analogue to being semisimple. In particular, we shall consider commutative Frobenius algebras whose loop map is equal to the identity.

Definition 3.2.22. A *special commutative Frobenius algebra* (SCFA) is a commutative Frobenius algebra $\mathcal{A} = (A, \mu, \eta, \delta, \epsilon)$ such that $\mu \circ \delta = 1_A$. Graphically:

$$\begin{array}{c}
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow
 \end{array}
 =
 \begin{array}{c}
 \downarrow \\
 \downarrow
 \end{array}
 \tag{3.12}$$

This is not a great loss of generality, given the following theorem.

Theorem 3.2.23. For any commutative Frobenius algebra $\mathcal{A} = (A, \mu, \eta, \delta, \epsilon)$ such that $\mu \circ \delta$ is invertible, there exists an invertible phase L such that $(A, \mu, \eta, \delta \circ L^{-1}, \epsilon \circ L)$ is an SCFA.

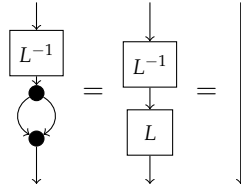
Proof. Since \mathcal{A} is semisimple, the loop map is invertible. Therefore, let $L = \mu \circ \delta$. Module endomorphisms are closed under inversion, so since L is a phase, L^{-1} is a phase. For, we show $(A, \delta \circ L^{-1}, \epsilon \circ L)$ is a comonoid:

$$\begin{array}{c}
 \downarrow \\
 \boxed{L^{-1}} \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow
 \end{array}
 =
 \begin{array}{c}
 \downarrow \\
 \boxed{L^{-1}} \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow
 \end{array}
 =
 \begin{array}{c}
 \downarrow \\
 \boxed{L^{-1}} \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow
 \end{array}
 =
 \begin{array}{c}
 \downarrow \\
 \boxed{L^{-1}} \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow
 \end{array}
 =
 \begin{array}{c}
 \downarrow \\
 \boxed{L^{-1}} \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow
 \end{array}
 =
 \begin{array}{c}
 \downarrow \\
 \boxed{L^{-1}} \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow
 \end{array}
 =
 \begin{array}{c}
 \downarrow \\
 \downarrow
 \end{array}$$

The Frobenius identity:

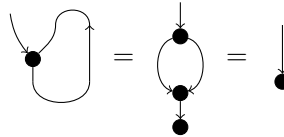
$$\begin{array}{c}
 \downarrow \\
 \boxed{L^{-1}} \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow
 \end{array}
 =
 \begin{array}{c}
 \downarrow \\
 \boxed{L^{-1}} \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow
 \end{array}
 =
 \begin{array}{c}
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow
 \end{array}
 =
 \begin{array}{c}
 \downarrow \\
 \boxed{L^{-1}} \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow
 \end{array}
 =
 \begin{array}{c}
 \downarrow \\
 \boxed{L^{-1}} \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow \\
 \bullet \\
 \downarrow
 \end{array}$$

Specialness follows by definition of L .



□

It is also worth noting that for special Frobenius algebras, defining just the monoid part (or just the comonoidal part) is enough to fix the entire structure. This is because the partial trace of μ is equal to ϵ .



By condition (3.) of Theorem 3.2.7, (A, μ, η, ϵ) is enough to define the entire Frobenius algebra.

Example 3.2.24. Fix a basis e_i for a finite-dimensional vector space V . We can define the e_i to be the vectors “copied” by δ and “deleted” by ϵ .

$$\delta :: e_i \mapsto e_i \otimes e_i \qquad \epsilon :: e_i \mapsto 1$$

This clearly forms a cocommutative comonoid. We can complete the Frobenius algebra by letting the e_i be a basis of idempotents of μ .

$$\mu :: e_i \otimes e_i \mapsto e_i \qquad \eta :: \sum e_i$$

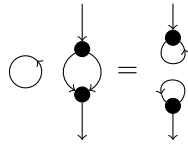
This forms a commutative Frobenius algebra, and by definition, $\mu\delta = 1_V$. We can do this for any basis e_i , and its a well-known fact that *any* semisimple commutative algebra (A, μ, η) over an algebraically closed field has a basis of idempotents, summing to η , so in particular, all SCFAs are of this form.

Now, we consider the other extreme: the case where the loop map is disconnected:

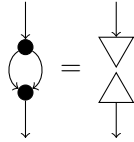
(3.13)

Due to a theorem by Herrmann [30], we can actually obtain an explicit form for equation (3.13).

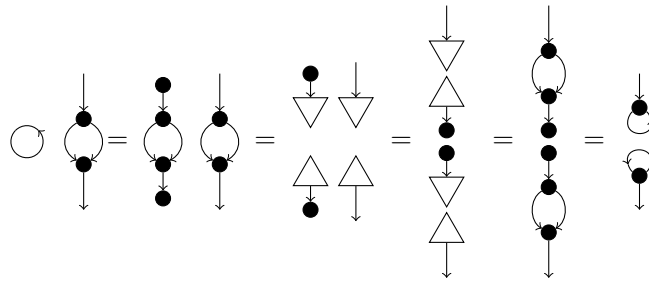
Theorem 3.2.25. *Let \mathcal{A} be a commutative Frobenius algebra with a disconnected loop map. Then the following equation holds:*



Proof. Assume the following equation holds, for any maps $x : A \rightarrow I, y : I \rightarrow A$:



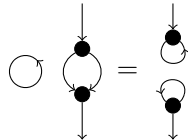
Then



□

A commutative Frobenius algebra with a disconnected loop is called *anti-special*.

Definition 3.2.26. An *anti-special commutative Frobenius algebra* (ACFA) is a commutative Frobenius algebra such that:



In addition to having a unit and a counit, anti-special Frobenius algebras have canonical disconnecting points which we shall refer to as the *anti-unit* and *anti-counit*.

Definition 3.2.27. For an ACFA, the *anti-unit* $\tilde{\eta}$ and the *anti-counit* $\tilde{\epsilon}$ are defined as follows:

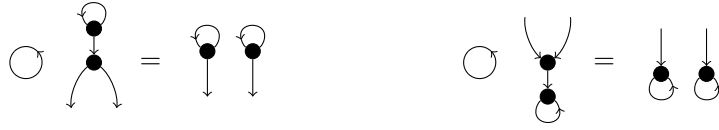


Special and anti-special Frobenius algebras have well-behaved normal forms.

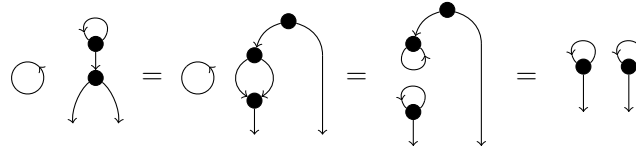
Theorem 3.2.28. For an SCFA \mathcal{S} , any connected \mathcal{S} -diagram is equivalent to a spider.

Proof. Any connected \mathcal{S} -diagram is equivalent to one in the form given in Theorem 3.2.20. We can then use equation (3.12) to remove all of the loops. □

Lemma 3.2.29. For an ACFA \mathcal{A} , the comultiplication map copies the anti-unit and the multiplication map copies the anti-counit.

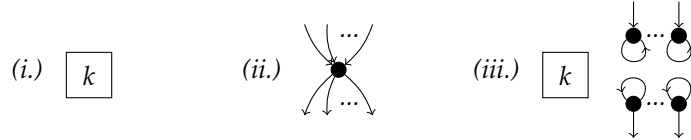


Proof. Follows straightforwardly by an application of Theorem 3.2.20 and the anti-specialness condition.

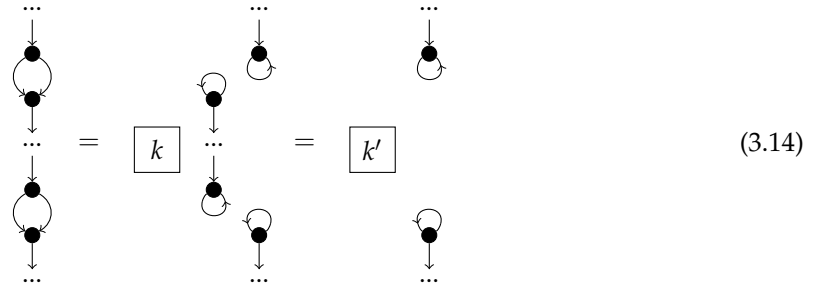


The upside-down equation is proved similarly. □

Theorem 3.2.30. Suppose \bigcirc is an invertible scalar. For an ACEA \mathcal{A} , any connected \mathcal{A} -diagram is equivalent to one of the following, for scalar map $k : I \rightarrow I$.



Proof. If a connected \mathcal{A} -diagram contains no loops, it is equivalent to a spider by Theorem 3.2.20. An \mathcal{A} -diagram containing more than one loop is equivalent to a scalar multiple of two disconnected diagrams containing one loop each.



Similarly, a diagram containing just a single loop can be made into two disconnected diagrams containing single loops.



If the diagram has no inputs or outputs, it is in the form of (i.), so assume it has at least one input or output. In that case, any diagram equivalent to the RHS of (3.14) or (3.15) can be put into the form of (iii.) using Lemma 3.2.29. □

Part I

Graphical Languages and Rewriting

Chapter 4

Rewrite Systems

Rewrite systems provide a model of computation that is particularly well suited to formalising dynamical systems, computing algebraic identities, and constructing proofs by automated or semi-automated means. The most well-studied type of rewrite systems are *term rewrite systems*. A term rewrite system consists of a set of *generators* (i.e. symbols with arities), *variables*, and *rewrite rules* between terms formed from generators and variables.

Term rewrite systems have applications in the study of programming languages, computer algebra systems, automated theorem proving, and many other areas of theoretical computer science. Rewriting terms is essentially the same as rewriting trees, and it was shown that many of these “tree” rewriting techniques could actually be applied to arbitrary graphs, or even objects of more general categories.

In this chapter, we shall review some of the basic principals of rewrite systems and the double-pushout approach to graph rewriting. We will then illustrate how the DPO approach is always well-defined in a particular class of graph-like categories called adhesive categories. However, to define a category suitable for rewriting string graphs, we shall need a weaker notion than adhesivity. For that reason, we introduce partial adhesive categories, and show how these categories inherit “enough adhesivity” from their ambient adhesive categories to do DPO rewriting.

4.1 Term Rewriting

Definition 4.1.1. A *term signature* $\Sigma = (G, a : G \rightarrow \mathbb{N})$ consists of a set G of *generators* and a function a assigning each generator an arity.

We define the set of *terms* for a signature and a set of variables recursively.

Definition 4.1.2. For a term signature $\Sigma = (G, a)$ and a set X of variables, we can form the set $T(\Sigma, X)$ of terms as follows:

- For all $x \in X$, x is a term.

- For $g \in G$ such that $a(g) = n$ and terms $t_i \in T(\Sigma, X)$, $g(t_1, \dots, t_n)$ is a term.

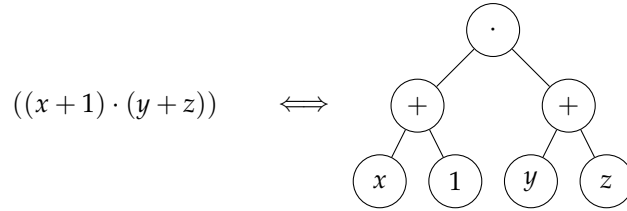
Variables are place-holders for other terms, i.e. other elements of $T(\Sigma, X)$. The mechanism by which variables are assigned values is called *substitution*.

Definition 4.1.3. For a set of terms $T(\Sigma, X)$, a function $\sigma : X \rightarrow T(\Sigma, X)$ is called a *substitution*. It can be lifted to a function $\hat{\sigma} : T(\Sigma, X) \rightarrow T(\Sigma, X)$ as follows. For a term $t \in T(\Sigma, X)$, replace every occurrence of any variable $x \in X$ in the term with $\sigma(x)$. The resulting term is $\hat{\sigma}(t)$.

A rewrite rule is a pair $(l, r) \in T(\Sigma, X) \times T(\Sigma, X)$, usually written $l \rightarrow r$. A rewrite rule can be used to rewrite a term $t \in T(\Sigma, X)$ to a new term $t' \in T(\Sigma, X)$. This occurs in two stages.

1. **Matching:** A substitution σ is chosen such that $\hat{\sigma}(l)$ occurs as a sub-term of t .
2. **Replacement:** The occurrence of $\hat{\sigma}(l)$ in t is replaced by $\hat{\sigma}(r)$.

We can elaborate on this process a bit. Terms are essentially just trees:



For a given tree, a subtree can be uniquely identified by its *lexicographic position*. The lexicographic position of a sub-tree t' of t is simply a list of natural numbers $p = [c_0, c_1, \dots, c_{n-1}]$. The root of t' can then be located by taking the c_0 -th child of the root vertex of t , the c_1 -th child of that vertex, and so on until c_{n-1} . In other words, if we represent a term t as a list of lists,

$$t' = t[c_0][c_1] \dots [c_{n-1}]$$

Thus, in the case of term rewriting, finding a matching l in $l \rightarrow r$ is simply a case of identifying a substitution and lexicographic position such that

$$t[c_0][c_1] \dots [c_{n-1}] = \hat{\sigma}(l)$$

Performing the rewrite is then just a case of replacing the subtree at that position:

$$t[c_0][c_1] \dots [c_{n-1}] := \hat{\sigma}(r)$$

A set \mathcal{R} of rewrite rules is called a *rewrite system*. We write $t \rightarrow_{\mathcal{R}} t'$ there exists a rule $l \rightarrow r \in \mathcal{R}$ that rewrites t into t' using the above procedure. This forms a relation $\rightarrow_{\mathcal{R}} \subseteq T(\Sigma, X) \times T(\Sigma, X)$ called a *reduction relation* of \mathcal{R} . It is not always the case that the LHS of some rule in \mathcal{R} will match a given term t . If there is no such matching, t is called *irreducible*. Otherwise, it is called *reducible*.

Example 4.1.4. Consider the algebraic theory of unital rings. This theory has two binary operations $(- + -)$ and $(- \cdot -)$ as well as two 0-ary operations (i.e. constants) 1 and 0. The usual ring axioms can be turned into a rewrite system by directing each of the equations.

1. $((x + y) + z) \rightarrow (x + (y + z))$
2. $((x \cdot y) \cdot z) \rightarrow (x \cdot (y \cdot z))$
3. $(x + 0) \rightarrow x$
4. $(0 + x) \rightarrow x$
5. $(x \cdot 1) \rightarrow x$
6. $(1 \cdot x) \rightarrow x$
7. $(x \cdot (y + z)) \rightarrow ((x \cdot y) + (x \cdot z))$
8. $((x + y) \cdot z) \rightarrow ((x \cdot z) + (y \cdot z))$

We can define a term

$$t := (0 + (((1 + y) + y) \cdot (x + z)))$$

Let $l \rightarrow r$ be rule 8 from above. To find a matching of l on t , first define a substitution:

$$\sigma :: \{x \mapsto (1 + y), y \mapsto y, z \mapsto (x + z)\}$$

Apply the substitution to l and r :

$$\begin{aligned}\hat{\sigma}(l) &= (((1 + y) + y) \cdot (x + z)) \\ \hat{\sigma}(r) &= (((1 + y) \cdot (x + z)) + (y \cdot (x + z)))\end{aligned}$$

Note that $\hat{\sigma}(l)$ now occurs as a subterm of t .

$$t = (0 + \hat{\sigma}(l))$$

We form a new term t' by replacing the occurrence of $\hat{\sigma}(l)$ with $\hat{\sigma}(r)$.

$$t' = (0 + \hat{\sigma}(r)) = (0 + (((1 + y) \cdot (x + z)) + (y \cdot (x + z))))$$

$\xrightarrow{*}_{\mathcal{R}}$ is used to represent the reflexive, transitive closure of $\rightarrow_{\mathcal{R}}$, and $\xleftarrow{*}_{\mathcal{R}}$ the reflexive, transitive, symmetric closure. In the case where \mathcal{R} represents the “directed version” of the axioms of an algebraic structure, it is often the goal to evaluate the truth of the following proposition:

$$t \xleftarrow{*}_{\mathcal{R}} t'$$

Or, “Is t equivalent to t' by the axioms of \mathcal{R} ?” As this is a statement of the word problem, it is not decidable for arbitrary rewrite systems \mathcal{R} . However, there are many “good” rewrite systems, where this proposition *is* decidable. Two characteristics of a rewrite system that make it “good” are *termination* and *confluence*.

Definition 4.1.5. A rewrite system \mathcal{R} is said to be *terminating* if there exists no infinite chain of rewrites:

$$t_1 \rightarrow t_2 \rightarrow \dots \rightarrow t_n \rightarrow \dots$$

In practice, one often proves termination by identifying a *reduction order* on terms.

Definition 4.1.6. A partially ordered set (P, \leq) is called *well-founded* (or Noetherian) if it has a smallest element and contains no infinite sequence of strictly decreasing elements:

$$p_1 > p_2 > \dots > p_n > \dots$$

Well-foundedness is usually defined by every non-empty subset $P' \subseteq P$ having at least one minimal element (i.e. an element that is not strictly greater than any other). Up to the Axiom of Choice, these two definitions are equivalent. Intuitively, a well-founded poset is a poset over which one can perform (generalised) induction. A standard example is (\mathbb{N}, \leq) , which corresponds to the usual notion of induction over the natural numbers.

Definition 4.1.7. Let (P, \leq) be a well-founded poset. A function $\omega : T(\Sigma, X) \rightarrow P$ is called a *reduction ordering* for a rewrite system \mathcal{R} if:

$$t_1 \rightarrow_{\mathcal{R}} t_2 \implies \omega(t_1) > \omega(t_2)$$

Clearly any rewrite system that admits a reduction ordering is terminating. Termination guarantees that even a naïve rewriting strategy (choosing rules at random, applying until there are no more matchings) will terminate. If we can rewrite a term t in any number of steps to an irreducible term t' , t' is called a *normal form* of t . However, with a rewrite system that is only terminating, there is no guarantee that two distinct sequences of rewrites will result in the same normal form. To get *unique* normal forms, we need an additional property called *confluence*.

Definition 4.1.8. A rewrite system \mathcal{R} is said to be *confluent* if sequences of rewrites starting with the same term can be rejoined. That is, for terms t, t_1, t_2 such that $t \xrightarrow{*} t_1$ and $t \xrightarrow{*} t_2$, there exists a term t' such that $t_1 \xrightarrow{*} t'$ and $t_2 \xrightarrow{*} t'$.

$$\begin{array}{ccc} t & \xrightarrow{*} & t_1 \\ \downarrow * & & \downarrow * \\ t_2 & \xrightarrow{*} & t' \end{array}$$

Theorem 4.1.9. *If a rewrite system \mathcal{R} is terminating and confluent, every term t has a unique normal form $t \downarrow$.*

Proof. The existence of at least one normal form is guaranteed by termination. Suppose t_1 and t_2 are normal forms for t . Then $t \xrightarrow{*} t_1$ and $t \xrightarrow{*} t_2$, so by confluence there exists t' such that $t_1 \xrightarrow{*} t'$ and $t_2 \xrightarrow{*} t'$. However, since t_1 and t_2 are irreducible, the only possibility is that $t_1 = t' = t_2$. \square

Terminating, confluent rewrite systems provide a decidable solution to the word problem.

Theorem 4.1.10. *For terms t_1 and t_2 , $t_1 \leftrightarrow^* t_2$ iff $t_1 \downarrow = t_2 \downarrow$.*

Proof. (\Leftarrow) follows from the definition of $(-)\downarrow$. For (\Rightarrow), assume $t_1 \leftrightarrow^* t_2$. Then, there exists a finite sequence of forward and backward rewrite steps between t_1 and t_2 .

$$t_1 \xleftarrow{*} p_1 \xrightarrow{*} q_1 \xleftarrow{*} \dots \xrightarrow{*} q_{n-1} \xleftarrow{*} p_n \xrightarrow{*} q_n \xleftarrow{*} t_2 \quad (4.1)$$

Note that the rewrite sequence consists of “peaks” p_i and “troughs” q_i . We proceed by induction on the number of peaks. An arbitrary rewrite sequence with n peaks, as in equation (4.1) can be reduced to a rewrite system with $n - 1$ peaks by applying confluence to $q_{n-1} \xleftarrow{*} p_n \xrightarrow{*} q_n$.

$$t_1 \xleftarrow{*} p_1 \xrightarrow{*} q_1 \xleftarrow{*} \dots \xrightarrow{*} q_{n-1} \xrightarrow{*} p'_n \xleftarrow{*} q_n \xleftarrow{*} t_2$$

If there are zero peaks, then there exists at most one trough q , and by confluence $t_1 \downarrow = q \downarrow = t_2 \downarrow$.

\square

4.2 Graph Rewriting

Definition 4.2.1. Let **Graph** be the category of (directed, multi-) graphs. It is defined as the functor category $[\mathbf{G}, \mathbf{Set}]$, for \mathbf{G} defined as:

$$E \begin{array}{c} \xrightarrow{s} \\ \xrightarrow{t} \end{array} V$$

E identifies the edges of the graph, and V the vertices. s and t are functions taking an edge to its source and target respectively. If $t(e) = v$ then e is called an *in-edge* of v and if $s(e) = v$ then e is called an *out-edge* of v .

For a graph $G : \mathbf{G} \rightarrow \mathbf{Set}$, we shall write V_G for $G(V)$, E_G for $G(E)$, etc. and drop the subscripts when it is clear from context. Since a graph homomorphism $f : G \rightarrow H$ is just a natural transformation, it is a pair of functions f_V, f_E such that:

$$\begin{array}{ccc} E_G & \xrightarrow{s} & V_G \\ f_E \downarrow & & \downarrow f_V \\ E_H & \xrightarrow{s} & V_H \end{array} \qquad \begin{array}{ccc} E_G & \xrightarrow{t} & V_G \\ f_E \downarrow & & \downarrow f_V \\ E_H & \xrightarrow{t} & V_H \end{array}$$

It is natural to ask if term rewriting (i.e. tree rewriting) techniques can be extended to arbitrary graphs. It turns out that most notions and techniques from term rewriting translate directly to graphs, but the concepts of matching and replacement become more complicated. This is because, unlike in the case of trees, there is no canonical “root vertex” of a graph, and hence no absolute notion of “position”. Proceeding by analogy to the term rewriting case, suppose we wish to apply a graph rewrite rule $L \rightarrow R$ to a graph G . This poses two problems:

1. How does one keep track of where a matching of the LHS of a graph rewrite rule has been made?
2. How does one decide where to attach to the RHS of a graph rewrite rule?

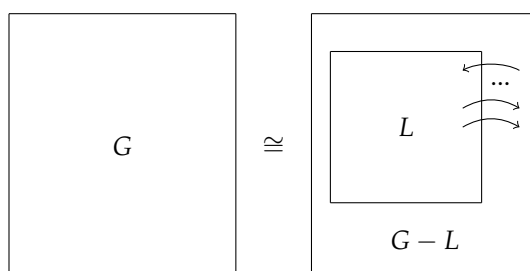
The first problem is solved by letting a matching be represented by an injective graph homomorphism $m : L \rightarrow G$, subject to certain conditions. The second problem is solved by requiring that all graph rewrite rules have a common subgraph embedded into the LHS and RHS called the *invariant subgraph* of the rewrite rule.

$$L \leftarrow I \hookrightarrow R$$

This invariant subgraph serves as “glue” to attach R to G after (the non-invariant part of) L has been removed. We can sketch out diagrammatically how this works. Suppose we have matching $m : L \rightarrow G$. Since m is an injection, we can think of L as a subgraph of G . Before we insert R , we must remove L . To do this, we might consider using the graph theoretic subtraction.

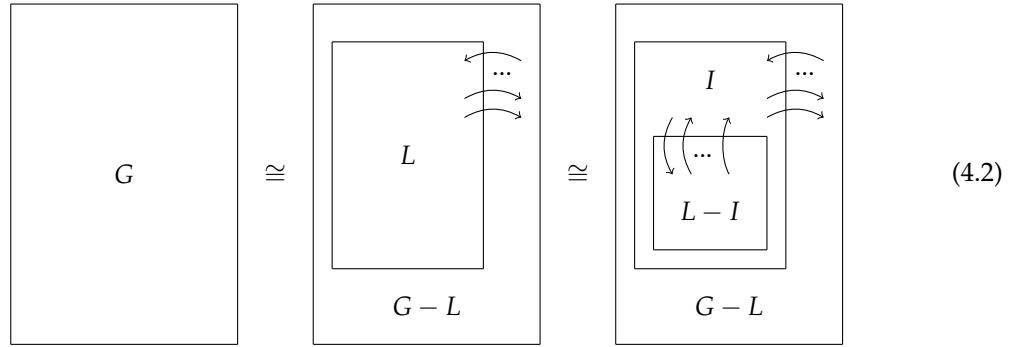
Definition 4.2.2. For a subgraph $G' \subseteq G$, the *graph theoretic subtraction* $G - G'$ is a new graph formed by removing G' , as well as any edges in to or out of G' , from G .

So G consists of a component L , a component $G - L$, and some edges between those components.

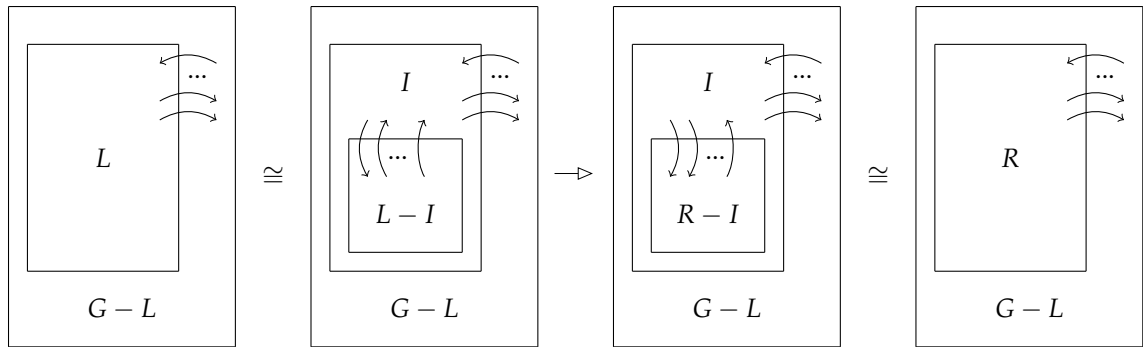


If we simply delete L from G , there is no way to keep track of how L was connected to G in the first place. To get around this, we treat the invariant subgraph of the rewrite rule as an *interface* for

L , and further decompose G :



The part of L that is not contained in I (i.e. $L - I$ and the edges between $L - I$ and I) is called the *interior* of L . Note that there are edges between G and I and there are edges between I and $L - I$, but there are no edges directly connecting G to $L - I$. We require this to be the case for any valid matching m of the rewrite rule. This is called the *no-dangling-edges* condition. Now, when we replace L with R , we know where the edges connected to I should go, since R also contains a copy of I .



This procedure can be formalised elegantly using pushouts. Begin by noting that the graph G with the interior of I removed is the (unique) graph G' such that the following square is a pushout:

$$\begin{array}{ccc}
 I & \hookrightarrow & L \\
 m' \downarrow & & \downarrow m \\
 G' & \hookrightarrow & G
 \end{array}$$

In other words, G is the result of gluing L and G' together along I . G' is called the *pushout complement* of $I \hookrightarrow L \xrightarrow{m} G$. Once we have G' and $m' : I \rightarrow G'$, we can glue on R by performing a second pushout:

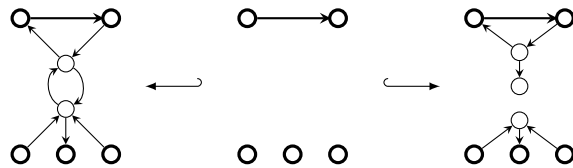
$$\begin{array}{ccc}
 I & \hookrightarrow & R \\
 m' \downarrow & & \downarrow \\
 G' & \longrightarrow & H
 \end{array}$$

So, to complete the rewrite $G \rightarrow H$, we first compute a pushout complement, then compute a pushout. This style of graph rewriting is called the *double pushout* (DPO) approach. We typically express the whole rewrite as a single DPO diagram:

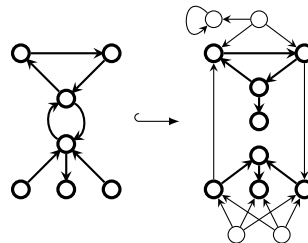
$$\begin{array}{ccccc}
 L & \xleftarrow{\quad} & I & \xrightarrow{\quad} & R \\
 m \downarrow & & \downarrow m' & & \downarrow \\
 G & \xleftarrow{\quad} & G' & \xrightarrow{\quad} & H
 \end{array}$$

Note that pushouts (and pushout complements) are defined up to isomorphism, so for $G \cong G'$, $H \cong H'$, $G \rightarrow H$ if and only if $G' \rightarrow H'$. As before, a set of graph rewrite rules \mathcal{R} is called a *graph rewrite system*, and we write $G \rightarrow_{\mathcal{R}} H$ if there exists a rule in \mathcal{R} that rewrites G to H , as above. The notions of confluence and termination are identical to those in Section 4.1, as well as the proofs of Theorems 4.1.9 and 4.1.10 for graphs, replacing term equality with graph isomorphism.

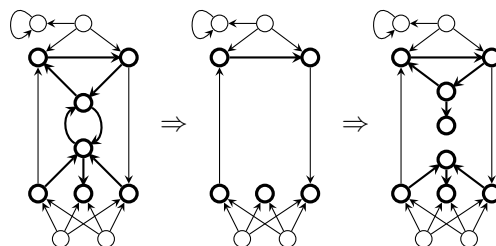
Example 4.2.3. Let the following be a rewrite rule $L \leftrightarrow I \hookrightarrow R$:



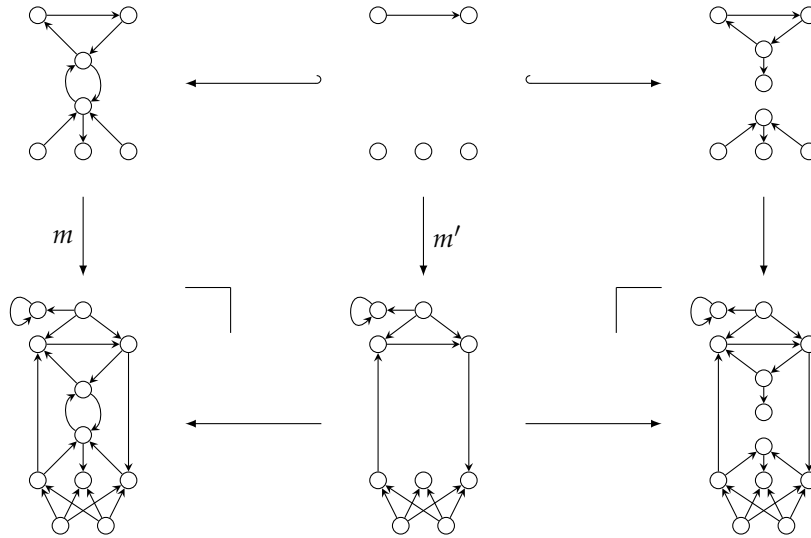
Then, we can find a matching of L on a bigger graph G :



We perform the rewrite by first removing the interior of L , then gluing R to remainder of the graph along the common subgraph I .



The full double pushout diagram looks like this:



4.3 Adhesive Categories

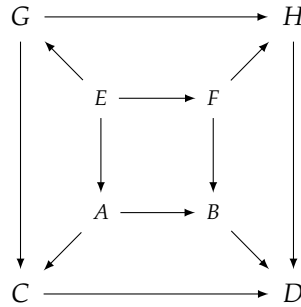
In the previous section, we discussed double-pushout rewriting in the category **Graph**. However, we may also wish to do rewriting on the objects of many categories that look a bit like the category of graphs, such as typed graphs, graphs with extra data, Petri nets, or objects of an arbitrary topos. There have been two main approaches to carrying out this abstraction. The first was to define DPO rewriting in the context of high-level replacement (HLR) systems, introduced by Ehrig et al [28]. The second approach, which we shall build on in this dissertation, relies on *adhesive categories*, introduced by Lack and Sobociński [44]. These two notions are actually compatible, as was shown with the definition of *adhesive HLR categories* in [59], which categories equipped with a class of monomorphisms that behave “adhesively”. Our construction of *partial adhesive categories* is somewhat in the spirit of adhesive HLR categories, but adhesive behaviour is localised to a certain class of well-behaved *spans* within the category rather than a certain class morphisms.

Adhesive categories provide a general context in which rewriting on graph-like structures is well-defined. Their definition relies on the notion of a special kind of pushout called a van Kampen square.

Definition 4.3.1. A van Kampen square (VK-square) is a pushout

$$\begin{array}{ccc}
 A & \longrightarrow & B \\
 \downarrow & & \downarrow \\
 C & \longrightarrow & D
 \end{array}$$

Such that for any commutative cube



where the back and left faces ($ABEF$ and $ACEG$) are pullbacks, the following are equivalent:

- the front and right faces ($CDGH$ and $BDFH$) are pullbacks
- the top face ($EFGH$) is a pushout

A pushout of a span $A \xleftarrow{f} B \xrightarrow{g} C$ where either f or g is a monomorphism is called a *pushout along a monomorphism*. An adhesive category is a category where pushouts along monomorphisms are van Kampen squares.

Definition 4.3.2. A category \mathcal{A} is said to be *adhesive* if

1. \mathcal{A} has pushouts along monomorphisms,
2. \mathcal{A} has pullbacks,
3. and pushouts along monomorphisms in \mathcal{A} are van Kampen squares.

At first sight, the definition of van Kampen squares can seem rather opaque, so it is useful to consider a concrete example. Suppose we have a set $X = A \cup B$ which is composed to two (possibly overlapping) subsets A and B . There are two equivalent ways to define a map into X :

1. For some set X' , simply define a function $f : X' \rightarrow X$.
2. For sets A', B' , define functions $f_A : A' \rightarrow A$ and $f_B : B' \rightarrow B$ such that f_A and f_B agree on $A \cap B$. I.e. for the restrictions $f_A|(A \cap B) : I \rightarrow A \cap B$, $f_B|(A \cap B) : I' \rightarrow A \cap B$, $I = I'$ and $f_A|(A \cap B) = f_B|(A \cap B)$.

First, we'll see how we can obtain (2.) from (1.). Starting with a map $f : X' \rightarrow X$, one can simply let f_A and f_B be the restrictions of f to the subsets A and B respectively. The restriction of a function to a subset of its codomain is just a pullback. We can therefore define f_A and f_B by this diagram:

$$\begin{array}{ccccc}
 A' & \longrightarrow & X' & \longleftarrow & B' \\
 f_A \downarrow & \lrcorner & \downarrow f & \llcorner & \downarrow f_B \\
 A & \longrightarrow & X & \longleftarrow & B
 \end{array} \tag{4.3}$$

To obtain (1.) from (2.), we start with maps f_A, f_B , such that $f_A|(A \cap B) = f_B|(A \cap B)$. Again using pullbacks to express restriction, this means the following diagram commutes (treating I from above as $A' \cap B'$):

$$\begin{array}{ccccc}
 A' & \longleftarrow & A' \cap B' & \longrightarrow & B' \\
 f_A \downarrow & & \lrcorner \downarrow \lrcorner & & \downarrow f_B \\
 A & \longleftarrow & A \cap B & \longrightarrow & B
 \end{array} \tag{4.4}$$

Let $X' = A' \cup B'$ and define f as:

$$f(x) = \begin{cases} f_A(x) & \text{if } x \in A \\ f_B(x) & \text{if } x \in B \end{cases}$$

This function is defined unambiguously, because if x is in both A and B then $f_A(x) = f_B(x)$. Since $X' = A' \cup B'$ is a pushout, we can equivalently define f as the induced map in the following diagram:

$$\begin{array}{ccccc}
 A' \cap B' & \longrightarrow & B' & & \\
 \downarrow & & \downarrow & \searrow f_B & \\
 A' & \longrightarrow & X' & & B \\
 \searrow f_A & & \downarrow f & & \downarrow \\
 & & A & \longrightarrow & X
 \end{array} \tag{4.5}$$

It is easy to show that these two procedures are the inverses of each other. If we combine diagrams (4.3), (4.4), and (4.5) into a single diagram, we get the commutative cube from Definition 4.3.1.

$$\begin{array}{ccc}
 A' & \xrightarrow{\quad} & X' \\
 \downarrow f_A & \swarrow & \downarrow f \\
 & A' \cap B' \xrightarrow{\quad} B' & \\
 & \downarrow & \searrow f_B \\
 & A \cap B \xrightarrow{\quad} B & \\
 \downarrow & \swarrow & \downarrow \\
 A & \xrightarrow{\quad} & X
 \end{array}$$

Now, as in Definition 4.3.1, suppose we have a commutative cube where the bottom face is a pushout (i.e. $X = A \cup B$) and the left and back faces are pullbacks (i.e. $f_A|(A \cap B) = f_B|(A \cap B)$). We can now read the van Kampen square condition as follows: the front and right faces are pullbacks (i.e. f restricts to f_A and f_B along A and B respectively) if and only if the top face is a pushout (i.e. $X' = A' \cup B'$).

Examples 4.3.3. Some examples of adhesive categories:

- **Set**, **Graph**, and **Set*** are adhesive categories.
- For adhesive categories \mathcal{C} and \mathcal{D} and an object $X \in \mathcal{C}$ then $\mathcal{C} \times \mathcal{D}$, X/\mathcal{C} , and \mathcal{C}/X are adhesive.
- For a small category \mathbb{X} and an adhesive category \mathcal{C} , the functor category $[\mathbb{X}, \mathcal{C}]$ is adhesive.
- Any elementary topos is adhesive.

Note that unlike toposes, adhesive categories are stable under coslices. In particular, **Set*** is an adhesive category, but not a topos.

Adhesive categories are useful for double-pushout rewriting because pushout complements for pushouts along monomorphisms are unique, when they exist.

Definition 4.3.4. A *pushout complement* for a pair of arrows $A \xrightarrow{m} B \xrightarrow{g} D$, is an object C and a pair of arrows $A \xrightarrow{f} C \xrightarrow{n} D$ such that the following is a pushout:

$$\begin{array}{ccc} A & \xrightarrow{m} & B \\ f \downarrow & & \downarrow g \\ C & \xrightarrow{n} & D \end{array} \quad \lrcorner$$

The pushout complement above should be thought of as “subtracting B from D , modulo A ”. If we think of A as the interface of B , then another way to put it is “removing the interior of B from D ”.

Notation 4.3.5. We occasionally write $B +_{m,f} C$ for pushout of the span $B \xleftarrow{m} A \xrightarrow{f} C$ and $D -_{m,g} B$ for the pushout complement of $A \xrightarrow{m} B \xrightarrow{g} D$.

We first need a few basic lemmas before showing that pushout complements in an adhesive category are unique.

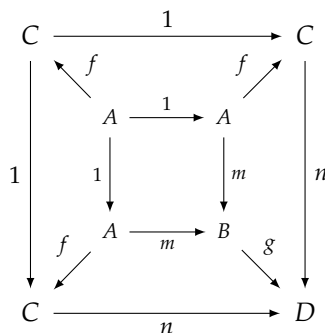
Lemma 4.3.6. [44] *In an adhesive category:*

- *monomorphisms are stable under pushout, and*
- *pushouts along monomorphisms are also pullbacks.*

Proof. Let m be a monomorphism, and let the following diagram be a pushout:

$$\begin{array}{ccc} A & \xrightarrow{m} & B \\ f \downarrow & & \downarrow g \\ C & \xrightarrow{n} & D \end{array} \quad \lrcorner$$

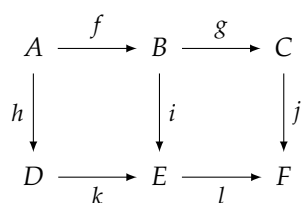
We need to show that n is mono and the above square is also a pullback. Construct a commutative cube, containing two copies of the given pushout square: one on the bottom face and one on the right face. Place a copy of f in the upper-left corner, and fill in the rest with identities.



All of the faces of this cube commute trivially. Commutative squares involving identities often (trivially) form pullbacks or pushouts. In particular, the top face is pushout, the left face is a pullback, and the back face is a pullback iff m is a monomorphism. Since m is defined to be a monomorphism and the pushout we started with is a VK-square, adhesivity shows that the front and right faces must be pullbacks. Since the front face is a pullback, n is a monomorphism and thus monomorphisms are stable under pushout. Furthermore, since the right face is the pushout we started with, we have also shown that pushouts along monomorphisms are pullbacks. \square

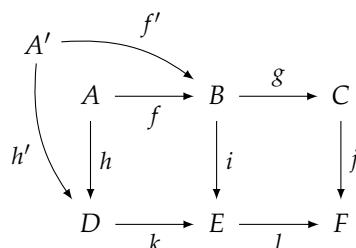
We now provide a few lemmas about pullbacks and pushouts that hold in *any* category.

Lemma 4.3.7. *For the following commutative diagram, where the right square is a pullback:*



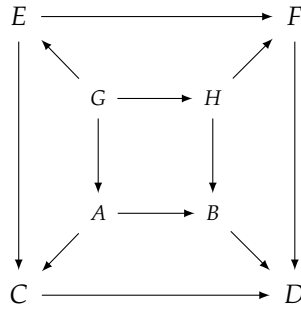
the outer square is a pullback iff the left square is a pullback.

Proof. Left implies outer is trivial. For the other direction, assume the outer square is a pullback. For an object A' let $f' : A' \rightarrow B$ and $h' : A' \rightarrow D$ be arrows such that $if' = kh'$.



Then gf' and h' form a cone under the outer pullback, so there exists a unique $u : A' \rightarrow A$ such that $gf' = gfu$ and $h' = hu$. From the universal property of the right pullback, it follows that $f' = fu$, so the left square is a pullback. \square

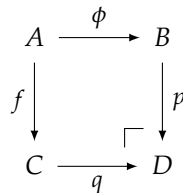
Lemma 4.3.8. *For a commutative cube, where the front, right, and back faces are pullbacks:*



the left face is also a pullback.

Proof. Since the back and right faces are both pullback squares, we can apply Lemma 4.3.7 to show that the back-right two faces form a larger pullback square. Then by commutativity of the cube, the square formed by the left-front two faces is also a pullback square. Applying Lemma 4.3.7 from right to left concludes that the left face is a pullback. \square

Lemma 4.3.9. *Isomorphisms are stable under pushout. For the following pushout, if ϕ is an isomorphism, then so too is q .*



Proof. Let $f\phi^{-1}$ and 1_C form a cocone to C over the pushout. It follows straightforwardly that the induced map $q' : D \rightarrow C$ is the inverse of q . \square

We are now ready to prove the uniqueness theorem for pushout complements.

Theorem 4.3.10. [44]. *If a pair of arrows (m, g) , where m is mono, has a pushout complement, it is unique up to isomorphism. That is, for any two pushout complements, (f, n) and (f', n') , there exists an isomorphism ϕ making the following diagram commute:*



Proof. Suppose both of the following are pushout squares:

$$\begin{array}{ccc}
 A & \xrightarrow{m} & B \\
 f \downarrow & & \downarrow g \\
 C & \xrightarrow{n} & D
 \end{array}
 \qquad
 \begin{array}{ccc}
 A & \xrightarrow{m} & B \\
 f' \downarrow & & \downarrow g \\
 C' & \xrightarrow{n'} & D
 \end{array}$$

Following a similar strategy to Lemma 4.3.6, we use these squares to build a commutative cube:

$$\begin{array}{ccccc}
 C'' & \xrightarrow{k} & C' & & \\
 \downarrow l & \swarrow h & \downarrow f' & & \downarrow n' \\
 & & A & \xrightarrow{1} & A \\
 & & \downarrow 1 & & \downarrow m \\
 & & A & \xrightarrow{m} & B \\
 & \swarrow f & & & \searrow g \\
 C & \xrightarrow{n} & D & &
 \end{array}$$

where the first pushout square forms the bottom face, and the second pushout square forms the right face. The front face is the pullback of n and n' , and the back face is the pullback of m with itself. This pullback consists of identities because m is mono. Now, f and f' form a cone under the pullback of n and n' , so let h be the induced map.

The right face is a pushout along a monomorphism, so by Lemma 4.3.6, it is also a pullback. We can therefore conclude from Lemma 4.3.8 that the left face is also a pullback. From the VK-square property, we can then conclude that the top face is a pushout. By Lemma 4.3.9, k is an isomorphism. We can also form a similar cube, but with the positions of the two pushouts interchanged:

$$\begin{array}{ccccc}
 C'' & \xrightarrow{l} & C & & \\
 \downarrow k & \swarrow h & \downarrow f & & \downarrow n \\
 & & A & \xrightarrow{1} & A \\
 & & \downarrow 1 & & \downarrow m \\
 & & A & \xrightarrow{m} & B \\
 & \swarrow f' & & & \searrow g \\
 C' & \xrightarrow{n'} & D & &
 \end{array}$$

and conclude similarly that l is an isomorphism. By construction, all faces commute, and thus by letting $\phi := kl^{-1}$, we can read off the statement of the theorem from the commutative cube (with the relevant arrows shown in bold). \square

As in the case of graphs, a *rewrite rule* in an adhesive category \mathcal{A} is a span of monomorphisms.

$$L \rightarrowtail R := L \xleftarrow{b_1} I \xrightarrow{b_2} R$$

DPO rewriting consists of three steps.

1. Identify a matching $m : L \rightarrow G$.
2. Compute the pushout complement G' of L in G .
3. Push out G' and R to obtain a rewritten graph H .

First, we define *matchings*. Adhesive categories only guarantee the *uniqueness* of pushout complements, not the existence. In most categories, these will not exist for arbitrary pairs of morphisms. Thus, we build the existence condition into the definition of matching.

Definition 4.3.11. For a rewrite rule

$$L \rightarrow R := L \xleftarrow{b_1} I \xrightarrow{b_2} R$$

a *matching* m of $L \rightarrow R$ on G is a monomorphism $m : L \rightarrow G$ such that the following pushout complement exists:

$$\begin{array}{ccc} I & \xrightarrow{b_1} & L \\ \downarrow & & \downarrow m \\ G' & \longrightarrow & G \end{array} \quad \square$$

This existential condition is not particularly useful in determining which morphisms are matchings. Luckily, in specific adhesive categories, we can often do better. For graph-like categories, we can usually ensure the existence of a pushout complement using some version of the *no-dangling-edges condition*.

Definition 4.3.12. A monomorphism $m : L \rightarrow G$ of a rewrite rule $L \xleftarrow{b_1} I \xrightarrow{b_2} R$ in the category **Graph** is said to satisfy the *no-dangling-edges* condition if for any vertex $v \in V_L - V_I$, all edges incident to $m(v)$ must be in the image of m .

Theorem 4.3.13. In **Graph**, a monomorphism m is a matching if and only if it satisfies the *no-dangling-edges condition*.

Proof. Since $I \xrightarrow{b_1} L \xrightarrow{m} G$ are both monos, we will assume without loss of generality that $I \subseteq L \subseteq G$. For (\Leftarrow) , we form the pushout complement G' as a graph with vertices $V_G - (V_L - V_I)$ and edges $E_G - (E_L - E_I)$. By the no-dangling-edges condition, if $s(e) \in V_L - V_I$ or $t(e) \in V_L - V_I$ then $e \in E_L$. If e were to be in E_I then $s(e) \in V_I$ and $t(e) \in V_I$ so $e \in V_L - V_I$. Therefore, the maps s and t have well-defined restrictions to G' . Furthermore, $G' \cup L = G$ and $G' \cap L = I$, so G' is the pushout complement.

For (\Rightarrow) , suppose m is a matching. Then there exists a pushout square:

$$\begin{array}{ccc}
I & \xrightarrow{b_1} & L \\
m' \downarrow & & \downarrow m \\
G' & \xrightarrow{b'_1} & G
\end{array}$$

Since monos in an adhesive category are stable over pushout, b'_1 and m' are also monos. As such, we identify I with its image under m' regard G as $G' \cup L$. Let v be a vertex in $V_L - V_I$, and e an edge incident to v in G . Suppose e is not in L (i.e. the image of $m : L \rightarrow G$), then e must be in G' . Thus v must also be in G' , so $v \in V_{G'} \cap V_L = V_I$, which is a contradiction. We therefore conclude that $e \in E_L$. \square

We can now define the notion of a rewrite in an adhesive category.

Definition 4.3.14. For a rewrite rule $L \rightarrow R$ and a matching $m : L \rightarrow G$, the *rewrite* of G into H is a double pushout diagram:

$$\begin{array}{ccccc}
L & \longleftarrow & I & \longrightarrow & R \\
m \downarrow & & \downarrow & & \downarrow \\
G & \longleftarrow & G' & \longrightarrow & H
\end{array}$$

4.4 Partial Adhesive Categories

Adamek sums up a procedure by which many categories are defined in *Abstract and Concrete Categories* [4]:

Many familiar constructs of an “algebraic” or “topological” nature have natural descriptions that can be accomplished in two steps. The first step [...] consists of defining algebraic (resp. topological) categories by means of certain functors. The second step consists of singling out full, concrete subcategories by imposing certain axioms on the objects.

We shall follow this prescription to construct the category of string graphs in the next chapter. We already saw in Examples 4.3.3 how adhesivity is inherited by categories defined “by means of certain functors”: namely slice, coslice, and functor categories. There is no reason for a full subcategory of an adhesive category to also be adhesive. However, we can still characterise certain subcategories of an adhesive category that inherit “enough adhesiveness” to do rewriting.

Definition 4.4.1 (Partial Adhesive Category). \mathcal{C} is called a *partial adhesive category* if it is a full subcategory of an adhesive category \mathcal{A} and the embedding functor $S : \mathcal{C} \rightarrow \mathcal{A}$ preserves monomorphisms.

The category \mathcal{C} inherits the unique pushout complement property for a certain class of pushouts in \mathcal{C} , which we shall call S -pushouts.

Definition 4.4.2 (S -spans and S -pushouts). Let \mathcal{C} be a partial adhesive category and $S : \mathcal{C} \rightarrow \mathcal{A}$ the embedding functor. A span $A \xleftarrow{f} B \xrightarrow{g} C$ in \mathcal{C} is called an S -span if it has a pushout and that pushout is preserved by S . Such pushouts are called S -pushouts.

Recall that full and faithful functors reflect colimits. Since S reflects *all* pushouts, we could equivalently define S -spans in \mathcal{C} as those spans which have a pushout *reflected* by S .

Definition 4.4.3 (S -pushout complement). An S -pushout complement for a pair of arrows (b, f) is a pushout complement, where the following diagram is an S -pushout:

$$\begin{array}{ccc} I & \xrightarrow{b} & L \\ c \downarrow & & \downarrow f \\ G' & \longrightarrow & G \end{array}$$

We call b the *boundary* of L and c the *coboundary* of L in G .

Lemma 4.4.4. *If a pair of arrows (b, f) , where b is mono, have an S -pushout complement, it is unique up to isomorphism.*

Proof. Let (c, g) and (c', g') be S -pushout complements. Then the following diagrams are pushouts in the adhesive category \mathcal{A} :

$$\begin{array}{ccc} SB & \xrightarrow{Sb} & SL \\ Sc \downarrow & & \downarrow Sf \\ SG' & \xrightarrow{Sg} & SG \end{array} \quad \begin{array}{ccc} SB & \xrightarrow{Sb} & SL \\ Sc' \downarrow & & \downarrow Sf \\ SG'' & \xrightarrow{Sg'} & SG \end{array}$$

Since S preserves monos, these are both pushout complements of (Sb, Sf) for Sb mono. So the following diagram commutes in \mathcal{A} , for ϕ' an isomorphism:

$$\begin{array}{ccc} SB & \xrightarrow{Sc} & SG' \\ Sc' \downarrow & \nearrow \phi' & \downarrow Sg \\ SG'' & \xrightarrow{Sg'} & SG \end{array}$$

Since S is full and faithful, there exists an isomorphism $\phi : G' \rightarrow G''$ such that $S\phi = \phi'$. Replacing ϕ' in the above diagram yields:

$$\begin{array}{ccc}
SB & \xrightarrow{Sc} & SG' \\
Sc' \downarrow & \swarrow S\phi & \downarrow Sg \\
SG'' & \xrightarrow{Sg'} & SG
\end{array}$$

Diagram (4.6) commutes by the faithfulness of S . □

Definition 4.4.5 (S -matching). For a rewrite rule $L \rightarrow R$, a monomorphism $m : L \rightarrow G$ is called an S -matching if $B \xrightarrow{b_1} L \xrightarrow{m} G$ has an S -pushout complement.

Clearly if m is an S -matching, then Sm is a matching. For the converse to be true, it suffices for the image of S to be closed under subobjects.

Definition 4.4.6 (S -rewrite). Let $L \rightarrow R := L \xleftarrow{b_1} I \xrightarrow{b_2} R$ be a rewrite rule and $m : L \rightarrow G$ be an S -matching. Then for G' the S -pushout complement of $B \xrightarrow{b_1} L \xrightarrow{m} G$, if the right pushout square in the following diagram exists and is an S -pushout:

$$\begin{array}{ccccc}
L & \xleftarrow{b_1} & B & \xrightarrow{b_2} & R \\
m \downarrow & & \downarrow & & \downarrow \\
G & \xleftarrow{\quad} & G' & \xrightarrow{\quad} & H
\end{array}$$

Then H is the S -rewrite of $L \rightarrow R$ at m .

We write H defined as in 4.4.6 as $G[L \rightarrow R]_m$, or more explicitly $G[L \xleftarrow{b_1} I \xrightarrow{b_2} R]_m$.

4.4.1 Example: The Category of Simple Graphs

Partial adhesive categories should be thought of as adhesive categories, with some extra axioms imposed on the objects. In the presence of these axioms, one may need to verify by hand the relevant S -pushouts and S -pushout complements exist for a particular class of rewrite rules or matchings. In practice, this tends to be fairly straightforward. In this section, we give the derivation of these properties for the category of simple graphs.

Let \mathbf{Gr} be the category of simple graphs, i.e. graphs where every pair of vertices is connected by at most one edge in either direction. Equivalently, a simple graph is just a binary relation from a set to itself. An object in \mathbf{Gr} consists of a set V of vertices, E of edges and an injection $e : E \hookrightarrow V \times V$. A simple graph homomorphism is a pair f_V, f_E such that

$$\begin{array}{ccc}
E_G & \xrightarrow{e} & V_G \times V_G \\
f_E \downarrow & & \downarrow f_V \times f_V \\
E_H & \xrightarrow{e'} & V_H \times V_H
\end{array}$$

There is an evident embedding of **Gr** into **Graph**:

$$S : (V_G, E_G, e : E \rightarrow V) \mapsto (V_G, E_G, \pi_1 \circ e, \pi_2 \circ e)$$

Under the identifications $s = \pi_1 \circ e$, $t = \pi_2 \circ e$, the notions of graph homomorphism in **Gr** and **Graph** are equivalent, so S is a full subcategory embedding. **Gr** is a reflective subcategory of **Graph**, so S has a left adjoint. As a right adjoint, S preserves limits and, in particular, monomorphisms. Therefore **Gr** is a partial adhesive category.

Lemma 4.4.7. *Let $A \xleftarrow{m} B \xrightarrow{n} C$ be a span in **Gr**, where m is a mono and n is a regular mono. Then m, n has an S -pushout.*

Proof. Since S is full and faithful, it suffices to show for m a mono, n a regular mono, that D defined by the following pushout in **Graph** is a simple graph.

$$\begin{array}{ccc} SA & \xrightarrow{Sm} & SB \\ Sn \downarrow & & \downarrow \\ SC & \longrightarrow & D \end{array} \quad \lrcorner$$

We can consider D to be a union of a simple graph SB and another simple graph SC . Regular monos in **Gr** are precisely the full subgraph embeddings, so $SA \cong SB \cap SC$ is a full subgraph of SC . Consider two vertices v, v' in D and edges e, e' such that $s(e) = s(e') = v$ and $t(e) = t(e') = v'$. The only way these edges can possibly be distinct is if e is in SB and e' is in SC . Thus v and v' are in $SB \cap SC$. Since the intersection is a full subgraph of SC , e' is in $SB \cap SC$, so $e = e'$. \square

We can an S -matching that is a regular a monomorphism a *regular S -matching*. We can show that when m is a regular S -matching in **Gr** then the associated S -rewrite exists and is unique.

Theorem 4.4.8. *For a rewrite rule $L \rightarrow R := L \xleftarrow{b_1} I \xrightarrow{b_2} R$ and a regular S -matching $m : L \rightarrow G$, the associated S -rewrite is well-defined. That is, the following two S -pushout squares exist:*

$$\begin{array}{ccccc} L & \xleftarrow{b_1} & B & \xrightarrow{b_2} & R \\ m \downarrow & & \downarrow n & & \downarrow \\ G & \longleftarrow & G' & \longrightarrow & H \end{array} \quad \lrcorner \quad (4.7)$$

Proof. The existence of the left S -pushout square follows from the fact that any subgraph of a simple graph is also a simple graph. Since the following is a pushout along a monomorphism in **Graph**, it is also a pullback, by Lemma 4.3.6.

$$\begin{array}{ccc}
SI & \xrightarrow{Sb_1} & SL \\
Sn \downarrow & & \downarrow Sm \\
SG' & \xrightarrow{Sf} & SG
\end{array}$$

Since S is full and faithful, it reflects pullbacks. So the following is a pullback in \mathbf{Gr} .

$$\begin{array}{ccc}
I & \xrightarrow{b_1} & L \\
n \downarrow & \lrcorner & \downarrow m \\
G' & \xrightarrow{f} & G
\end{array}$$

Regular monos are stable under pullback, so n is a regular mono. By Lemma 4.4.7, the right square in (4.7) is an S -pushout. \square

Thus, if we restrict to regular monomorphisms for matchings, DPO rewriting in the partial adhesive category \mathbf{Gr} is well-defined, and the procedure is *identical* to that in the category \mathbf{Graph} .

4.4.2 Commutation of S -Pushouts and S -Pushout Complements

Often we wish to glue objects together using S -pushouts. If we perform an S -rewrite that is confined to a single component of this glued-together object, it should not matter if we first apply the rewrite then compose or if we first compose then apply the rewrite. In later sections, we shall define categories whose morphisms consist of graphs modulo a rewrite system. In such categories, this commutation of gluing and rewriting is crucial to ensuring that the composition operation well-defined. Therefore, we now prove two lemmas regarding the compatibility of S -pushouts, S -pushout complements, and S -rewrites.

Lemma 4.4.9. *S -pushout complements commute with S -pushouts. Let the following diagram be an S -pushout:*

$$\begin{array}{ccc}
P & \xrightarrow{q} & H \\
p \downarrow & & \downarrow i_2 \\
G & \xrightarrow{i_1} & G +_{p,q} H
\end{array}$$

Assume $B \xrightarrow{b} K \xrightarrow{m} G$ and $B \xrightarrow{b} K \xrightarrow{i_1 m} G +_{p,q} H$ both have pushout complements,

$$\begin{array}{ccc}
B & \xrightarrow{b} & K \\
c \downarrow & & \downarrow m \\
G & \xrightarrow{s} & G
\end{array}
\qquad
\begin{array}{ccc}
B & \xrightarrow{b} & K \\
c' \downarrow & & \downarrow i_1 m \\
(G +_{p,q} H) & \xrightarrow{s'} & G +_{p,q} H
\end{array}$$

and the S -span (p, q) factors through $G -_{b,m} K$, i.e. there exists p' such that $sp' = p$ and (p', q) is an S -span.

Then, for a second S -pushout:

$$\begin{array}{ccc} P & \xrightarrow{q} & H \\ p' \downarrow & & \downarrow j_2 \\ G & \xrightarrow{j_1} & (G -_{b,m} K) +_{p',q} H \end{array}$$

there is an isomorphism $\phi : (G +_{p,q} H) -_{b,i_1} K \xrightarrow{\sim} (G -_{b,m} K) +_{p',q} H$, commuting with the coboundaries c and c' of K in G and $G +_{p,q} H$ respectively.

$$\begin{array}{ccc} B & \xrightarrow{c'} & (G +_{p,q} H) -_{b,i_1} K \\ c \downarrow & & \downarrow \phi \\ G -_{b,m} K & \xrightarrow{j_1} & (G -_{b,m} K) +_{p',q} H \end{array} \quad (4.8)$$

Proof. The proof follows from the associativity of pushouts and the uniqueness of pushout complements. First, note that, in the following diagram, [1] commutes and is a pushout because $sp' = p$:

$$\begin{array}{ccccc} & & P & \xrightarrow{q} & H \\ & & \downarrow p' & & \downarrow i_2 \\ B & \xrightarrow{c} & G -_{b,m} K & [1] & \\ \downarrow b & & \downarrow s & & \downarrow i_2 \\ K & \xrightarrow{m} & G & \xrightarrow{i_1} & G +_{p,q} H \end{array}$$

Next, we make the two pushouts in the opposite order.

$$\begin{array}{ccccc} & & P & \xrightarrow{q} & H \\ & & \downarrow p' & & \downarrow j_2 \\ B & \xrightarrow{c} & G -_{b,m} K & \xrightarrow{j_1} & (G -_{b,m} K) +_{p',q} H \\ \downarrow b & & [2] & & \downarrow k_2 \\ K & \xrightarrow{m} & G & \xrightarrow{i_1} & G +_{p,q} H \\ & & & & \downarrow k_1 \\ & & & & Q \end{array}$$

By associativity of pushouts, there exists an isomorphism ψ such that:

$$\begin{array}{ccc}
G +_{p,q} H & \xleftarrow{i_2} & H \\
i_1 m \uparrow & \searrow \psi & \downarrow k_2 j_2 \\
K & \xrightarrow{k_1} & Q
\end{array}$$

Since Q is only defined up to isomorphism, we are free to take $\psi = 1_{G +_{p,q} H}$, in which case $k_1 = i_1 m$. Then, square [2] from above becomes:

$$\begin{array}{ccc}
B & \xrightarrow{j_1 c} & (G -_{b,m} K) +_{p',q} H \\
b \downarrow & & \downarrow k_2 \\
K & \xrightarrow{i_1 m} & G +_{p,q} H
\end{array}$$

Compare this to the definition of $(G +_{p,q} H) -_{b,i_1 m} K$ as a pushout complement:

$$\begin{array}{ccc}
B & \xrightarrow{c'} & (G +_{p,q} H) -_{b,i_1 m} K \\
b \downarrow & & \downarrow s' \\
K & \xrightarrow{i_1 m} & G +_{p,q} H
\end{array}$$

The result then follows from uniqueness of pushout complements. \square

Theorem 4.4.10. *S-adhesive rewrites commute with S-adhesive pushouts. Let $m : L \rightarrow G$ be an S-matching of $L \rightarrow R := L \xleftarrow{b_1} I \xrightarrow{b_2} R$. The rewrite is computed as the double pushout:*

$$\begin{array}{ccccc}
L & \xleftarrow{b_1} & B & \xrightarrow{b_2} & R \\
m \downarrow & & \downarrow c & & \downarrow m' \\
G & \xleftarrow{s} & G -_{b,m} L & \xrightarrow{s'} & G[L \rightarrow R]_m
\end{array}$$

Let (p, q) , (\hat{p}, q) and (p', q) be three S-spans, such that:

$$\begin{array}{ccc}
G & \xleftarrow{p} & P \\
s \uparrow & & \downarrow q \\
G -_{b_1, m} L & \xleftarrow{\hat{p}} & P & \xrightarrow{q} & H \\
s' \downarrow & & \downarrow p' & & \\
G[L \rightarrow R]_m & & & &
\end{array} \tag{4.9}$$

Then, for the pushout injection $i_1 : G \rightarrow G +_{p,q} H$, if $i_1 m$ is mono, the following is an isomorphism:

$$(G[L \rightarrow R]_m) +_{p',q} H \cong (G +_{p,q} H)[L \rightarrow R]_{i_1 m}$$

Proof. $G[L \rightarrow R]_m$ is the pushout of R and $G -_{b_1, m} L$ along B , so by uniqueness of pushout complements, we can choose $(G -_{b_1, m} L)$ to be equal to $((G[L \rightarrow R]_m) -_{b_2, m'} R)$, for the same coboundary c . Then, by two applications of Lemma 4.4.9, we can choose $(G -_{b_1, m} L) +_{\hat{p}, q} H = ((G[L \rightarrow R]_m) -_{b_2, m'} R) +_{\hat{p}, q} H$ as the pushout complement of both of the following squares.

$$\begin{array}{ccccc}
 L & \longleftarrow & B & \longrightarrow & R \\
 i_1 m \downarrow & & \downarrow c' & & \downarrow \\
 G +_{p, q} H & \longleftarrow & (G -_{b_1, m} L) +_{\hat{p}, q} H & \longrightarrow & G[L \rightarrow R] +_{p', q} H
 \end{array}$$

Note that c' becomes the coboundary for both squares because diagram (4.8) commutes. This is then exactly the computation of the rewrite $(G +_{p, q} H)[L \rightarrow R]_{i_1 m}$. The theorem holds because S-rewrites are unique up to isomorphism. \square

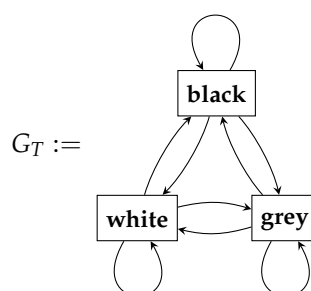
We shall use these two theorems throughout this dissertation to show that rewriting is compatible with several notions of composing graphs.

Chapter 5

String Graphs and Monoidal Theories

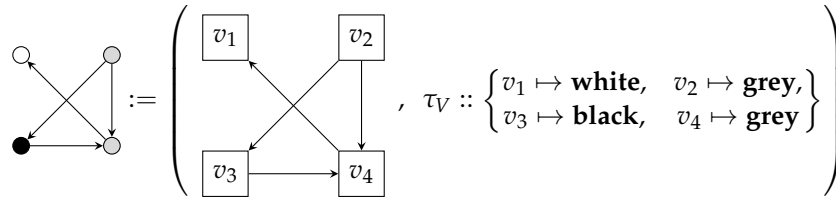
In this chapter, we come to one of the primary contributions of this dissertation: the formalisation of the diagrammatic language of monoidal categories using certain typed graphs called *string graphs*. In the previous chapter, we introduced the notion of a partial adhesive category. Our primary reason for doing so was to enable double-pushout graph rewriting in the category of string graphs.

Before passing to string graphs, we will look at the category of typed graphs, defined as a slice category \mathbf{Graph}/G_T . Objects of the slice category are pairs $(G, \tau_G : G \rightarrow G_T)$. These should be thought of as a graph G along with a map τ_G giving a *type* in G_T to every vertex and edge in G . Morphisms are simply graph homomorphisms respecting this type map: $f : G \rightarrow H$ such that $\tau_H \circ f = \tau_G$. It may seem odd at first that the type map is a graph homomorphism rather than simply a pair of functions defined on vertices and edges. However, the first can encode the latter, so these “unrestricted” vertex and edge typings are merely a special case of homomorphic graph typings. As an example, suppose we fix a set X of vertex types and have only one edge type. Then G_T can be defined as the connected graph whose vertices are the elements of X . For instance, if we let $X = \{\mathbf{black}, \mathbf{white}, \mathbf{grey}\}$, we can form a connected graph:

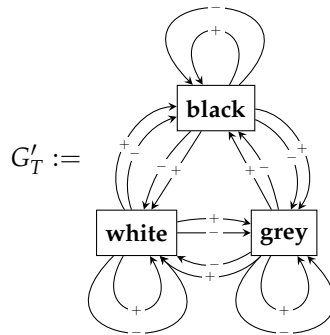


Since G_T is isomorphic to the totally connected directed graph with 3 vertices, any function $\tau_V : V_G \rightarrow V_{G_T}$ extends uniquely to a graph homomorphism $\tau : G \rightarrow G_T$. We can think of the fibres of τ_V (i.e. the inverse images $\tau_V^{-1}(\mathbf{black})$, $\tau_V^{-1}(\mathbf{white})$, and $\tau_V^{-1}(\mathbf{grey})$) as sets of vertices in G that are coloured “black”, “white”, or “grey” respectively. We can represent a graph with coloured

vertices as a pair consisting of a graph G and a typing function τ .



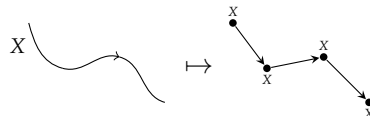
Typographs can express edge typing as well. For instance, to add a set of edge types $Y = \{+, -\}$, we can simply add a copy of Y for every pair of vertices (v_1, v_2) in G_T connecting v_1 to v_2 .



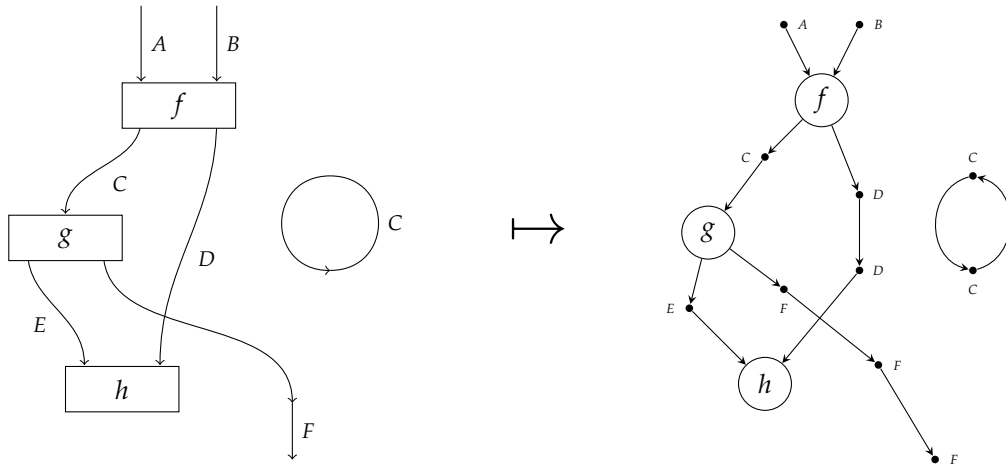
One can even restrict which vertex types can be connected by which edge types by taking subgraphs of G'_T . In the coming sections, we will use this restriction to make sure that our diagrams are well-typed, i.e. composition of “boxes” in the language of string diagrams should respect the types on “wires”.

5.1 String Graphs

String diagrams consist of *boxes*, which represent morphisms in a monoidal category, and *wires*, which are used to connect boxes together. We turn string diagrams into a *string graphs*, which are typed graphs with two distinguished kinds of vertex: *box-vertices* and *wire-vertices*. As the name suggests, *box-vertices* represent the boxes (i.e. morphisms/generators) in the diagrammatic language. These represent the “logical” or “semantic” vertices of the string graph. An important characteristic of wires, which distinguishes them from normal edges in a graph, is that they are not required to have boxes at both ends and they can be connected to themselves to form *circles*. For that reason, we will represent wires as chains of special “place-holder” vertices called *wire-vertices*.

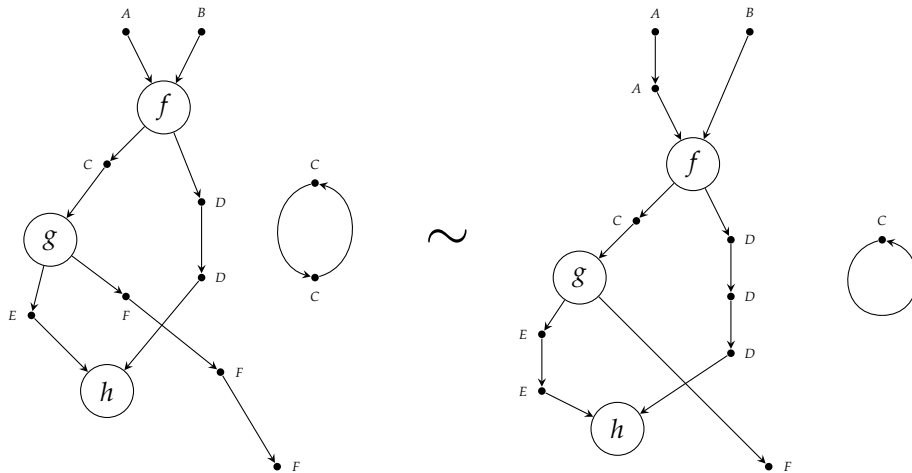


Note how the wire-vertices carry the type of the wire. Representing boxes as *box-vertices*, it is possible to translate any string diagram into a string graph.



Edge types (not shown) are used to keep track of the ordering of inputs and outputs to the boxes (see Definition 5.1.1).

The number of wire-vertices making up a wire is irrelevant, so for the purposes of representing a string diagram, the following two string graphs are equivalent:



If we were to treat the two graphs above as 1D simplicial complexes that define topological graphs, then the geometric realisation of the wires on the left are homeomorphic to those of the wires on the right. For that reason, the two graphs above are called *wire-homeomorphic*. We can formalise the notion of wire-homeomorphism as a confluent, terminating graph rewrite system and prove that string graphs, up to wire-homeomorphism can be used to construct free monoidal categories. This construction is essentially the graph version of the topological construction described in Section 2.3.2.

For a monoidal signature T , we can define a category \mathbf{SGraph}_T of string graphs with generators taken from T . We do this by turning the monoidal signature T into a typegraph G_T and defining \mathbf{SGraph}_T is a full subcategory of \mathbf{Graph}/G_T . We shall then prove that the embedding of \mathbf{SGraph}_T into \mathbf{Graph}/T preserves monos, so \mathbf{SGraph}_T is a partial adhesive category.

First, we show the construction of G_T . Recall that a monoidal signature $T = (O, M, \text{dom}, \text{cod})$ consists of sets O, M and functions $\text{dom} : M \rightarrow w(O), \text{cod} : M \rightarrow w(O)$ into the set $w(O)$ of lists of elements of O .

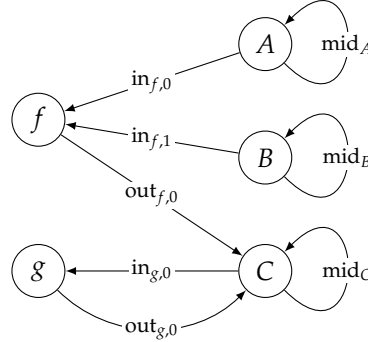
Definition 5.1.1. For a monoidal signature T , the *derived typegraph* G_T of T has vertices $O + M$ and the following edges:

- a self-loop mid_X for every $X \in O$,
- an edge $\text{in}_{f,i}$ for $f \in M, 0 \leq i < \text{Length}(\text{dom}(f))$, connecting $\text{dom}(f)[i]$ to f , and
- an edge $\text{out}_{f,j}$ for $f \in M, 0 \leq j < \text{Length}(\text{cod}(f))$, connecting f to $\text{cod}(f)[j]$.

Example 5.1.2. Let T be the following monoidal signature:

$$T := \left\{ \begin{array}{c} A \quad B \\ \downarrow \quad \downarrow \\ \boxed{f} \\ \downarrow \\ C \end{array} , \begin{array}{c} C \\ \downarrow \\ \boxed{g} \\ \downarrow \\ C \end{array} \right\}$$

Then, the derived typegraph G_T is:



Definition 5.1.3. Let $(G, \tau : G \rightarrow G_T)$ be an object in \mathbf{Graph}/G_T , for a monoidal signature T . By definition of G_T , the vertices of G_T are $O + M$. A vertex $v \in V_G$ is called a *box-vertex* of $\tau_V(v) \in M$. It is called a *wire-vertex* if $\tau_V(v) \in O$. Let $B(G) \subseteq V_G$ be the set of all box-vertices, and $W(G)$ be the set of wire-vertices.

Note that since we have omitted self-loops on box-vertices in G_T , any path between two box-vertices in a G_T -typed graph must pass *through* a wire-vertex. This is important to the definition of string graphs, as the “object” types from the signature T are carried by wire-vertices.

There are two restrictions that we place on typed graphs (G, τ) to make string graphs. The first is that wires should not split or merge. Namely, any wire-vertex in G should have at most one in-edge and one out-edge. In other words, for a graph $G = (V_G, E_G, s, t)$, the restrictions of s and t to the wire vertices $W(G)$ are both monomorphisms, i.e. s', t' defined by the pullbacks below are monomorphisms:

$$\begin{array}{ccccc}
E' & \xrightarrow{s'} & W(G) & \xleftarrow{t'} & E'' \\
\downarrow & \lrcorner & \downarrow & \lrcorner & \downarrow \\
E_G & \xrightarrow{s} & V_G & \xleftarrow{t} & E_G
\end{array}$$

The second condition is that the incident edges of a box-vertex b in G should match those of its type, $\tau_V(b)$. In other words, there should be the same number of inputs and outputs to b as there are to $\tau_V(b)$. We formalise this condition by introducing the notion of a *local isomorphism*.

Definition 5.1.4. For a vertex $v \in V_G$, the *edge neighbourhood* of v is the set of edges $N(v) := s^{-1}(v) \cup t^{-1}(v)$.

Fix a graph homomorphism $f : G \rightarrow H$. Then for a vertex v in G and an adjacent edge e , the edge $f(e)$ is adjacent to $f(v)$. Thus $f_E(N(v)) \subseteq N(f(v))$. Let $f^v : N(v) \rightarrow N(f(v))$ be the function defined by $f^v(e) = f_E(e)$, for $e \in N(v)$.

Definition 5.1.5. For G_T -typed graphs $(G, \tau_G), (H, \tau_H)$, a typed graph homomorphism $f : G \rightarrow H$ is called a *local isomorphism* if for every box $b \in B(G)$, $f^b : N(b) \rightarrow N(f(b))$ is a bijection.

In particular for a G_T -typed graph (G, τ) , the typing map τ can be considered as a typed graph homomorphism from (G, τ) to $(G_T, 1_{G_T})$. Thus, we can require that it be a local isomorphism.

Definition 5.1.6. A G_T -typed graph (G, τ) is called a *string graph* if τ is a local isomorphism and every wire-vertex in G has at most one in-edge and one out-edge. The category \mathbf{SGraph}_T is the full subcategory of \mathbf{Graph}/T whose objects are string graphs.

Since the typing maps τ in \mathbf{SGraph}_T are local isomorphisms, we can show that *every* arrow in \mathbf{SGraph}_T is a local isomorphism.

Proposition 5.1.7. *Every arrow in \mathbf{SGraph}_T is a local isomorphism.*

Proof. Let $(G, \tau_G), (H, \tau_H)$ be G_T -typed graphs. By definition τ_G and τ_H are both local isomorphisms. For any $f : (G, \tau_G) \rightarrow (H, \tau_H)$ in \mathbf{Graph}/G_T , the following diagram commutes:

$$\begin{array}{ccc}
G & \xrightarrow{\tau_G} & G_T \\
f \downarrow & \nearrow \tau_H & \\
H & &
\end{array}$$

Thus, for any v in G we get this triangle in **Set**:

$$\begin{array}{ccc}
N(v) & \xrightarrow{\tau_G^v} & N(\tau_G(v)) \\
f^v \downarrow & \nearrow \tau_H^{f(v)} & \\
N(f(v)) & &
\end{array}$$

Since τ_G^v and $\tau_H^{f(v)}$ are both bijections, f^v is a bijection, so f is a local isomorphism. \square

Proposition 5.1.8. *A morphism in \mathbf{SGraph}_T is a monomorphism iff it is injective.*

Proof. Suppose $m : G \rightarrow H$ in \mathbf{SGraph}_T is not injective. m is a local isomorphism, so if m takes two distinct edges e_1 and e_2 to the same edge, then the adjacent vertices of e_1 and e_2 must also be distinct.

To show that m cannot be mono, we will define a string graph K and distinct maps $f, g : K \rightarrow G$ such that $mf = mg$. If m takes two distinct box-vertices v_1, v_2 in G to a single box-vertex in H , then let K be the subgraph of G consisting of just v_1 and its neighbourhood. If m takes two distinct wire-vertices to a single wire-vertex in H , then let K be a T -string graph consisting of a single, isolated wire-vertex. In either case, there are two distinct maps f, g such that $mf = mg$. \square

Corollary 5.1.9. *\mathbf{SGraph}_T is a partial adhesive category.*

5.1.1 S-pushouts of String Graphs

For the constructions to come, it is useful to characterise the S -pushouts in \mathbf{SGraph}_T . To simplify matters, it suffices to characterise the S -pushouts of spans $G \xleftarrow{m} K \xrightarrow{n} H$ where m and n are both monomorphisms. We show that a span of monos is an S -span if and only if it is *boundary-coherent*. Before we define boundary-coherence, we need the notion of a boundary.

Definitions 5.1.10. If a wire-vertex has no in-edges, it is called an *input*. We write the set of inputs of a string graph G as $\text{In}(G)$. Similarly, a wire-vertex with no out-edges is called an *output*, and the set of outputs is written $\text{Out}(G)$. The inputs and outputs define a string graph's *boundary*, written $\text{Bound}(G)$. If a boundary vertex has no in-edges and no out-edges, (it is both an input and output) it is called an *isolated wire-vertex*. A string graph consisting of only isolated wire-vertices is called a *point graph*.

By abuse of notation, we may treat $\text{In}(G)$, $\text{Out}(G)$, and $\text{Bound}(G)$ as sets or as point graphs. The intended usage will be clear from context.

Definition 5.1.11. A pair of morphisms $f : K \rightarrow G$, $g : K \rightarrow H$ in \mathbf{SGraph}_T is called *boundary-coherent* if:

1. for all $v \in \text{In}(K)$ at least one of $f(v)$ and $g(v)$ is an input, and

2. for all $v \in \text{Out}(K)$ at least one of $f(v)$ and $g(v)$ is an output.

Theorem 5.1.12. *A span of monomorphisms $G \xleftarrow{m} I \xrightarrow{n} H$ in \mathbf{SGraph}_T has an S -pushout if and only if m and n are boundary-coherent.*

Proof. For (\Leftarrow) , it suffices to show that K is a string graph, for the following pushout in \mathbf{Graph}/G_T :

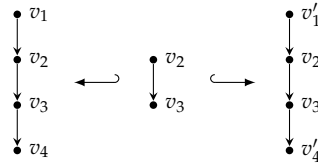
$$\begin{array}{ccc}
 SI & \xrightarrow{Sm} & SG \\
 Sn \downarrow & & \downarrow g \\
 SH & \xrightarrow{f} & K
 \end{array} \quad (5.1)$$

Since K is isomorphic to the union of SG and SH , the images of f and g cover K . Let $f(b) \in B(K)$ be a box-vertex in the image of f but not g . Then the neighbourhood of b is identical to that of $f(b)$, so the inclusion of edges $f^b : N(b) \rightarrow N(f(b))$ is a bijection. Similarly, for $g(b) \in B(K)$ not in the image of f , $g^b : N(b) \rightarrow N(g(b))$ is a bijection. Finally, pick a box-vertex $b \in B(I)$, then since m and n are local isomorphisms, Sm and Sn restrict to bijections on $N(b) \subseteq E_I$. Thus, all of the edges in $N(Sm(b))$ are identified with edges in $N(Sn(b))$ in K , and the inclusions f and g restrict to bijections on $N(Sm(b))$ and $N(Sn(b))$ respectively. So f and g are local isomorphisms. Since the images of f and g cover K , it follows that the typing map τ_K is a local isomorphism (cf. the proof of Proposition 5.1.7).

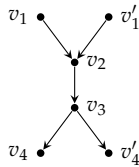
Now, suppose a wire-vertex $v \in W(K)$ has out-edges e_1, e_2 . The only way these can possibly be distinct is if v is in the image of both f and g . Then, there must be a v' such that $f \circ Sn(v') = g \circ Sm(v') = v$. If v' is not an output in I , then the out-edges of $m(v') \in B(G)$ and $n(v') \in B(H)$ must be in the images of m and n , respectively, so $e_1 = e_2$. If it is an output, then at least one of $m(v'), n(v')$ must be an output, so v has at most one out-edge. We can show similarly that v must have at most one in-edge. So K is a string graph.

For (\Rightarrow) , suppose the span $G \xleftarrow{m} I \xrightarrow{n} H$ is not boundary-coherent. If the pushout K given by (5.1) is not a string graph, then either (a) the span m, n does not have a pushout or (b) m, n does have a pushout, but it is not preserved by S . In either case, the span does not have an S -pushout, so it suffices to show that K is not a string graph. If the span m, n is not boundary-coherent, there exists wire-vertex v in I such that either (a) v is an input and $f(v)$ and $g(v)$ both have in-edges or (b) v is an output and $f(v)$ and $g(v)$ both have out-edges. If (a) is true, then the image of v will have two distinct in-edges in K . If (b) is true, it will have two distinct out-edges. In either case, K is not a string graph. \square

Example 5.1.13. Consider the following span of string graphs, which is not boundary-coherent:



If we push out this span in \mathbf{Graph}/G_T , we get:



which is clearly not a string graph. Therefore this span does not have an S -pushout.

Note that the empty graph is the initial object in \mathbf{SGraph}_T , so we have an easy corollary.

Corollary 5.1.14. \mathbf{SGraph}_T has finite coproducts and S preserves them.

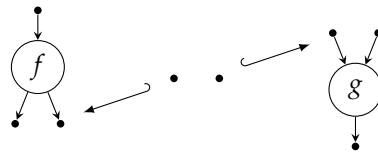
Proof. The proof follows from the observation that for any string graphs G, H , the span of initial arrows $G \xleftarrow{!} \emptyset \xrightarrow{!} H$ is trivially boundary-coherent. \square

One particularly important type of boundary-coherent span is a *plugging*. These are used to “plug” the inputs of one graph into the outputs of another graph.

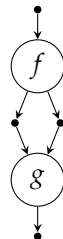
Definition 5.1.15. A boundary-coherent span $G \xleftarrow{m} P \xrightarrow{n} H$ where P is a point graph and G and H contain no isolated vertices is called a *plugging*.

Recall that wire-vertices in a point graph are both inputs and outputs. Thus boundary-coherence forces the images $m(p)$ and $n(p)$ of a wire-vertex $p \in P$ to have opposite polarities. That is to say, exactly one of $m(p), n(p)$ is an input and exactly one is an output.

Example 5.1.16. The following S -span defines a plugging:



Pushing out the span yields a string graph with the two smaller graphs plugged together:

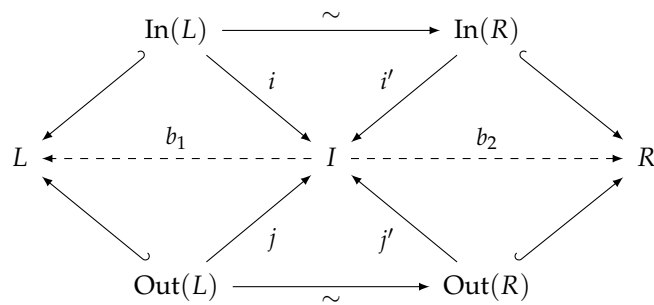


5.2 Rewriting with String Graphs

If a string graph contains no isolated points, then the set of inputs and outputs is disjoint, i.e. $\text{Bound}(G) \cong \text{In}(G) + \text{Out}(G)$. We will define string graph rewrite rules as pairs of string graphs with no isolated wire-vertices whose inputs and outputs are in bijection. Such a pair uniquely induces a span $L \xleftarrow{b_1} I \xrightarrow{b_2} R$ for $I \cong \text{In}(L) + \text{Out}(L) \cong \text{In}(R) + \text{Out}(R)$.

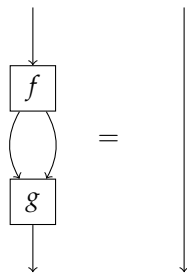
Definition 5.2.1. A string graph rewrite rule $L \rightarrow R$ is a span $L \xleftarrow{b_1} I \xrightarrow{b_2} R$ where:

1. L and R contain no isolated wire-vertices,
2. $\text{In}(L) \cong \text{In}(R)$, $\text{Out}(L) \cong \text{Out}(R)$,
3. $I \cong \text{In}(L) + \text{Out}(L) \cong \text{In}(R) + \text{Out}(R)$, and
4. the following diagram commutes for b_1 and b_2 the induced maps of the coproduct inclusions i, j and i', j' respectively:

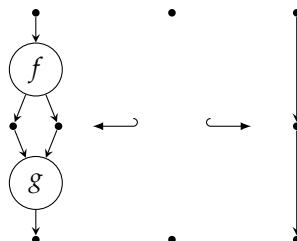


Since L contains no isolated wire-vertices, the images of $\text{In}(L)$ and $\text{Out}(L)$ are disjoint, so b_1 is injective. For the same reason, b_2 is also injective.

Example 5.2.2. Let $f : A \rightarrow A \otimes A$ and $g : A \otimes A \rightarrow A$ be morphisms in a monoidal category such that $f \circ g = 1_A$. This can be expressed as an equation between string diagrams:



...or as a rewrite rule between string graphs:



Theorem 5.2.3. Let $L \rightarrow R := L \xleftarrow{b_1} I \xrightarrow{b_2} R$ be a string graph rewrite rule. Any monomorphism $m : L \rightarrow G$ is an S -matching.

Proof. Since m is a local isomorphism and b_1 covers the boundary of L , m satisfies the no-dangling-edges condition, so it has a pushout complement G' in \mathbf{Graph}/G_T . For m to be an S -matching, it suffices to show that G' is a string graph. I consists only of wire-vertices, so by the no-dangling-edges condition, the adjacent edges of the box-vertices in G' are the same as they were in G , so the typing map $\tau_{G'}$ is still a local isomorphism. The fact that G' is a string graph then follows from it being a subgraph of G . \square

5.2.1 Wires and Wire-Homeomorphism

String graphs are meant to be the discrete version of (topological) string diagrams. In string diagrams, wires can be thought of as copies of the unit interval $[0, 1] \subset \mathbb{R}$, considered as an oriented manifold. Boxes are distinguished points, to which we ascribe semantic meaning. Clearly if we replace a wire in a string diagram with a homeomorphic wire, we get the same string diagram. In other words, the meaning of a string diagram is unaffected by shortening or lengthening wires.

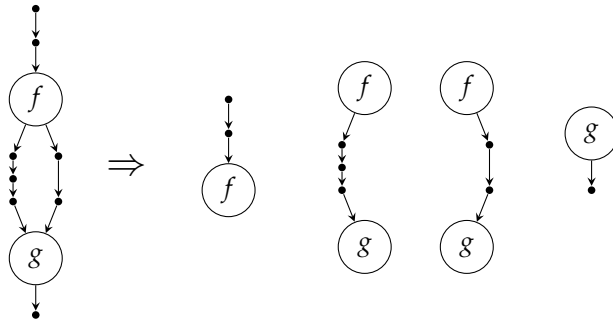
A *simple chain* is a connected, acyclic graph where each vertex has at most one in-edge and one out-edge. A vertex in a chain with only an in-edge or only an out-edge is called an *endpoint*. A *simple cycle* is connected graph where each vertex has exactly one in-edge and one out-edge.

Definition 5.2.4. For a string graph G , a *closed wire* $\widehat{W} \subseteq G$ is a simple cycle of wire-points or a simple chain such that

1. the endpoints of \widehat{W} are either box-vertices or in the boundary of G , and
2. all other vertices in \widehat{W} are wire-vertices.

It is worth noting that wires are not necessarily string graphs, as their typing function need not be a local isomorphism at the endpoints.

Example 5.2.5. A string graph and its 4 closed wires:



Every wire-vertex in a wire will be of a single type in G_T . This is called the *wire type*.

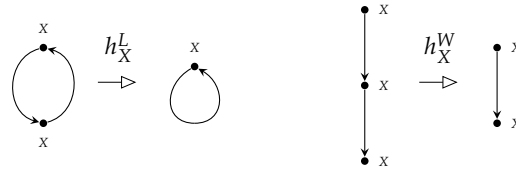
Definition 5.2.6. Two wires \widehat{W} and \widehat{W}' are said to be *homeomorphic* if they have the same wire type and (a) they are both simple cycles, or (b) they are simple chains where the endpoints and edges adjacent to the endpoints are of the same type.

Informally, we justify this terminology by noting that, if we consider \widehat{W} and \widehat{W}' as simplicial complexes, then for homeomorphic wires, the geometric realisations of the two complexes as topological spaces (i.e. as a circle or the unit interval) are homeomorphic.

Definition 5.2.7. Two string graphs G and G' are called *wire-homeomorphic* if G' can be obtained from G by replacing any number of closed wires \widehat{W} with homeomorphic wires \widehat{W}' .

For any string graph G , there is a unique smallest wire-homeomorphic graph $G \downarrow$ obtained by replacing every wire in G with a homeomorphic wire containing a single wire-vertex. We justify the notation $G \downarrow$ by formalising wire-homeomorphism using a rewrite system \mathbb{H} .

Definition 5.2.8. For a monoidal signature $T = (O, M, \text{dom}, \text{cod})$, the rewrite system \mathbb{H} is defined as follows. For every $X \in O$, we define a loop contraction rule h_X^L and a wire contraction rule h_X^W :



For every $f \in M$ and $0 \leq i < \text{Length}(\text{dom}(f))$, $0 \leq j < \text{Length}(\text{cod}(f))$, we define an input contraction rule $h_{f,i}^I$ and an output contraction rule $h_{f,j}^O$:



Proposition 5.2.9. Two string graphs G, H are wire-homeomorphic if and only if $G \xleftrightarrow{*} \mathbb{H} H$.

Proof. For (\Leftarrow) , we observe that all of the rules in \mathbb{H} leave the endpoints of a wire fixed, while decreasing the number of other wire-vertices. For (\Rightarrow) , it suffices to show that \mathbb{H} lets us increase or decrease the number of wire-vertices in any wire in G . Consider the types of wire \widehat{W} that can occur in G :

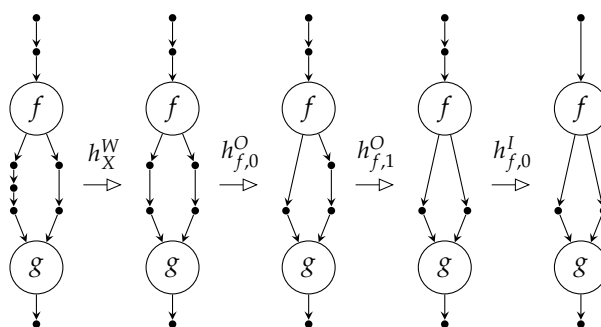
1. a simple cycle,
2. a chain starting with a boundary and ending with a boundary,
3. a chain ending with a box-vertex, or
4. a chain starting with a box-vertex.

These alternatives are not mutually exclusive, but they are exhaustive. In the case of (1.) the rules h_X^L or h_X^W can always decrease the size of \widehat{W} if applied forwards, and increase the size of \widehat{W} if applied backwards. In the case of (2.) apply h_X^W , for (3.) apply $h_{f,i}^L$, and for (4.) apply $h_{f,j}^O$. \square

Lemma 5.2.10. *The rewrite system \mathbb{H} is confluent (up to graph isomorphism) and terminating.*

Proof. Termination comes from observing that each contraction rule strictly decreases the total number of wire-vertices. Confluence follows from noting that any forward-directed rewrite procedure starting with G terminates at the unique minimal wire-homeomorphic graph $G \downarrow$. \square

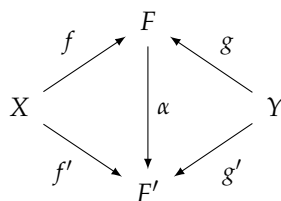
Example 5.2.11. Normalising a string graph with respect to \mathbb{H} :



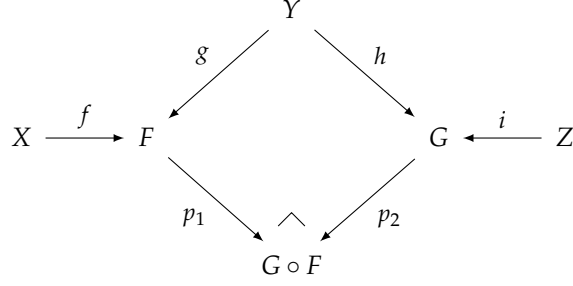
Note that there are multiple ways this string graph could be normalised, but since \mathbb{H} is confluent, the end result will always be the same.

5.3 Cospans over the Category of String Graphs

For any category \mathcal{C} with pushouts, we can form its *cospan bicategory* $\text{Csp}(\mathcal{C})$. The 0-cells of $\text{Csp}(\mathcal{C})$ are the objects from \mathcal{C} , the 1-cells are cospans $X \xrightarrow{f} F \xleftarrow{g} Y$, and the 2-cells are cospan homomorphisms. A cospan homomorphism from $X \xrightarrow{f} F \xleftarrow{g} Y$ to $X \xrightarrow{f'} F' \xleftarrow{g'} Y$ is a morphism $\alpha : F \rightarrow F'$ in \mathcal{C} that commutes with the cospan maps.



2-cell composition is the usual composition of morphisms in \mathcal{C} . For two cospans $X \xrightarrow{f} F \xleftarrow{g} Y$ and $Y \xrightarrow{h} G \xleftarrow{i} Z$, the composition is formed by pushing out over Y .



The composed cospan is then $X \xrightarrow{p_1 f} G \circ F \xleftarrow{p_2 i} Z$. Defining $\text{Id}_X := X$, identity 1-cells are cospans of identity maps: $X \xrightarrow{1} \text{Id}_X \xleftarrow{1} X$. It follows from general properties of pushouts that the following are cospan isomorphisms:

$$(H \circ G) \circ F \cong H \circ (G \circ F) \quad \text{Id}_Y \circ G \cong G \cong G \circ \text{Id}_X$$

We can form the ordinary category $\text{csp}(\mathcal{C})$ by taking objects to be the 0-cells from $\text{Csp}(\mathcal{C})$ and arrows to be isomorphism-classes of 1-cells in $\text{Csp}(\mathcal{C})$.

Sobociński and Sassone [64, 61] have extensively studied rewrite systems in the context of bicategories, and in particular bicategories of cospans over adhesive categories. In this section, we will focus particular types of cospan constructions over string graphs that will be used in sections 5.4 and 5.5 to construct free monoidal categories and free monoidal categories containing algebraic structures.

Recall that for string graphs, certain S -pushouts called *pluggings* perform the function of composition. Using cospans of string graphs $X \xrightarrow{d} G \xleftarrow{c} Y$ where X and Y are point graphs, we can “pin” the inputs and outputs for a particular graph in place (i.e. distinguish domain from codomain and induce a total order) and allow us to define composition unambiguously. To ensure that the cospans we consider are meaningful in terms of morphisms in monoidal categories and every cospan composition is a plugging, we shall restrict our attention to *framed point graphs* and *framed cospans* in \mathbf{SGraph}_T .

Definition 5.3.1. A *framed point graph* is a triple $(X, <, \text{sgn})$ where X is a point graph, $<$ is a total order on V_X , and $\text{sgn} : V_X \rightarrow \{+, -\}$ is called a *signing map*. A cospan $X \xrightarrow{d} G \xleftarrow{c} Y$ is called a *framed cospan* if:

1. X and Y are framed point graphs,
2. G contains no isolated wire-vertices,
3. the induced map $[d, c] : X + Y \rightarrow G$ restricts to an isomorphism $[d, c]' : X + Y \xrightarrow{\sim} \text{Bound}(G)$,
4. for every $v \in V_X$, $d(v) \in \text{In}(G) \Leftrightarrow \text{sgn}(v) = +$, and

5. for every $v \in V_Y$, $c(v) \in \text{Out}(G) \Leftrightarrow \text{sgn}(v) = +$.

The function sgn assigns a polarity to each element of the boundary. A positive polarity marks a wire that runs in the usual (downward) direction, whereas a negative polarity marks a wire that runs in the dual (upward) direction.

Notation 5.3.2. For a framed point graph X , let X^* be the same framed point graph with the signs reversed.

A totally downward-directed framed cospan is called *positive*.

Definition 5.3.3. A framed point graph X is called *positive* if $\text{sgn}(v) = +$ for all $v \in V_X$. A framed cospan $X \xrightarrow{d} G \xleftarrow{c} Y$ is called *positive* if both X and Y are positive.

Proposition 5.3.4. For two framed cospans:

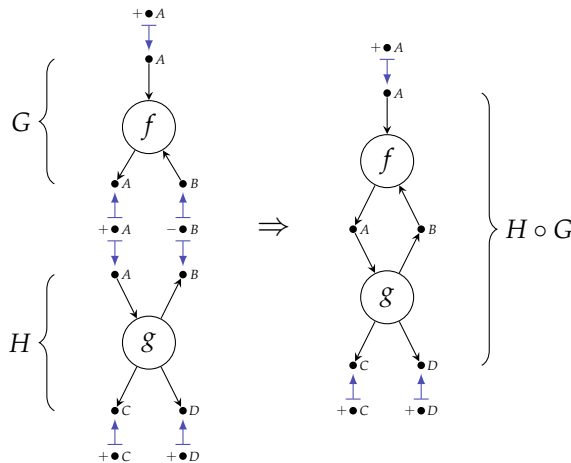
$$X \xrightarrow{d} G \xleftarrow{c} Y \qquad Y \xrightarrow{d'} H \xleftarrow{c'} Z$$

The span $G \xleftarrow{c} Y \xrightarrow{d'} H$ is a *plugging*.

Proof. For each $v \in V_Y$, if $\text{sgn}(v) = +$ then $c(v) \in \text{Out}(G)$ and $d'(v) \in \text{In}(H)$. If $\text{sgn}(v) = -$, then $c(v) \in \text{In}(G)$ and $d'(v) \in \text{Out}(H)$. Thus $G \xleftarrow{c} Y \xrightarrow{d'} H$ is boundary-coherent. Y is a point graph, so the span is a plugging. \square

As a consequence, for framed cospans G, H , the composition $H \circ G$ exists and is computed by S-pushout.

Example 5.3.5. Composing framed cospans by plugging:



Definition 5.3.6. For a framed point graph (X, \leq, sgn) , the *pseudo-identity* cospan

$$X \xrightarrow{d} \mathbb{1}_X \xleftarrow{c} X$$

is compatible with cospan composition. In particular, we show for $|G|$ an equivalence class of cospans over a given rewrite system, we can define $|H| \circ |G| := |H \circ G|$ in a way that does not depend on the choice of representatives G and H .

Definition 5.4.1. Let $X \xrightarrow{d} G \xleftarrow{c} Y$ and $X \xrightarrow{d'} H \xleftarrow{c'} Y$ be cospans in \mathbf{SGraph}_T , and let $L \rightarrowtail R$ be a string graph rewrite rule. For a matching m , the following rewrite:

$$\begin{array}{ccccc}
 L & \xleftarrow{b_1} & B & \xrightarrow{b_2} & R \\
 m \downarrow & \lrcorner & \downarrow & \lrcorner & \downarrow \\
 G & \xleftarrow{i_1} & G' & \xrightarrow{i_2} & H
 \end{array}$$

is called a *cospan rewrite* if the maps i_1 and i_2 are cospan homomorphisms. That is, there exist maps \widehat{d}, \widehat{c} such that the following diagram commutes:

$$\begin{array}{ccccc}
 & & G & & \\
 & d \nearrow & \uparrow i_1 & \nwarrow c & \\
 X & \xrightarrow{\widehat{d}} & G' & \xleftarrow{\widehat{c}} & Y \\
 & d' \searrow & \downarrow i_2 & \swarrow c' & \\
 & & H & &
 \end{array}$$

We can show that any rewrite on string graphs lifts to a cospan rewrite, for unique maps \widehat{d}, \widehat{c} . This result relies on the fact that morphisms in a framed cospan factor uniquely through the S -pushout complements associated with a string graph rewrite.

Lemma 5.4.2. Let $L \xleftarrow{b_1} I \xrightarrow{b_2} R$ be a string graph rewrite rule, $X \xrightarrow{d} G \xleftarrow{c} Y$ a framed cospan, and $m : L \rightarrow G$ an S -matching on G . Then, for the associated S -pushout complement:

$$\begin{array}{ccc}
 I & \xrightarrow{b_1} & L \\
 \downarrow & & \downarrow m \\
 G' & \xrightarrow{i} & G
 \end{array}$$

the cospan maps d and c factor uniquely through i . That is, there exists unique \widehat{d}, \widehat{c} such that:

$$\begin{array}{ccc}
 X & \xrightarrow{\widehat{d}} & G' & \xleftarrow{\widehat{c}} & Y \\
 \searrow d & & \downarrow i & & \swarrow c \\
 & & G & &
 \end{array}$$

Proof. G' is the subgraph of G that has the interior of L removed and i is the inclusion of G' in G . For any vertex v in the interior of L , v cannot be in $\text{Bound}(L)$ by definition of string graph rewrite rule. Thus, there exists no graph homomorphism m such that $m(v) \in \text{Bound}(G)$. By the definition of framed cospan, the image of d is contained in $\text{Bound}(G)$, so for all $v' \in V_X$, $d(v') \in V_{G'}$. Letting $\widehat{d}(v') = d(v')$, we have $i \circ \widehat{d} = d$. The existence of \widehat{c} follows similarly. Uniqueness is automatic, since i is a monomorphism. \square

Corollary 5.4.3. *A string graph rewrite $G \rightarrow H$ lifts uniquely to a rewrite of framed cospans:*

$$X \xrightarrow{d} G \xleftarrow{c} Y \rightarrow X \xrightarrow{d'} H \xleftarrow{c'} Y$$

Notation 5.4.4. For a string graph rewrite system \mathcal{R} and a cospan $X \xrightarrow{d} G \xleftarrow{c} Y$, we write $|G|_{\mathcal{R}}$ for the set of cospans G' such that:

$$X \xrightarrow{d} G \xleftarrow{c} Y \xleftarrow{*} \mathcal{R} X \xrightarrow{d'} G' \xleftarrow{c'} Y$$

We drop the subscript \mathcal{R} when it is clear from context.

Theorem 5.4.5. *Rewriting commutes with composition. Let the following diagram define a composition of framed cospans:*

$$\begin{array}{ccccc}
 & & Y & & \\
 & c_1 \swarrow & & \searrow d_2 & \\
 X & \xrightarrow{d_1} & G & & H \xleftarrow{c_2} Z \\
 & & \searrow i_1 & & \swarrow i_2 \\
 & & \wedge & & \\
 & & H \circ G & &
 \end{array} \tag{5.2}$$

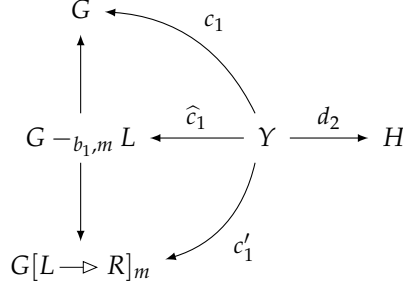
For a string graph rewrite rule $L \rightarrow R$ and an S -matching $m : L \rightarrow G$, the composed morphism $i_1 m : L \rightarrow H \circ G$ is an S -matching and:

$$H \circ (G[L \rightarrow R]_m) \cong (H \circ G)[L \rightarrow R]_{i_1 m} \tag{5.3}$$

Similarly, for $n : L \rightarrow H$, $i_2 n : L \rightarrow H \circ G$ is an S -matching and:

$$(H[L \rightarrow R]_n) \circ G \cong (H \circ G)[L \rightarrow R]_{i_2 n} \tag{5.4}$$

Proof. i_1 is a monomorphism because the pushout in (5.2) is a plugging, so $i_1 m$ is a monomorphism, hence an S -matching by Theorem 5.2.3. By Lemma 5.4.2, the map c_1 factors through the pushout complement $G -_{b_1, m} L$ as in diagram (4.9):



All three pairs (c_1, d) , (\widehat{c}_1, d) , and (c'_1, d) are pluggings, so they are S-spans. Isomorphism (5.3) follows from Theorem 4.4.10. Isomorphism (5.4) follows similarly. \square

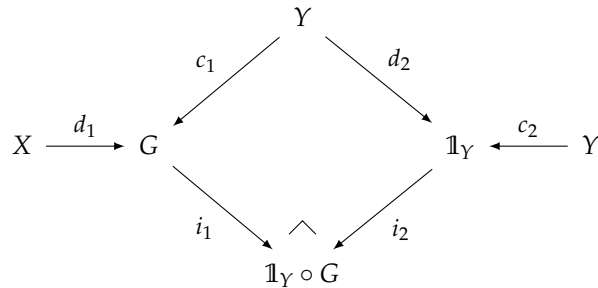
Corollary 5.4.6. *For a string graph rewrite system \mathcal{R} , composition given by $|H|_{\mathcal{R}} \circ |G|_{\mathcal{R}} := |H \circ G|_{\mathcal{R}}$ is well defined, and does not depend on the choices of G and H .*

Proof. If $G' \in |G|_{\mathcal{R}}$ and $H' \in |G|_{\mathcal{R}}$, then by Theorem 5.4.5, we can always find matchings to rewrite $H' \circ G'$ into $H \circ G$, so $H' \circ G' \in |H \circ G|_{\mathcal{R}}$. \square

Theorem 5.4.7. *For a rewrite system \mathcal{R} containing the wire-homeomorphism rules \mathbb{H} , $\text{FCsp}(\mathcal{R}, T)$ is a category where:*

- objects are framed point graphs,
- arrows are equivalence classes $|G|_{\mathcal{R}}$ of framed cospans,
- identities are defined by pseudo-identity cospans: $|\mathbb{1}_X|_{\mathcal{R}}$, and
- composition is given by: $|H|_{\mathcal{R}} \circ |G|_{\mathcal{R}} := |H \circ G|_{\mathcal{R}}$.

Proof. Composition is associative, because it is defined by pushouts as in cospan categories, so it remains to show that the pseudo-identity string graphs yield genuine identities in $\text{FCsp}(\mathcal{R}, T)$. For a framed cospan G , consider the composition:



$\mathbb{1}_Y \circ G$ contains a copy of G , as well as some extra edges and wire-vertices. For every wire-vertex $v \in W(\mathbb{1}_Y \circ G) - W(G)$, there exists a unique edge $e \in E_{\mathbb{1}_Y \circ G} - E_G$ adjacent to that vertex. We can always find a rewrite rule in $\mathbb{H} \subseteq \mathcal{R}$ that produces a string graph isomorphic to $\mathbb{1}_Y \circ G$ with v and

e removed. Since all edges $e \in E_{\mathbb{1}_Y \circ G} - E_G$ arise in this way, we can repeat the process until the resultant string graph is isomorphic to G . So $|\mathbb{1}_Y \circ G|_{\mathcal{R}} = |G|_{\mathcal{R}}$. Similarly, $|G \circ \mathbb{1}_X|_{\mathcal{R}} = |G|_{\mathcal{R}}$. \square

We often want to consider the full subcategory of $\text{FCsp}(\mathcal{R}, T)$ containing only equivalence classes of “downward-directed” cospans. This category is called $\text{FCsp}^+(\mathcal{R}, T)$.

Definition 5.4.8. $\text{FCsp}^+(\mathcal{R}, T)$ be the full subcategory of $\text{FCsp}(\mathcal{R}, T)$ whose objects are positive framed point graphs.

Theorem 5.4.9. $\text{FCsp}^+(\mathcal{R}, T)$ is a symmetric traced category and $\text{FCsp}(\mathcal{R}, T)$ is a compact closed category.

Proof. First, we show that $\text{FCsp}(\mathcal{R}, T)$ is a compact closed category. For any framed graphs A, B , $A \otimes B$ has vertices $V_A + V_B$ which can be given a total order by placing all of the elements in V_B above those in V_A (i.e. the usual disjoint union of totally ordered sets). The monoid product of framed cospans is given coproducts in \mathbf{SGraph}_T . For cospans:

$$A \xrightarrow{d_1} G \xleftarrow{c_1} C \qquad B \xrightarrow{d_2} H \xleftarrow{c_2} C$$

the underlying string graphs of $A \otimes B$ and $C \otimes D$ are coproducts $A + B$ and $C + D$, so there is an induced cospan over $G \otimes H := G + H$:

$$\begin{array}{ccccc} A & \hookrightarrow & A \otimes B & \longleftarrow & B \\ d_1 \downarrow & & \downarrow [d_1, d_2] & & \downarrow d_2 \\ G & \hookrightarrow & G \otimes H & \longleftarrow & H \\ c_1 \uparrow & & \uparrow [c_1, c_2] & & \uparrow c_2 \\ C & \hookrightarrow & C \otimes D & \longleftarrow & D \end{array}$$

It can easily be verified that this induced cospan is framed. The rest of the structure maps are analogous to those from string diagrams:

$$\sigma_{A,B} := \begin{array}{c} \begin{array}{cc} \bullet^+ A & \bullet^+ B \\ \downarrow & \downarrow \\ \bullet A & \bullet B \\ \swarrow & \searrow \\ \bullet B & \bullet A \\ \uparrow & \uparrow \\ \bullet^+ A & \bullet^+ B \end{array} \end{array} \qquad d_A := \begin{array}{c} \begin{array}{cc} \bullet A & \bullet A \\ \uparrow & \uparrow \\ \bullet A & \bullet A \end{array} \end{array} \qquad e_A := \begin{array}{c} \begin{array}{cc} \bullet^+ A & \bullet^- A \\ \downarrow & \downarrow \\ \bullet A & \bullet A \\ \uparrow & \uparrow \end{array} \end{array}$$

All of the axioms of a compact closed category then follow from string graph isomorphism and edge homeomorphism. For example:

$$(e_A \otimes 1_A) \circ (1_A \otimes d_A) = \text{Diagram 1} \Rightarrow \text{Diagram 2} \cong \text{Diagram 3} \xrightarrow{*} \text{Diagram 4} = 1_A$$

Since any full subcategory of a compact closed category is a symmetric traced category, $\text{FCsp}^+(\mathcal{R}, T)$ is a symmetric traced category, with the trace operation defined using the compact structure from $\text{FCsp}(\mathcal{R}, T)$. \square

5.5 Free Monoidal Categories

In this section, we will prove that the category $\text{FCsp}^+(\mathbb{H}, T)$ is the free symmetric trace category on a monoidal signature T . Free categories are special in that they are sound and complete with respect to the axioms of that type of category. That is, we shall prove that two morphisms are equal by the axioms of a symmetric traced category *if and only if* they are equal in $\text{FCsp}^+(\mathbb{H}, T)$. Once we prove this, it is a simple matter to show that $\text{FCsp}(\mathbb{H}, T)$ is the free *compact closed* category on a monoidal signature, by proving that the category $\text{FCsp}(\mathbb{H}, T)$ is equivalent to the result of performing the “Int” construction [34] on the free symmetric traced category $\text{FCsp}^+(\mathbb{H}, T)$. For simplicity, we will focus on strict categories in this section.

Before we get to the bulk of the proof, we introduce some notation. The first thing we introduce is the notion of *indexing* a morphism.

Definition 5.5.1. For a small, strict monoidal category \mathcal{V} , fix a set of *atomic objects* O , such that any object in \mathcal{V} is a monoidal product of elements of O . For an object $X \in \text{ob}\mathcal{V}$, an *X-word* is a monoidal product $X_{i_1} \otimes \dots \otimes X_{i_M} = X$ such that all i_k are distinct and $X_{i_k} \in O$.

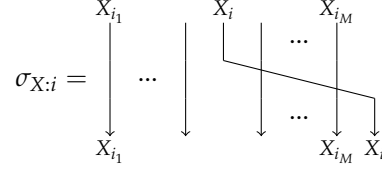
We will assume that O contains “enough copies” of every atomic object to find X -words for every object X . Replacing X with an X -word is simply the act of binding each position in the monoidal product to a unique index that we can refer to later.

Definition 5.5.2. For a morphism $f : X \rightarrow Y$, an *indexing* of f is a choice of an X -word and a Y -word such that

$$f = f' : X_{i_1} \otimes \dots \otimes X_{i_M} \rightarrow Y_{j_1} \otimes \dots \otimes Y_{j_N}$$

A morphism from an X -word to a Y -word for any X, Y is called *indexed*.

For an X -word $X_{i_1} \otimes \dots \otimes X_{i_M}$, and an index $i \in \{i_1, \dots, i_M\}$, $\sigma_{X:i}$ is the (unique) symmetry map that permutes the object X_i to the end of the list and leaves the other objects fixed.



In any strict symmetric traced category, we can define a contraction operator $C_i^j(-)$ which “traces together” the i -th input with the j -th output on an indexed morphism.

Definition 5.5.3. Let $f : X_{i_1} \otimes \dots \otimes X_{i_M} \rightarrow Y_{j_1} \otimes \dots \otimes Y_{j_N}$ be an indexed morphism in a symmetric traced category such that for indices $i \in \{i_1, \dots, i_M\}$ and $j \in \{j_1, \dots, j_N\}$, $X_i = Y_j$. Then we define the contraction $C_i^j(f)$ as follows:

$$C_i^j(f) := \text{Tr}^{X_i=Y_j}(\sigma_{Y:j} \circ f \circ \sigma_{X:i}^{-1})$$

Note that a contraction of an indexed morphism yields an indexed morphism, so we can contract many times. Also, the resulting morphism does not depend on the order in which we perform contractions.

Lemma 5.5.4. Contractions are commutative. For an indexed morphism f distinct indices i, i' and j, j' :

$$C_i^j(C_{i'}^{j'}(f)) = C_{i'}^{j'}(C_i^j(f))$$

Proof. Let $X = X_{i_1} \otimes \dots \otimes X_{i_M}$ and $Y = Y_{j_1} \otimes \dots \otimes Y_{j_N}$, let $i, i' \in \{i_1, \dots, i_M\}$ be distinct, and let $j, j' \in \{j_1, \dots, j_N\}$ be distinct.

$$\begin{aligned} C_i^j(C_{i'}^{j'}(f)) &= \text{Tr}^{X_i=Y_j}(\sigma_{Y:j} \circ \text{Tr}^{X_{i'}=Y_{j'}}(\sigma_{Y:j'} \circ f \circ \sigma_{X:i'}^{-1}) \circ \sigma_{X:i}^{-1}) \\ &= \text{Tr}^{X_i=Y_j}(\text{Tr}^{X_{i'}=Y_{j'}}((\sigma_{Y:j} \otimes 1_{Y_{j'}}) \circ \sigma_{Y:j'} \circ f \circ \sigma_{X:i'}^{-1} \circ (\sigma_{X:i}^{-1} \otimes 1_{X_{i'}}))) \\ &= \text{Tr}^{X_i \otimes X_{i'}=Y_j \otimes Y_{j'}}((\sigma_{Y:j} \otimes 1_{Y_{j'}}) \circ \sigma_{Y:j'} \circ f \circ \sigma_{X:i'}^{-1} \circ (\sigma_{X:i}^{-1} \otimes 1_{X_{i'}})) \\ &= (*) \end{aligned}$$

Let X' be a new X -word equal to X with the factors X_i and $X_{i'}$ deleted. Let Y' be a Y -word equal

to Y with the factors at Y_j and $Y_{j'}$ omitted.

$$\begin{aligned}
(*) &= \text{Tr}^{X_i \otimes X_{i'} = Y_j \otimes Y_{j'}}((\sigma_{Y:j} \otimes 1_{Y_{j'}}) \circ \sigma_{Y:j'} \circ f \circ \sigma_{X:i'}^{-1} \circ (\sigma_{X:i}^{-1} \otimes 1_{X_{i'}}) \circ (1_{X'} \otimes 1_{X_i} \otimes 1_{X_{i'}})) \\
&= \text{Tr}^{X_i \otimes X_{i'} = Y_j \otimes Y_{j'}}((\sigma_{Y:j} \otimes 1_{Y_{j'}}) \circ \sigma_{Y:j'} \circ f \circ \sigma_{X:i'}^{-1} \circ (\sigma_{X:i}^{-1} \otimes 1_{X_{i'}}) \circ (1_{X'} \otimes (\sigma_{X_i, X_{i'}}^{-1} \circ \sigma_{X_i, X_{i'}}))) \\
&= \text{Tr}^{X_{i'} \otimes X_i = Y_{j'} \otimes Y_j}((1_{Y'} \otimes \sigma_{X_i, X_{i'}}) \circ (\sigma_{Y:j} \otimes 1_{Y_{j'}}) \circ \sigma_{Y:j'} \circ f \circ \sigma_{X:i'}^{-1} \circ (\sigma_{X:i}^{-1} \otimes 1_{X_{i'}}) \circ (1_{X'} \otimes \sigma_{X_i, X_{i'}}^{-1})) \\
&= \text{Tr}^{X_{i'} \otimes X_i = Y_{j'} \otimes Y_j}((1_{Y'} \otimes \sigma_{Y_j, Y_{j'}}) \circ (\sigma_{Y:j} \otimes 1_{Y_{j'}}) \circ \sigma_{Y:j'} \circ f \circ \sigma_{X:i'}^{-1} \circ (\sigma_{X:i}^{-1} \otimes 1_{X_{i'}}) \circ (1_{X'} \otimes \sigma_{X_i, X_{i'}}^{-1})) \\
&= (*)
\end{aligned}$$

Now, we need to have a look at the symmetry maps. By the coherence theorem for symmetric monoidal categories, a composition of symmetry maps is uniquely defined by the permutation it performs. First note that $(1_{Y'} \otimes \sigma_{Y_j, Y_{j'}}) \circ (\sigma_{Y:j} \otimes 1_{Y_{j'}})$ sends Y_j to the end of the list, so:

$$(1_{Y'} \otimes \sigma_{Y_j, Y_{j'}}) \circ (\sigma_{Y:j} \otimes 1_{Y_{j'}}) = \sigma_{Y:j}$$

Note that the $\sigma_{Y:j}$ on the LHS refers to a different map than the one on the RHS, as it has a different domain and codomain. By naturality:

$$\sigma_{Y:j} \circ \sigma_{Y:j'} = (\sigma_{Y:j'} \otimes 1_{Y_j}) \circ \sigma_{Y:j}$$

It can be shown similarly that:

$$\sigma_{X:i'}^{-1} \circ (\sigma_{X:i}^{-1} \otimes 1_{X_{i'}}) \circ (1_{X'} \otimes \sigma_{X_i, X_{i'}}^{-1}) = \sigma_{X:i'}^{-1} \circ \sigma_{X:i}^{-1} = \sigma_{X:i}^{-1} \circ (\sigma_{X:i'}^{-1} \otimes 1_{X_i})$$

Substituting in to the above expression, we complete the proof:

$$\begin{aligned}
(*) &= \text{Tr}^{X_{i'} \otimes X_i = Y_{j'} \otimes Y_j}((\sigma_{Y:j'} \otimes 1_{Y_j}) \circ \sigma_{Y:j} \circ f \circ \sigma_{X:i'}^{-1} \circ (\sigma_{X:i}^{-1} \otimes 1_{X_i})) \\
&= \text{Tr}^{X_{i'} = Y_{j'}}(\text{Tr}^{X_i = Y_j}((\sigma_{Y:j'} \otimes 1_{Y_j}) \circ \sigma_{Y:j} \circ f \circ \sigma_{X:i'}^{-1} \circ (\sigma_{X:i}^{-1} \otimes 1_{X_i}))) \\
&= \text{Tr}^{X_{i'} = Y_{j'}}(\sigma_{Y:j'} \circ \text{Tr}^{X_i = Y_j}(\sigma_{Y:j} \circ f \circ \sigma_{X:i'}^{-1} \circ \sigma_{X:i}^{-1})) \\
&= C_{i'}^{j'}(C_i^j(f))
\end{aligned}$$

□

Definition 5.5.5. For a (small, strict) traced symmetric category \mathcal{V} , define a set M of *atomic* morphisms, such that any morphism in \mathcal{V} can be obtained from those morphisms and the traced symmetric structure. An indexed morphism is called *disconnected* if it is of the form $f = f_1 \otimes \dots \otimes f_K$, where each f_k is an indexed morphism in M :

$$f_k : X_{i_{k,1}} \otimes \dots \otimes X_{i_{k,M_k}} \rightarrow Y_{j_{k,1}} \otimes \dots \otimes Y_{j_{k,N_k}}$$

Definition 5.5.6. Let $f = f_1 \otimes \dots \otimes f_M$ be a disconnected indexed map. For distinct indices $\{i_1, \dots, i_P\} \subseteq \{i_{1,1}, \dots, i_{K,M_K}\}$ and $\{j_1, \dots, j_P\} \subseteq \{j_{1,1}, \dots, j_{K,N_K}\}$, a map f' is said to be in *contraction normal form* (CNF) if:

$$f' = C_{i_1}^{j_1}(C_{i_2}^{j_2}(\dots(C_{i_P}^{j_P}(f))\dots))$$

Definition 5.5.7. Let f and f' be given as in Definition 5.5.6. A component f_k of f is said to be *totally contracted* if the indices of all of its inputs occur in $\{i_1, \dots, i_p\}$ and the indices of all of its outputs occur in $\{j_1, \dots, j_p\}$.

Lemma 5.5.8. Let f and f' be given as in Definition 5.5.6. By re-indexing the contraction, we can reorder the totally contracted components of f arbitrarily.

Proof. It suffices to show that we can send any totally-contracted f_k to the far right side of f by re-indexing the contraction.

$$C_{i_1}^{j_1}(C_{i_2}^{j_2}(\dots(C_{i_p}^{j_p}(f_1 \otimes \dots \otimes f_k \otimes \dots \otimes f_K))\dots)) = C_{i'_1}^{j'_1}(C_{i'_2}^{j'_2}(\dots(C_{i'_p}^{j'_p}(f_1 \otimes \dots \otimes f_K \otimes f_k))\dots))$$

We can show this by applying the previous lemma and naturality of symmetries. First, write f as $f = f^L \otimes f_k \otimes f^R$ for $f^L : X^L \rightarrow Y^L$, $f^R : X^R \rightarrow Y^R$, and $f_k : X^k \rightarrow Y^k$.

$$C_{i_1}^{j_1}(\dots(C_{i_p}^{j_p}(f^L \otimes f_k \otimes f^R)\dots)) = (*)$$

We then pre-compose with the identity (i.e. a swap map and its inverse). Since f_k and all the maps after it are totally-contracted, we can eliminate one of the swap maps by re-indexing the i 's. Similarly, we can introduce a swap after f by re-indexing the j 's.

$$\begin{aligned} (*) &= C_{i_1}^{j_1}(\dots(C_{i_p}^{j_p}((f^L \otimes f_k \otimes f^R) \circ (X^L \otimes \sigma_{X^R, X^k}) \circ (X^L \otimes \sigma_{X^k, X^R}))\dots)) \\ &= C_{i'_1}^{j'_1}(\dots(C_{i'_p}^{j'_p}((f^L \otimes f_k \otimes f^R) \circ (X^L \otimes \sigma_{X^R, X^k}))\dots)) \\ &= C_{i'_1}^{j'_1}(\dots(C_{i'_p}^{j'_p}((Y^L \otimes \sigma_{Y^k, Y^R}) \circ (f^L \otimes f_k \otimes f^R) \circ (X^L \otimes \sigma_{X^R, X^k}))\dots)) = (*) \end{aligned}$$

An application of naturality completes the proof:

$$\begin{aligned} (*) &= C_{i'_1}^{j'_1}(\dots(C_{i'_p}^{j'_p}((f^L \otimes f^R \otimes f_k) \circ (X^L \otimes \sigma_{X^k, X^R}) \circ (X^L \otimes \sigma_{X^R, X^k}))\dots)) \\ &= C_{i'_1}^{j'_1}(\dots(C_{i'_p}^{j'_p}(f^L \otimes f^R \otimes f_k)\dots)) \end{aligned}$$

We can therefore send f_k to an arbitrary position by performing this procedure in reverse. \square

A totally contracted identity map whose input is connected to its output is called a *minimal circle*.

Lemma 5.5.9. Let f, f' be Defined as in 5.5.6. If $f_k = 1_{X_{i_k,1}} = 1_{Y_{j_k,1}}$ is a totally contracted identity map that is not a minimal circle, then it can be removed by re-indexing.

Proof. From Lemma 5.5.8, we can assume the totally contracted identity map is on the far right. Let M be the index of the input and N be the index of the output. By 5.5.4, we can move the two contractions involving the identity map all the way to the inside, so f' is of the form:

$$f' = C(C(\dots(C_M^j(C_i^N(f'' \otimes 1_{X_M})))\dots))$$

We can reduce the inner map using the definition of $\sigma_{Y;j}$ and the trace axioms.

$$\begin{aligned}
C_M^j(C_i^N(f'' \otimes 1_{X_M})) &= \text{Tr}^{X_M=Y_j}(\sigma_{Y;j} \circ \text{Tr}^{X_i=Y_N}((f'' \otimes 1_{X_M}) \circ \sigma_{X;i}^{-1})) \\
&= \text{Tr}^{X_M=Y_j}(\sigma_{Y;j} \circ \text{Tr}^{X_i=Y_N}((f'' \otimes 1_{X_M}) \circ (\sigma_{X;i}^{-1} \otimes 1_{X_M}) \circ (1_{X'} \otimes \sigma_{X_M, X_i}^{-1}))) \\
&= \text{Tr}^{X_M=Y_j}(\sigma_{Y;j} \circ \text{Tr}^{X_i=Y_N}(((f'' \circ \sigma_{X;i}^{-1}) \otimes 1_{X_M}) \circ (1_{X'} \otimes \sigma_{X_M, X_i}^{-1}))) \\
&= \text{Tr}^{X_M=Y_j}(\sigma_{Y;j} \circ f'' \circ \sigma_{X;i}^{-1} \circ \text{Tr}^{X_i=Y_N}((1_{X'} \otimes \sigma_{X_M, X_i}^{-1}))) \\
&= \text{Tr}^{X_M=Y_j}(\sigma_{Y;j} \circ f'' \circ \sigma_{X;i}^{-1} \circ (1_{X'} \otimes \text{Tr}^{X_i=Y_N}(\sigma_{X_M, X_i}^{-1}))) \\
&= \text{Tr}^{X_M=Y_j}(\sigma_{Y;j} \circ f'' \circ \sigma_{X;i}^{-1} \circ (1_{X'} \otimes 1_{X_M=X_i})) \\
&= \text{Tr}^{X_i=Y_j}(\sigma_{Y;j} \circ f'' \circ \sigma_{X;i}^{-1}) = C_i^j(f'')
\end{aligned}$$

□

Theorem 5.5.10. $\text{FCsp}^+(\mathbb{H}, T)$ is the free strict symmetric traced category on the monoidal signature T .

Proof. Let $T = (O, M, \text{dom}, \text{cod})$ be a monoidal signature, \mathcal{V} be a strict symmetric traced category, and $F : T \rightarrow \mathcal{V}$ be a monoidal signature homomorphism. There is an evident signature homomorphism $E : T \rightarrow \text{FCsp}^+(\mathbb{H}, T)$, taking each object $Z \in O$ to the framed point graph consisting of a single wire-vertex of type Z and taking each $g \in M$ to a homeomorphism-class of cospans consisting only of a box-vertex of type g and its inputs and outputs.

Let X be a framed, positive point graph, where $V_X = \{p_1 < p_2 < \dots < p_N\}$. Then, for all p_i , $F(\tau_X(p_i))$ is an object in \mathcal{V} . Let:

$$\widehat{F}(X) = F(\tau_X(p_1)) \otimes \dots \otimes F(\tau_X(p_N))$$

This map is uniquely specified, and it respects the monoidal product on objects in $\text{FCsp}^+(\mathbb{H}, T)$. Let $|G| : X \rightarrow Y$ be an arrow in $\text{FCsp}^+(\mathbb{H}, T)$, represented by a cospan $X \xrightarrow{d} G \xleftarrow{c} Y$. Let $\{x_1 < x_2 < \dots < x_M\}$ be set of wire-vertices in G that are in the image of d , inheriting the total ordering from X . Let $\{y_1 < y_2 < \dots < y_N\}$ be the same for Y . Define a function F' from V_G to the morphisms of \mathcal{V} as follows:

$$F'(v) = \begin{cases} F(\tau_G(v)) & \text{if } v \text{ is a box-vertex} \\ 1_{F(\tau_G(v))} & \text{if } v \text{ is a wire-vertex} \end{cases}$$

Let $\{z_1, \dots, z_K\} \subseteq V_G$ be the set of vertices not in the image of d or c . Define a disconnected indexed morphism in \mathcal{V} .

$$f = F'(x_1) \otimes \dots \otimes F'(x_M) \otimes F'(y_1) \otimes \dots \otimes F'(y_N) \otimes F'(z_1) \otimes \dots \otimes F'(z_K)$$

We can form a CNF term from $|G|$ by adding a contraction $C_i^j(-)$ for every edge e in G . We define the indices for these contractions as follows. If $\tau_G(e) = \text{mid}_Z$, then $F'(s(e))$ and $F'(t(e))$ are both identity maps. Let i be the unique input of $F'(t(e))$ and j is the unique output of $F'(s(e))$. If

$\tau_G(e) = \text{in}_{g,k}$, then $F'(s(e))$ is an identity map. Let j be the unique output of $F'(s(e))$ and let i be the k -th input of $F'(t(e))$. If $\tau_G(e) = \text{out}_{g,k}$, then $F'(t(e))$ is an identity map. Let i be the unique input of $F'(t(e))$ and let j be the k -th output of $F'(s(e))$. Define $\widehat{F}(|G|)$ as:

$$\widehat{F}(|G|) = C_{i_1}^{j_1}(C_{i_2}^{j_2}(\dots(C_{i_p}^{j_p}(f_1 \otimes \dots \otimes f_k \otimes \dots \otimes f_k)) \dots))$$

For this to be well-defined, we need to show that this does not depend on the choice of G . Choosing an isomorphic string graph cospan amounts to picking a different order for the internal vertices z_i and the contractions $C_i^j(-)$. Any internal vertices in G will be totally contracted in $\widehat{F}(|G|)$, so by Lemmas 5.5.4 and 5.5.8, choosing an isomorphic cospan will not affect the value of $\widehat{F}(|G|)$. Choosing a homeomorphic cospan amounts to varying the number of totally contracted identities that are not minimal circles. By Lemma 5.5.9, this does not affect the value of $\widehat{F}(|G|)$ either.

It is straightforward to show that \widehat{F} takes identities to identities and respects composition and traces, so \widehat{F} is a symmetric traced functor from $\text{FCsp}^+(\mathbb{H}, T)$ to \mathcal{V} . Thus \widehat{F} satisfies the universal property of the free symmetric traced category.

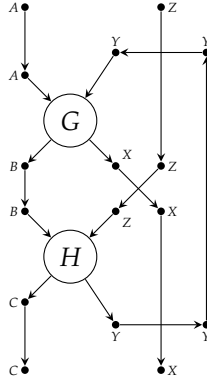
$$\begin{array}{ccc} T & \xrightarrow{E} & \text{FCsp}^+(\mathbb{H}, T) \\ & \searrow F & \downarrow \widehat{F} \\ & & \mathcal{V} \end{array}$$

Since it is possible to build any positive framed cospan using the cospans in the image of E and the traced symmetric structure, \widehat{F} is the unique map making this diagram commute. \square

Theorem 5.5.11. *There is a symmetric monoidal equivalence of categories $\text{Int}(\text{FCsp}^+(\mathbb{H}, T)) \cong \text{FCsp}(\mathbb{H}, T)$, such that the following diagram commutes:*

$$\begin{array}{ccc} \text{FCsp}^+(\mathbb{H}, T) & \hookrightarrow & \text{Int}(\text{FCsp}^+(\mathbb{H}, T)) \\ & \searrow & \downarrow \cong \\ & & \text{FCsp}(\mathbb{H}, T) \end{array} \quad (5.5)$$

Proof. The category $\text{Int}(\text{FCsp}^+(\mathbb{H}, T))$ has as objects pairs of positive framed point graphs (A, X) and as arrows \mathbb{H} -equivalence classes of cospans. An arrow $|G| : (A, X) \rightarrow (B, Y)$ is represented by a cospan $A + Y \xrightarrow{d} G \xleftarrow{c} B + X$. Composition is done by composing on the first object and “composing backwards” via the trace on the second object.



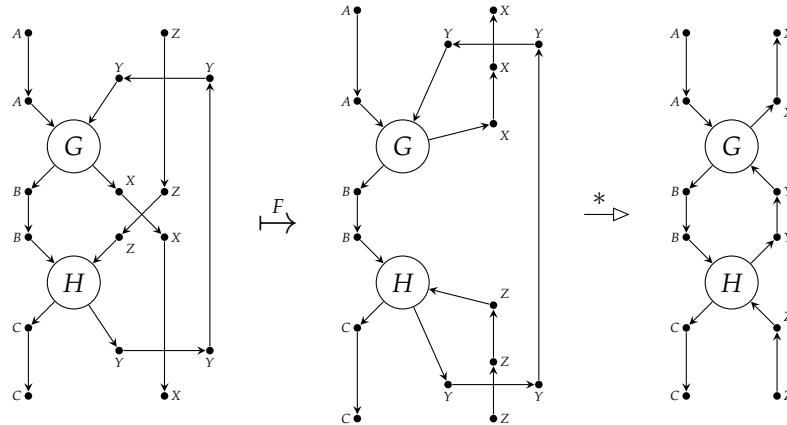
Since the domain and codomain are disjoint unions of framed point graphs, we can decompose the map d into maps $d_1 : A \rightarrow G$ and $d_2 : Y \rightarrow G$ and the map c into maps $c_1 : B \rightarrow G$ and $c_2 : X \rightarrow G$.

$$A + Y \xrightarrow{[d_1, d_2]} G \xleftarrow{[c_1, c_2]} B + X$$

We can then form a new cospan by interchanging X and Y and flipping their sign maps. Since d_2 and c_2 are just string graph homomorphisms, we can consider them to have domains X^* and Y^* , respectively. Thus we obtain a cospan representing an arrow in $\text{FCsp}(\mathbb{H}, T)$.

$$A + X^* \xrightarrow{[d_1, d_2]} G \xleftarrow{[c_1, d_2]} B + Y^* \quad (5.6)$$

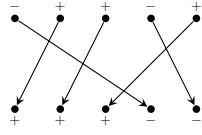
Let $F((A, X)) = A + X^*$ and let $F(|G|)$ be the \mathbb{H} -equivalence class of cospans represented by (5.6). We can show that F respects composition:



It can also be verified that F preserves all of the traced symmetric structure, up to isomorphism. Flipping X and Y induces a bijection of hom-sets:

$$\text{hom}_{\text{FCsp}^+(\mathbb{H}, T)}(A + Y, B + X) \cong \text{hom}_{\text{FCsp}(\mathbb{H}, T)}(A + X^*, B + Y^*)$$

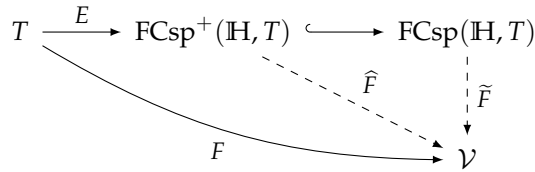
so F is full and faithful. For all positive framed point graphs A, X , objects $A + X^*$ are in the image of F . For an arbitrary object Z in $\text{FCsp}(\mathbb{H}, T)$, we can define a symmetry isomorphism that sends all the positive points to the left and all the negative points to the right.



So F is essentially surjective. Therefore there is a traced symmetric equivalence of categories $\text{Int}(\text{FCsp}^+(\mathbb{H}, T)) \cong \text{FCsp}(\mathbb{H}, T)$. $\text{FCsp}^+(\mathbb{H}, T)$ fully embeds into $\text{Int}(\text{FCsp}^+(\mathbb{H}, T))$ as objects (A, \emptyset) and cospans $A + \emptyset \xrightarrow{d} G \xleftarrow{c} B + \emptyset$. The functor F acts trivially on these cospans, so diagram (5.5) commutes. \square

Corollary 5.5.12. $\text{FCsp}(\mathbb{H}, T)$ is the free compact closed category on a monoidal signature T .

Proof. The embedding of T in $\text{FCsp}(\mathbb{H}, T)$ factors through $\text{FCsp}^+(\mathbb{H}, T)$. The universal property follows from composing free constructions.

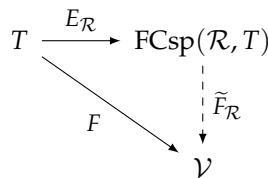


F is a monoidal signature homomorphism, \hat{F} is the unique traced symmetric functor induced by F , and \tilde{F} is the unique compact closed functor induced by \hat{F} . \square

A consequence of this construction, is we can quite easily build “free categories containing an X ” (e.g. the traced symmetric and compact closed analogues of Lawvere theories, PROPs, etc.) as rewrite categories. For an algebraic theory \mathcal{E} , we can translate all of the equations in \mathcal{E} into rewrite rules, forming a rewrite system \mathcal{R} . Applying the axioms of an algebraic structure to a morphism in a monoidal category corresponds precisely to rewriting with the rules in \mathcal{R} . Purely by construction, a category of the form $\text{FCsp}(\mathcal{R}, T)$ satisfies the axioms of \mathcal{E} , and nothing else. We can formalise this using another unique factorisation property.

For a rewrite rule $L \rightarrow R \in \mathcal{R}$, the string graphs L and R (regarded as cospans over their inputs and outputs) define morphisms $|L|_{\mathbb{H}}, |R|_{\mathbb{H}} \in \text{hom}_{\text{FCsp}(\mathbb{H}, T)}(X, Y)$ in the free compact closed category over T .

Theorem 5.5.13. Let \mathcal{R} be a string graph rewrite system containing \mathbb{H} . Let $F : T \rightarrow \mathcal{V}$ be a valuation of a monoidal signature T such that, for the functor $\tilde{F} : \text{FCsp}(\mathbb{H}, T) \rightarrow \mathcal{V}$ and every rewrite rule $L \rightarrow R \in \mathcal{R}$, $\tilde{F}(|L|_{\mathbb{H}}) = \tilde{F}(|R|_{\mathbb{H}})$, the valuation F factors uniquely as:



Proof. Note that for any rewrite system $\mathcal{R} \supseteq \mathbb{H}$, then $\langle \leftarrow^* \triangleright \rangle_{\mathcal{R}} \supseteq \langle \leftarrow^* \triangleright \rangle_{\mathbb{H}}$, so we can think of the morphisms of $\text{FCsp}(\mathcal{R}, T)$ as equivalence classes of morphisms in $\text{FCsp}(\mathbb{H}, T)$, where the quotient functor Q is identity-on-objects, and takes an equivalence class $|G|_{\mathbb{H}}$ to the unique (possibly larger) equivalence class $|H|_{\mathcal{R}} \supseteq |G|_{\mathbb{H}}$. This is a well-defined functor by Theorem 5.4.5, and it is possible to define $E_{\mathcal{R}} := Q \circ E$. Thus it suffices to show that \tilde{F} factors uniquely through Q :

$$\begin{array}{ccc}
 T & \xrightarrow{F} & \mathcal{V} \\
 E \downarrow & \nearrow \tilde{F} & \uparrow \tilde{F}_{\mathcal{R}} \\
 \text{FCsp}(\mathbb{H}, T) & \xrightarrow{Q} & \text{FCsp}(\mathcal{R}, T)
 \end{array}$$

For existence, let $\tilde{F}_{\mathcal{R}}(X) := \tilde{F}(X)$ on objects. On arrows, let $\tilde{F}_{\mathcal{R}}(|G|_{\mathcal{R}}) := \tilde{F}(|G|_{\mathbb{H}})$. For this to be well-defined, it must not depend on the choice of representative $G \in |G|_{\mathcal{R}}$. First, assume that $G \rightarrow_{\mathcal{R}} G'$ for some G' . Then, there exists a rule $L \rightarrow R \in \mathcal{R}$ and a matching $m : L \rightarrow G$ that rewrites G into G' . By deforming G and G' , we can always find edge-homeomorphic cospans $H \in |G|_{\mathbb{H}}$ and $H' \in |G'|_{\mathbb{H}}$ such that:

$$H = H_{\text{out}} \circ (L \otimes H_{\text{side}}) \circ H_{\text{in}} \quad \text{and} \quad H' = H_{\text{out}} \circ (R \otimes H_{\text{side}}) \circ H_{\text{in}}$$

where \otimes and \circ are tensor and composition of framed cospans. Since $|\cdot|_{\mathbb{H}}$ respects tensor and composition:

$$|G|_{\mathbb{H}} = |H_{\text{out}}|_{\mathbb{H}} \circ (|L|_{\mathbb{H}} \otimes |H_{\text{side}}|_{\mathbb{H}}) \circ |H_{\text{in}}|_{\mathbb{H}} \quad \text{and} \quad |G'|_{\mathbb{H}} = |H_{\text{out}}|_{\mathbb{H}} \circ (|R|_{\mathbb{H}} \otimes |H_{\text{side}}|_{\mathbb{H}}) \circ |H_{\text{in}}|_{\mathbb{H}}.$$

Then, since \tilde{F} is a strict monoidal functor, we have that:

$$\begin{aligned}
 \tilde{F}(|G|_{\mathbb{H}}) &= \tilde{F}(|H_{\text{out}}|_{\mathbb{H}}) \circ (\tilde{F}(|L|_{\mathbb{H}}) \otimes \tilde{F}(|H_{\text{side}}|_{\mathbb{H}})) \circ \tilde{F}(|H_{\text{in}}|_{\mathbb{H}}) = \\
 &\tilde{F}(|H_{\text{out}}|_{\mathbb{H}}) \circ (\tilde{F}(|R|_{\mathbb{H}}) \otimes \tilde{F}(|H_{\text{side}}|_{\mathbb{H}})) \circ \tilde{F}(|H_{\text{in}}|_{\mathbb{H}}) = \tilde{F}(|G'|_{\mathbb{H}})
 \end{aligned}$$

Since $|G|_{\mathcal{R}}$ represents the closure of G with respect to $\rightarrow_{\mathcal{R}}$, this argument can be iterated to show that for any $G, G' \in |G|_{\mathcal{R}}$, $\tilde{F}(|G|_{\mathbb{H}}) = \tilde{F}(|G'|_{\mathbb{H}})$. Thus $\tilde{F}_{\mathcal{R}}$ is well-defined. Regarding the hom-sets of $\text{FCsp}(\mathcal{R}, T)$ is quotients of hom-sets in $\text{FCsp}(\mathbb{H}, T)$, $\tilde{F}_{\mathcal{R}}$ and is unique functor induced by the universal property of quotients in **Set**. \square

The exact same argument carries through replacing FCsp with FCsp^+ everywhere. Thus, rewrite categories form the most general compact closed (or symmetric traced) categories containing the equivalence closure of a rewrite system. They form the formal basis for reasoning about all of graphical theories we introduce in the next part.

Part II

Entanglement, Graphically

Chapter 6

Quantum Information and Entanglement

Quantum information theory is the study of how data can be encoded and manipulated using microscopic systems subject to quantum effects. Over the past two decades, it has grown into a large and diverse field, with applications in security, foundations of physics, and perhaps most notably quantum computing. In this chapter, we introduce the basics of quantum mechanics, quantum information theory, and models of quantum computing.

6.1 Quantum Mechanics

This section is provided for the non-physicist to briefly introduce the basic concepts of quantum mechanics used in this dissertation. Those familiar with QM can safely skip it.

Quantum mechanics is a strange but very successful theory of the universe at small scales. Its Hilbert-space formulation has four key components, which we shall focus on in detail:

1. **States** encode all of the information about a quantum system. These are represented as normalised vectors in a (complex) Hilbert space.
2. **Observables** give us “questions” to ask about a quantum system, and provide the mathematical means to turn a state into a probability distribution over measurement outcomes. These are given as self-adjoint operators on a Hilbert space.
3. **Dynamics** describe how a state evolves over time. These are expressed as unitary operators.
4. **Compound systems** are expressed as tensor products of simpler systems.

To describe these components, we use Dirac’s *bra-ket* notation. Recall that for any vector v in a Hilbert space \mathcal{H} , there is a natural way to get a linear map $\phi_v : \mathcal{H} \rightarrow \mathbb{C}$ (i.e. a vector in the dual space \mathcal{H}^*), using the inner product defined on \mathcal{H} :

$$\phi_v(u) = \langle v|u \rangle$$

Vectors in the dual space of \mathcal{H} are used so often that we employ the following notational trick. We write a vector $v \in \mathcal{H}$ as a *ket* $|v\rangle \in \mathcal{H}$, and we write the associated linear map ϕ_v as a *bra* $\langle v|$. $\langle v|$ is a function from \mathcal{H} to \mathbb{C} , so we can apply it to a vector $|u\rangle \in \mathcal{H}$. Then, some notational magic happens:

$$\langle v| |u\rangle = \langle v|u\rangle$$

For this reason, we refer to the inner product as a *bra-ket*. We define an operation $(-)^{\dagger}$ taking bras to kets and vice-versa.

$$(|v\rangle)^{\dagger} = \langle v| \quad \text{and} \quad (\langle v|)^{\dagger} = |v\rangle$$

This operation naturally extends to linear maps:

$$L^{\dagger} |u\rangle = (\langle u| L)^{\dagger}$$

Fixing an orthonormal basis, with respect to $\langle -| - \rangle$, we can represent $|v\rangle$ as a column vector and L as a matrix. Then, $(-)^{\dagger}$ just becomes the conjugate-transpose:

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \mapsto (\bar{v}_1 \quad \bar{v}_2) \quad \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \mapsto \begin{pmatrix} \bar{a}_{11} & \bar{a}_{21} \\ \bar{a}_{12} & \bar{a}_{22} \end{pmatrix}$$

$(-)^{\dagger}$ extends to a contravariant functor $\dagger : \mathbf{Hilb}^{\text{op}} \rightarrow \mathbf{Hilb}$, giving $(\mathbf{Hilb}, \otimes, \mathbb{C})$ the structure of a \dagger -monoidal category.

A vector $|\psi\rangle \in \mathcal{H}$ is the same thing as a linear map $\mathbb{C} \rightarrow \mathcal{H}$ sending $1 \in \mathbb{C}$ to $|\psi\rangle$. We use these two notions interchangeably. As string diagrams, we represent kets as triangles with a single out-edge and bras as triangles with a single in-edge.

$$|\psi\rangle = \begin{array}{c} \triangle \\ \psi \\ \downarrow \end{array} \quad \langle\psi| = \begin{array}{c} \downarrow \\ \psi \\ \triangle \end{array}$$

In quantum mechanics, pure states are unit vectors in a Hilbert space \mathcal{H} . Crucially, the fact that \mathcal{H} is a vector space gives us a way to super-impose several quantum states to form a new state.

$$|\zeta\rangle = \sum_i \alpha_i |\psi_i\rangle$$

In this case, we say that the state $|\zeta\rangle$ is in a superposition of the states $|\psi_i\rangle$.

If $|\psi\rangle$ is a state, then $\langle\psi|$ can be thought of as a function that measures the extent to which some given state is $|\psi\rangle$. This is the essential content of the Born rule, which provides a method for turning a state and an observable into a probability distribution on measurement outcomes. An observable O is a self-adjoint ($O = O^{\dagger}$) operator from a Hilbert space \mathcal{H} to itself. If \mathcal{H} is finite-dimensional then all self-adjoint operators diagonalise:

$$O = \sum_i \alpha_i |v_i\rangle\langle v_i|$$

We can choose α_i such that each of the eigenvectors $|x_i\rangle$ are normalised, in which case we call them the *eigenstates* of O . We can therefore interpret O as a set $\{\alpha_i\}$ of *measurement outcomes* and a set

$\{|v_i\rangle\}$ of possible outcome states. Picture an experimental setup, where we have some quantum state $|\psi\rangle$ in a box and a measuring apparatus hooked to it. We dial in O as the thing we want to measure and push a button. Suppose for simplicity that O only has two possible outcomes:

$$O = 1 |v_1\rangle \langle v_1| + 2 |v_2\rangle \langle v_2|$$

When we push the button, the screen says 2. Thus, we know the second measurement outcome occurred. In that sense, the eigenvalue α_i should be thought of as a “marker” for the i -th measurement outcome. The most informative observables are the ones where all of these markers are distinct and non-zero. These are called *non-degenerate* observables. Three non-degenerate observables that play a particularly important role in quantum information and quantum computing are the Pauli spin operators:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (6.1)$$

Furthermore, upon getting outcome 2, we know that the state $|\psi\rangle$ must be $|v_2\rangle$. Suppose beforehand that we had prepared $|\psi\rangle$ in some superposition of $|v_1\rangle$ and $|v_2\rangle$. As soon as we measured $|\psi\rangle$, this superposition *collapsed* to a single state $|v_2\rangle$. This phenomenon is known as the collapse of the quantum state, or “collapse of the wavefunction”. What this means physically is a question of interpretation, but mathematically it means after the measurement occurs, we can treat the quantum system as if it is in the state $|v_2\rangle$.

We compute the probability of getting outcome i using the *Born rule*.

$$\text{Prob}(i, |\psi\rangle) = |\langle v_i | \psi \rangle|^2 = \langle \psi | v_i \rangle \langle v_i | \psi \rangle$$

The key point here is that the Born rule is a function of the inner product. Because of the role it plays in measurement, we sometimes refer to elements of the dual space $\langle \psi | \in \mathcal{H}^*$ as *effects*.

Since $|\psi\rangle$ is normalised, the sum of the probabilities of all outcomes is 1, so $\text{Prob}(i, |\psi\rangle)$ is a probability distribution. These probabilities provide our only access to the “real” quantum state, so we consider two states to be equal if they give the same probability distributions with respect to any observable. However, the Born rule yields the same probabilities for states $|\psi\rangle$ and $e^{i\theta} |\psi\rangle$.

$$\overline{e^{i\theta}} e^{i\theta} \langle \psi | v_i \rangle \langle v_i | \psi \rangle = e^{-i\theta} e^{i\theta} \langle \psi | v_i \rangle \langle v_i | \psi \rangle = \langle v_i | \psi \rangle \langle \psi | v_i \rangle$$

The scalar factor $e^{i\theta}$ is called a *global phase*. We always identify states (and hence operators) differing only by a global phase.

The main reason the Pauli matrices are so interesting is that every distinct pair of them is complementary. Two observables are called *complementary* if their associated bases of eigenstates are equally-far apart. Let O, O' be observables in a D -dimensional Hilbert space with eigenstates $\{|v_i\rangle\}$

and $\{|v'_j\rangle\}$. Then O and O' are called complementary if their bases are *mutually unbiased*. That is, they satisfy the following equation for all i, j :

$$|\langle v_i | v'_j \rangle|^2 = \frac{1}{D} \quad (6.2)$$

We can interpret this definition using the Born rule. If a state $|\psi\rangle$ is in the i -th eigenstate of O , then measuring O will obtain outcome i with *certainty*. So, if we know $|\psi\rangle$ is in an eigenstate of O , we have maximal knowledge about the O observable. However, if we measure $|\psi\rangle$ with respect to the O' observable, we are equally likely to get any outcome. So, maximal knowledge about O implies minimal knowledge about O' .

Unlike measurements which cause a state to collapse, dynamic evolution of quantum states is always reversible. We can evolve a quantum state in time by applying a unitary ($U^\dagger = U^{-1}$) operator. One should interpret the dagger of a unitary operator as that some operator, “done backwards”. To be a bit more explicit, suppose we represent the evolution of a state according to the Schrödinger equation, for a self-adjoint operator H called a *Hamiltonian*.

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H |\psi(t)\rangle \quad (6.3)$$

If H does not depend on t (e.g. if the forces acting on a particle are constant), then the value of a solution at time t has a simple expression in terms of the value at some initial time 0.

$$|\psi(t)\rangle = e^{-(i/\hbar)tH} |\psi(0)\rangle$$

Letting $U(t) = e^{-(i/\hbar)tH}$, then since H is self-adjoint, $U(t)$ is unitary for all t and describes the evolution of $|\psi(0)\rangle$ under H for time t . $U(-t)$ then corresponds to the same evolution, but with time running backwards. Since $U(t)U(-t) = U(-t)U(t) = U(0) = 1_{\mathcal{H}}$, it must be the case that $U(t)^\dagger = U(-t)$.

6.2 Compound Systems and Entanglement

Suppose we have a particle in the state $|\psi\rangle$ sitting in some potential well and another particle $|\phi\rangle$ sitting in another one far away. We use the tensor product to “pair up” the states of the two particles. That is, the overall state of the system is $|\psi\rangle \otimes |\phi\rangle$. But, since this is a quantum state, it could be in some superposition:

$$\frac{1}{\sqrt{2}} (|\psi_1\rangle \otimes |\phi_1\rangle + |\psi_2\rangle \otimes |\phi_2\rangle)$$

Suppose one state is in a superposition $|\psi_1\rangle + |\psi_2\rangle$. Then the combined state is also in a superposition:

$$(|\psi_1\rangle + |\psi_2\rangle) \otimes |\phi\rangle = (|\psi_1\rangle \otimes |\phi\rangle) + (|\psi_2\rangle \otimes |\phi\rangle)$$

This bilinearity justifies our use of the tensor product, since \otimes provides the most general bilinear pairing for two spaces. Some states in the tensor product space $\mathcal{H}_1 \otimes \mathcal{H}_2$ can be written in the form $|\psi\rangle \otimes |\phi\rangle$ for $|\psi\rangle \in \mathcal{H}_1$ and $|\phi\rangle \in \mathcal{H}_2$. Such states are called *separable* or *product states*. However the vast majority of states $|\Psi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$ cannot be written this way. These are called *entangled states*.

A measurement on one subsystem of an entangled state collapses the entire state, thus affecting the other subsystem. For instance, suppose we have some observable

$$O = 1 |v_1\rangle \langle v_1| + 2 |v_2\rangle \langle v_2|$$

as before, and an entangled state

$$|\Psi\rangle = \frac{1}{\sqrt{2}} (|v_1\rangle \otimes |v_1\rangle + |v_2\rangle \otimes |v_2\rangle)$$

and we measure O on the left subsystem, getting outcome 1. The whole state is now:

$$|\Psi'\rangle = |v_1\rangle \otimes |v_1\rangle$$

If $|\Psi\rangle$ were a product state, then the second system would be unaffected, so entanglement is the crucial property that allows such correlations at a distance.

Entanglement is also a source of computational complexity for many-body systems. For finite dimensional Hilbert spaces, the dimension of the tensor product of two spaces is the product of the dimensions of each space.

$$\dim(\mathcal{H}_1 \otimes \mathcal{H}_2) = \dim(\mathcal{H}_1) \dim(\mathcal{H}_2)$$

So, the dimension of a compound system increases *exponentially* with the number of subsystems.

$$\dim(\underbrace{\mathcal{H} \otimes \dots \otimes \mathcal{H}}_N) = \dim(\mathcal{H})^N$$

Computing with such states quickly becomes untenable, even for low-dimensional \mathcal{H} , which is the main reason for trying to understand quantum phenomena like entanglement from a more structural level.

6.3 Mixed State Quantum Mechanics

Often it is more convenient to work with probabilistic mixtures of quantum states, rather than states that are totally determined. This is because nearly all procedures for preparing a quantum state in a lab only succeed with some probability. A set of quantum pure states along with associated probabilities $\{(|\psi_i\rangle, p_i)\}$ is called an *ensemble*. We can compute the probability of getting a particular measurement outcome on an ensemble using the Born rule, adjusting for probabilities.

$$\text{Prob}(i, \{p_j, |\psi_j\rangle\}) = \sum_j p_j |\langle v_i | \psi_j \rangle|^2 = \sum_j p_j \langle v_i | \psi_j \rangle \langle \psi_j | v_i \rangle = \langle v_i | \left(\sum_j p_j |\psi_j\rangle \langle \psi_j| \right) | v_i \rangle \quad (6.4)$$

As in the case with pure states, we consider two ensembles to represent the same “state” if they produce identical probability distributions for all observables under the Born rule. From (6.4), we can see that two ensembles produce the same probability distributions if and only if:

$$\sum_i p_i |\psi_i\rangle\langle\psi_i| = \sum_i q_i |\phi_i\rangle\langle\phi_i|$$

For normalised vectors $|\psi_i\rangle$ and $\sum_i p_i = 1$, this is the general form for a trace-1 positive operator. For that reason, we call trace-1 positive operators *mixed states*. Because they encode probability densities, they are also sometimes called *density matrices*. Pure states can be represented as density matrices of the form $|\psi\rangle\langle\psi|$.

Just as states cannot in practice be prepared with certainty, so too is the case for quantum evolutions and measurements. The mixed version of a unitary evolution is a completely positive map (CPM). For a finite-dimensional Hilbert space \mathcal{H} , let $\mathcal{L}(\mathcal{H})$ be the vector space of linear maps $\mathcal{H} \rightarrow \mathcal{H}$. CPMs are just linear maps $\Phi : \mathcal{L}(\mathcal{H}) \rightarrow \mathcal{L}(\mathcal{H}')$ that take positive operators to positive operators.

Outcomes for pure measurements span an orthonormal basis. A particular observable O corresponds to the decomposition of the identity into 1-dimensional projectors corresponding to each outcome for O .

$$1_{\mathcal{H}} = \sum_i |v_i\rangle\langle v_i|$$

The probability of getting a *particular* measurement outcome on a mixed state can be computed as in 6.4 by tracing the composition of the projector and the state’s density matrix.

$$\text{Prob}(i, \rho) = \langle v_i | \rho | v_i \rangle = \text{Tr}(\langle v_i | \rho | v_i \rangle) = \text{Tr}(\rho | v_i \rangle \langle v_i |)$$

For that reason, pure measurements are often referred to as *projective measurements*. The mixed version of a projective measurement is a *positive operator-valued map* (POVM). In finite dimensions, this is just a set of positive operators $P_i \in \mathcal{L}(\mathcal{H})$ that sum to the identity. As in the projective case, probabilities are computed by tracing the composition of the positive operator and the state’s density matrix.

$$\text{Prob}(i, \rho) = \text{Tr}(\rho P_i)$$

For the majority of this dissertation, we will only need concepts from pure-state quantum mechanics. However, when we look at multipartite entanglement in chapter 8, it will occasionally be useful to ignore a subsystem of an entangled pure state. This can be done probabilistically by tracing out that subsystem and renormalising (if necessary). This is called the *reduced density matrix* of an entangled state. For a state $|\Psi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$ and $\rho_{12} := |\Psi\rangle\langle\Psi|$, we can ignore the system \mathcal{H}_2 by tracing it out.

$$\rho_1 = \text{Tr}^{\mathcal{H}_2}(\rho_{12})$$

6.4 Quantum Computation

Quantum computation refers to the encoding of data into quantum states and the use of evolution and measurements to perform computations on that data. As in programming language design, quantum computation can be carried out using one of a variety of paradigms. We shall introduce two of them here: the *circuit model* and *measurement-based quantum computation* (MBQC).

For a D -dimensional Hilbert space, we often fix an orthonormal basis $|0\rangle, |1\rangle, \dots, |D-1\rangle$ called the *computational basis*. The two-dimensional Hilbert space \mathbb{C}^2 plays a special role in quantum computation, and is called the space of quantum bits, or *qubits*. The basis vectors $|0\rangle, |1\rangle \in \mathbb{C}^2$ can be thought of as classical bits, embedded in the bigger space of qubits. We also introduce a special notation for tensor products of basis vectors using bit strings:

$$|00101110\rangle = |0\rangle \otimes |0\rangle \otimes |1\rangle \otimes |0\rangle \otimes |1\rangle \otimes |1\rangle \otimes |1\rangle \otimes |0\rangle$$

In \mathbb{C}^2 the space of qubits, the basis $|0\rangle, |1\rangle$ corresponds to the eigenbasis of the Z observable. We use the term “measuring in the computational basis” to mean performing a measurement with respect to Z . We also define the X basis $|+\rangle, |-\rangle$ and the Y basis $|i\rangle, |-i\rangle$ as follows.

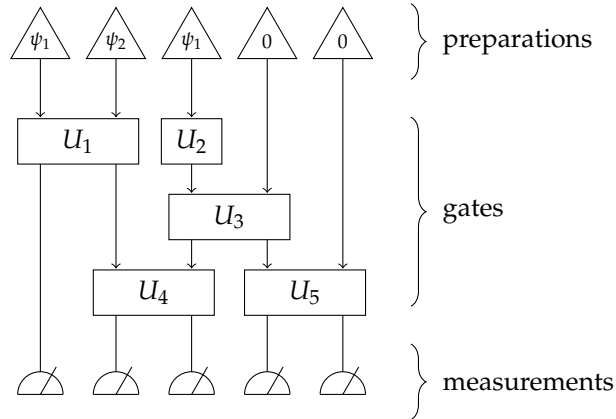
$$\begin{aligned} |+\rangle &:= \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) & |i\rangle &:= \frac{1}{\sqrt{2}} (|0\rangle + i|1\rangle) \\ |-\rangle &:= \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) & |-i\rangle &:= \frac{1}{\sqrt{2}} (|0\rangle - i|1\rangle) \end{aligned}$$

6.4.1 The Circuit Model

In the circuit model, quantum computation proceeds in three steps:

1. Prepare an N -qubit quantum state (usually a product state). Some of these qubits are treated as inputs and others simply as “helper” qubits called ancillas (which are usually initialised to $|0\rangle$).
2. Evolve the prepared state using small (usually 1- or 2-qubit), fixed-time evolutions called quantum gates.
3. Measure some or all of the qubits, yielding the result of the computation. Unmeasured qubits are sometimes treated as outputs.

Graphically, we represent a circuit evaluation as a string diagram.



Usually measurements are performed over the computational basis. That is, we measure the Pauli Z observable given in equation (6.1). Using the Pauli operators as the Hamiltonian in the Schrödinger equation (6.3), we can produce unitary evolutions we call the *phase gates*.

$$X_\theta = e^{-(i\theta/2)X} \quad Y_\theta = e^{-(i\theta/2)Y} \quad Z_\theta = e^{-(i\theta/2)Z}$$

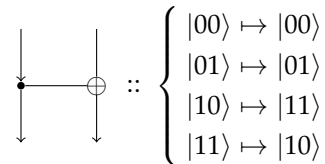
Up to a global phase, we can recover identities and the Pauli gates themselves as phase gates.

$$X_0 = Y_0 = Z_0 = 1_{\mathbb{C}^2} \quad X_\pi = X \quad Y_\pi = Y \quad Z_\pi = Z$$

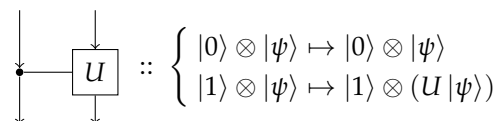
It is a well known fact that *any* single-qubit unitary can be constructed from phase gates as $U = Z_\alpha X_\beta Z_\gamma$. This is called the *Euler decomposition* of the unitary. An important single-qubit gate that is not a phase gate is the Hadamard gate H .

$$H = Z_{-\pi/2} X_{-\pi/2} Z_{-\pi/2} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

H interchanges the eigenbasis $|0\rangle, |1\rangle$ of Z with the eigenbasis $|+\rangle, |-\rangle$ of X . A simple consequence is that $HZ_\theta H = X_\theta$. Perhaps the most common 2-qubit gate is the *controlled-NOT* or CNOT gate.



It is called the controlled-NOT gate because the first qubit controls whether the second qubit has a NOT (a.k.a. X) gate applied to it. A generalisation of CNOT gates are controlled-unitary gates, which conditionally apply a unitary to the second qubit.



These are examples of gates that can take product states to entangled states. As such, they are sometimes referred to as *entangling gates*. For example, applying a CNOT gate to the state

$|+\rangle \otimes |0\rangle$ yields an entangled state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ called the *Bell state*. Examples of gates that are not entangling gates are tensor products of 1-qubit gates, such as $X \otimes Z$. These will *always* take product states to product states.

Just as AND, OR, and NOT gates can be used to construct arbitrary classical circuits, we have a notion for a set of quantum gates being able to construct arbitrary quantum circuits. We say a set of gates is *universal for quantum computation* if any N -qubit unitary map can be constructed from compositions of those gates.

Theorem 6.4.1 ([7]). *The gates Z_θ , H , and CNOT are universal for quantum computation.*

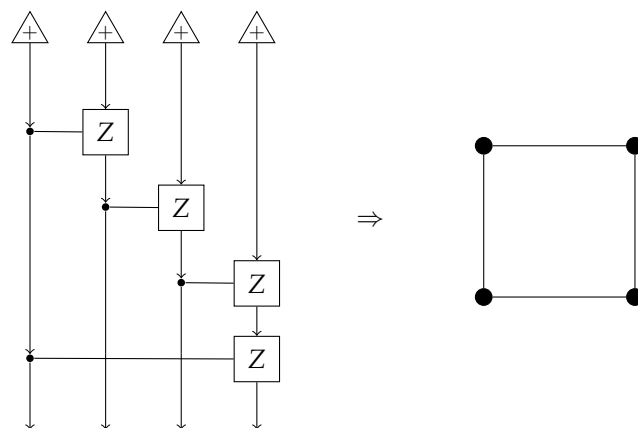
Many important quantum algorithms, such as the quantum Fourier transform, Shor’s factoring algorithm, and Grover’s search algorithm, can be presented in the circuit model.

6.4.2 Measurement-based Quantum Computation

Measurement based quantum computation (MBQC), which is sometimes called one-way quantum computation, provides a different, equally powerful paradigm for quantum computation. For comparison to the previous section, we can organise the MBQC procedure into three steps.

1. Prepare a known, highly-entangled state called a *graph state*. This graph state may be entangled to some qubits in an unknown state called input qubits.
2. Perform measurements at arbitrary angles, where the choice of angles can depend on previous measurement outcomes.
3. Optionally, perform single-qubit corrections on unmeasured, output qubits.

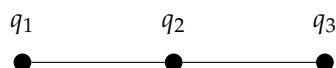
Graph states are constructed by preparing a collection of (non-input) qubits in the $|+\rangle$ state, then applying controlled- Z gates to pairs of qubits to introduce entanglement. We represent such a state by drawing a vertex for every qubit and an edge whenever a controlled- Z gate is applied.



This representation is unambiguous because controlled-Z gates are symmetric and commute past each other. We define generalised measurements, or *measurements with angles* as follows. An X_θ measurement on single qubit consists of first applying the unitary gate X_θ then measuring the X observable. A Z_θ measurement consists of first applying the unitary gate Z_θ then measuring the Z observable. Since they are measurements, neither of these operations is deterministic.

Each measurement has two possible outcomes: a “desired” outcome, and an “erroneous” outcome. The latter can be thought of as the quantum version of an “accidental bit flip” during the course of the computation. The key to MBQC is that if we choose our measurement angles wisely, we can *correct* these errors as we go. If there are no measurements left to perform, we complete the calculation by applying any remaining corrections as single-qubit unitaries on the output qubits. This is known as *feeding forward* corrections. There are several techniques for identifying and using graph states for deterministic MBQC, such as identifying a *generalised flow* for the graph [12]. To give a feel for how feed-forward works, we provide a simple example.

Example 6.4.2. Prepare the following graph state, where q_1 and q_3 are in an unknown state (i.e. they are inputs) and q_2 is prepared in the $|+\rangle$ state.



First measure q_3 in Z , getting outcome $i \in \{0, 1\}$. Letting $Z_0 = 1_{C^2}$ and $Z_1 = Z$, apply Z_i to q_1 and apply $Z_i H$ to q_2 . Treating q_1 and q_2 as outputs, this procedure computes the CNOT of q_1 and q_3 , regardless of the outcome of the measurement of q_3 . As Z_i depends on the outcome of measuring q_3 , this is an example of feeding-forward a measurement outcome to a correction.

This may seem like an excessively roundabout way to apply a CNOT gate, especially since a naïve implementation involves applying 2 controlled-Z gates to prepare the graph state. However, if one assumes we have a stock of suitably nice graph states, we can actually perform *arbitrary* quantum computations using (comparatively easy) single qubit unitaries and measurements.

Chapter 7

Categorical Quantum Mechanics

Categorical quantum mechanics (CQM) refers to a broad program initiated by Abramsky and Coecke in 2004 [3] that emphasises the abstract, categorical, and compositional aspects of quantum mechanics. The core thesis is that the structure-rich setting of Hilbert spaces and linear operators obscures the causes of many quantum phenomena. Therefore reasoning in the comparatively sparse context of an arbitrary monoidal category yields practical and foundational insights that were previously hidden. Depending on the aims of a particular project under the umbrella of CQM, this proscription of Hilbert spaces can be taken literally, as in Paquette’s PhD thesis [57] for example, in order to obtain structural results about quantum theory that are independent of Hilbert space formulation. This route could prove fruitful in light of doubts that the Hilbert space formalism is the “correct” way to think about quantum mechanics, as expressed in these now-infamous words of its progenitor.

I would like to make a confession which may seem immoral: I do not believe absolutely in Hilbert space any more. [54, John von Neumann (1935)]

Alternatively, one can take a less “hard-line” approach by using categorical techniques to complement and expand upon concrete results based on Hilbert spaces. We adopt this approach in the sections to come.

This chapter offers an introduction to categorical quantum mechanics and a handful of illustrative examples. We employ notions from CQM to show how complementary observables can be studied as interacting Frobenius algebras and offer several new results about special types of complementary observables called strongly complementary observables. Most notably, we give a classification theorem for strongly complementary pairs of observables and show that a set of pairwise strongly complementary observables must contain no more than 2 distinct observables.

7.1 Compact Categories and Teleportation

Quantum protocols exploit the unique features of quantum mechanics (typically entanglement and superposition) to perform a task that would be difficult or impossible classically. The canonical example of a quantum protocol is quantum teleportation, whereby one party (called Alice) can transmit an arbitrary quantum state to a second party (called Bob) using only a shared entangled state and a classical data channel. To start, Alice has a quantum state $|\psi\rangle$ and Alice and Bob share a Bell pair. That is, a pair of qubits in the state $|00\rangle + |11\rangle$ (ignoring normalisation factors).

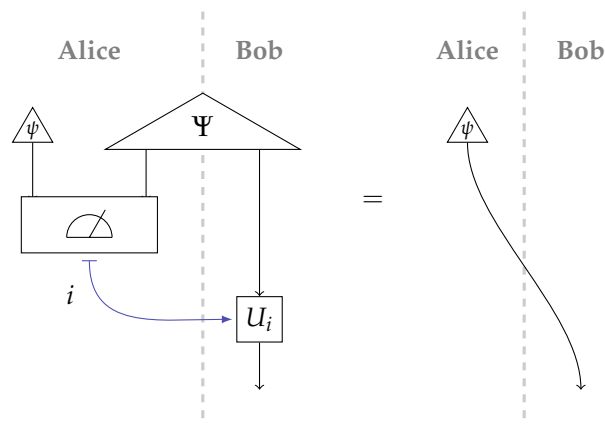
Alice performs an entangled, 2-qubit measurement called a *Bell measurement* on $|\psi\rangle$ and her half of the Bell pair. A Bell measurement consists of measuring the two qubits in the Bell basis:

$$|\Psi_0\rangle = |00\rangle + |11\rangle \quad |\Psi_1\rangle = |00\rangle - |11\rangle \quad |\Psi_2\rangle = |01\rangle + |10\rangle \quad |\Psi_3\rangle = |01\rangle - |10\rangle$$

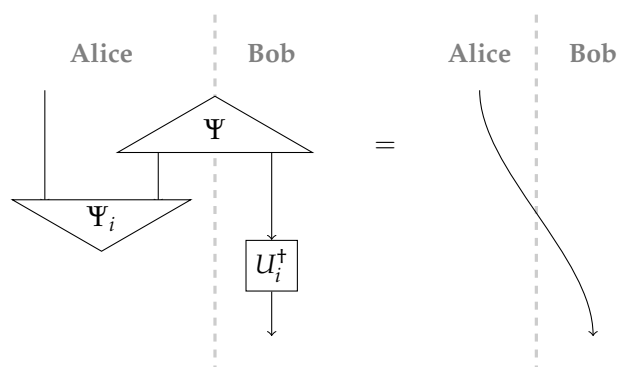
She gets an outcome $i \in \{0,1,2,3\}$, which she then sends to Bob. Bob then applies a unitary correction to his half of the Bell pair, based on i :

$$U_0 = 1_{\mathbb{C}^2} \quad U_1 = Z \quad U_2 = X \quad U_3 = XZ$$

Once this is done, Bob's qubit will be in the state $|\psi\rangle$, i.e. the state of $|\psi\rangle$ has been teleported to Bob. We can represent this protocol in circuit language:



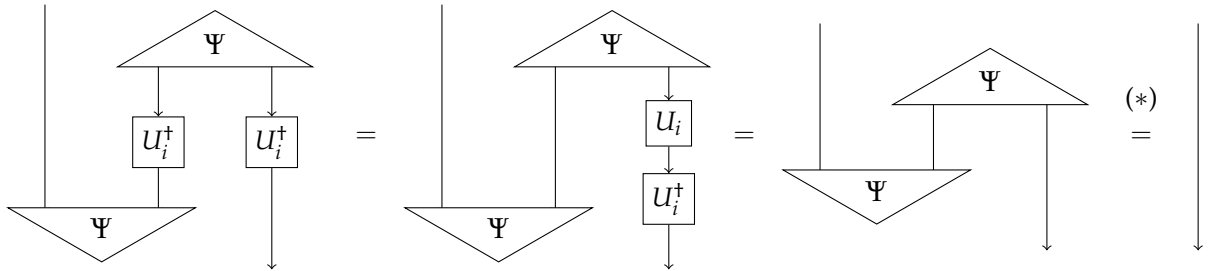
This protocol works because, by performing a Bell basis measurement, Alice projects out her two qubits using the associated bra $\langle\Psi_i|$.



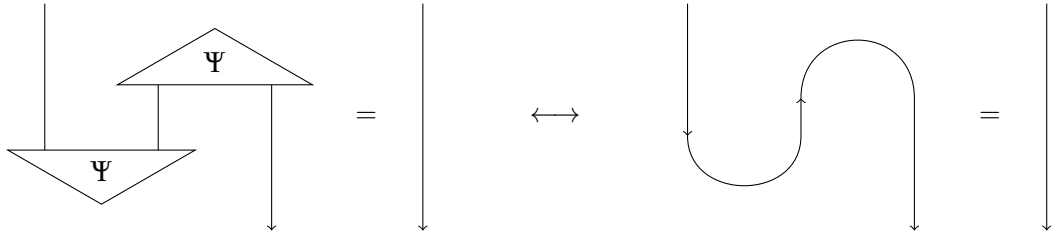
We can express all four elements of the Bell basis in terms of the Bell state and the corrections we defined before.¹

$$|\Psi_i\rangle = (1 \otimes U_i) |\Psi_0\rangle = (U_i^\dagger \otimes 1) |\Psi_0\rangle \quad \langle\Psi_i| = \langle\Psi_0| (1 \otimes U_i^\dagger) = \langle\Psi_0| (U_i \otimes 1)$$

Thus, we can pull the U_i out of the measurement all the way to the end, and prove the teleportation protocol works for all i .



The crucial step is (*). This identity should look familiar. Teleportation, like many quantum protocols exploits the fact that **FHilb** is compact-closed and finite-dimensional Hilbert spaces are all self-dual ($X \cong X^*$).



Using this insight, one can perform teleportation in any self-dual compact-closed category, including **Rel**, **Mat**(\mathbb{R}), and (perhaps surprisingly) **Spek**, the category Rob Spekkens' toy theory. The last example is surprising, because **Spek** can be defined using a local hidden variable model. Thus teleportation succeeds even in the absence of non-locality for a physical theory. For more details, see [18, 17].

7.2 Complementary Observables as Frobenius Algebras

The eigenstates of an observable play a key role in quantum mechanics. They form the set of possible outcome states one obtains by performing a measurement. Classical data is obtained from a quantum system via measurements, so an orthonormal basis of measurement outcomes can be thought of as a particular *classical context* embedded in the overall quantum state space. In studying the interaction of multiple classical contexts (especially complementary ones), we can see the

¹Note that applying U_i to the right qubit of $|\Psi\rangle$ has the same affect as applying U_i^\dagger on the left qubit because all of the maps U_i , written as matrices over the computational basis, have real entries. A more generalised scheme is provided in [3], replacing $(-)^{\dagger}$ with $(-)^{\circ\text{T}}$.

unique features of quantum mechanics. The question is, can we study the concept of a “basis” over an object in an arbitrary \dagger -compact category?

Recall in example 3.2.24, we used a basis of a vector space to construct a special commutative Frobenius algebra.

$$\delta :: e_i \mapsto e_i \otimes e_i \qquad \epsilon :: e_i \mapsto 1 \qquad \mu :: e_i \otimes e_i \mapsto e_i \qquad \eta :: \sum e_i$$

It was also noted that *all* SCFAs over an algebraically closed field are of this form. So, there is a one-to-one correlation between SCFAs and arbitrary bases. However, projective measurements like the ones we have described have outcomes in an *orthonormal* basis. Orthonormal bases can be captured in a \dagger -compact closed category using \dagger -special commutative Frobenius algebras.

Definition 7.2.1. A \dagger -special commutative Frobenius algebra, or \dagger -SCFA, $(A, \delta^\dagger, \epsilon^\dagger, \delta, \epsilon)$ is a \dagger -Frobenius algebra such that $\delta^\dagger \delta = 1_A$.

In [21] Coecke, Pavlovic, and Vicary showed that \dagger -SCFAs in **FHilb** are in one-to-one correspondence with orthonormal bases. So, for any orthonormal basis $\{|i\rangle\}$ in a finite-dimensional Hilbert space, there exists a unique \dagger -SCFA whose comultiplication copies the basis vectors and whose counit deletes them.

$$\delta :: |i\rangle \mapsto |ii\rangle \qquad \epsilon :: |i\rangle \mapsto 1$$

The basis vectors $|i\rangle$ are called the *classical points* of δ . This respects the no-cloning principal in quantum mechanics, because δ cannot copy any arbitrary state, only those in $\{|i\rangle\}$. In fact, one can prove that these are the *only* vectors copied by δ and deleted by ϵ , so a basis can always be recovered from a \dagger -SCFA by taking the set of classical points. For an observables O and O' , let the associated \dagger -SCFAs be:

$$\delta_O := \begin{array}{c} \downarrow \\ \circ \\ \swarrow \searrow \end{array} \qquad \epsilon_O := \begin{array}{c} \circ \\ \downarrow \end{array} \qquad \delta_{O'} := \begin{array}{c} \downarrow \\ \circ \\ \swarrow \searrow \end{array} \qquad \epsilon_{O'} := \begin{array}{c} \circ \\ \downarrow \end{array}$$

Then, we write classical points as triangles of the same colour, and their associated bras as upside-down triangles of that colour.

$$\begin{array}{cccc} \begin{array}{c} \triangle_i \\ \downarrow \\ \circ \\ \swarrow \searrow \end{array} = \begin{array}{c} \triangle_i \quad \triangle_i \\ \downarrow \quad \downarrow \end{array} & \begin{array}{c} \triangle_i \\ \downarrow \\ \circ \end{array} = 1 & \begin{array}{c} \downarrow \\ \circ \\ \swarrow \searrow \\ \triangle_i \end{array} = \begin{array}{c} \downarrow \quad \downarrow \\ \triangle_i \quad \triangle_i \end{array} & \begin{array}{c} \circ \\ \downarrow \\ \triangle_i \end{array} = 1 \\ \begin{array}{c} \triangle_i \\ \downarrow \\ \circ \\ \swarrow \searrow \end{array} = \begin{array}{c} \triangle_i \quad \triangle_i \\ \downarrow \quad \downarrow \end{array} & \begin{array}{c} \triangle_i \\ \downarrow \\ \bullet \end{array} = 1 & \begin{array}{c} \downarrow \\ \circ \\ \swarrow \searrow \\ \triangle_i \end{array} = \begin{array}{c} \downarrow \quad \downarrow \\ \triangle_i \quad \triangle_i \end{array} & \begin{array}{c} \bullet \\ \downarrow \\ \triangle_i \end{array} = 1 \end{array}$$

Two observables are complementary if their bases of eigenstates are mutually unbiased. That is, for any i, j , $|\langle v_i | v'_j \rangle|^2 = 1/D$. In the graphical notation:

$$\begin{array}{c} \triangleup_i \\ \downarrow \\ \triangleleft_j \end{array} \begin{array}{c} \triangleup_j \\ \downarrow \\ \triangleleft_i \end{array} = \frac{1}{D}$$

A question posed by Coecke and Duncan [15] was, “Can we represent complementarity purely in terms of interacting Frobenius algebras?” It turns out that complementarity is equivalent to a simple diagrammatic identity between two \dagger -SCFAs. First, we can move $1/D$ in the above equation to the other side and express it as a circle, as the trace of the identity always equals D . Then, replace 1 on the RHS with “deleted points”.

$$\bigcirc = \begin{array}{c} \triangleup_i \\ \downarrow \\ \triangleleft_j \end{array} \begin{array}{c} \bullet \\ \downarrow \\ \triangleleft_j \end{array} \quad (7.1)$$

Frobenius algebras fix an isomorphism of a space with its dual space. In the case of \dagger -SCFAs, this isomorphism takes a classical point to its adjoint: $|v_i\rangle^{\circ T} = \langle v_i|$ and $|v'_j\rangle^{\circ T} = \langle v'_j|$. Graphically:

$$\begin{array}{c} \triangleup_i \\ \downarrow \\ \bigcirc \end{array} = \begin{array}{c} \downarrow \\ \triangleleft_i \end{array} \quad \begin{array}{c} \bullet \\ \downarrow \\ \triangleleft_j \end{array} = \begin{array}{c} \triangleup_j \\ \downarrow \end{array}$$

We can simplify the LHS of equation (7.1) using this fact.

$$\bigcirc = \begin{array}{c} \triangleup_i \\ \downarrow \\ \triangleleft_j \end{array} \begin{array}{c} \triangleup_j \\ \downarrow \\ \triangleleft_i \end{array} = \begin{array}{c} \triangleup_i \\ \downarrow \\ \bullet \end{array} \begin{array}{c} \triangleup_i \\ \downarrow \\ \triangleleft_j \end{array} = \begin{array}{c} \triangleup_i \\ \downarrow \\ \triangleleft_j \end{array} \begin{array}{c} \bullet \\ \downarrow \\ \triangleleft_j \end{array} = \begin{array}{c} \triangleup_i \\ \downarrow \\ \bullet \end{array} \begin{array}{c} \bullet \\ \downarrow \\ \triangleleft_j \end{array} \stackrel{(*)}{=} \begin{array}{c} \triangleup_i \\ \downarrow \\ \bullet \end{array} \begin{array}{c} \triangleup_i \\ \downarrow \\ \bullet \end{array} \begin{array}{c} \bullet \\ \downarrow \\ \triangleleft_j \end{array}$$

The equation (*) is due to two applications of the spider theorem to merge the grey and white vertices, leaving a single edge connecting grey to white. Plugging this back into equation (7.1), we get:

$$\bigcirc = \begin{array}{c} \triangleup_i \\ \downarrow \\ \bullet \end{array} \begin{array}{c} \triangleup_i \\ \downarrow \\ \bullet \end{array} \begin{array}{c} \bullet \\ \downarrow \\ \triangleleft_j \end{array} = \begin{array}{c} \triangleup_i \\ \downarrow \\ \bullet \end{array} \begin{array}{c} \bullet \\ \downarrow \\ \triangleleft_j \end{array}$$

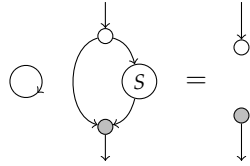
Since this equation holds for all i, j and the classical points span the entire space, we can conclude that a more general identity holds:

$$\bigcirc = \begin{array}{c} \downarrow \\ \bullet \end{array} \begin{array}{c} \downarrow \\ \bullet \end{array} \quad (7.2)$$

Suppose we define a map S , serving as an antipode:

$$\begin{array}{c} \downarrow \\ \textcircled{S} \end{array} = \begin{array}{c} \bullet \\ \downarrow \\ \bullet \end{array} \begin{array}{c} \downarrow \\ \bullet \end{array} \quad (7.3)$$

Then, equation (7.2) resembles the equation found in Definition 3.1.4 of a Hopf algebra.

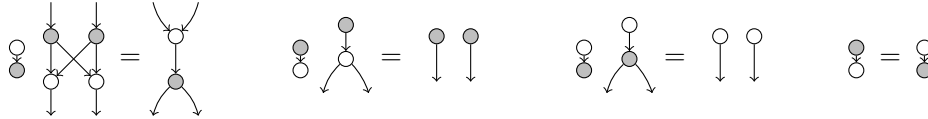


For that reason, we refer to (7.2) as the *Hopf law*.

Theorem 7.2.2. *Two \dagger -SCFAs correspond to complementary observables if and only if they satisfy the Hopf law.*

All complementary observables satisfy the Hopf law. When a complementary pair of observables actually extends to a (scaled) Hopf algebra, we call them *strongly complementary*.

Definition 7.2.3. Two observables O and O' are called *strongly complementary* if their associated \dagger -SCFAs satisfy the following equations, called the *scaled bialgebra equations*.

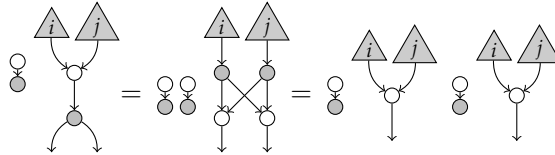


Note that we have only required that $(A, \mu_O, \eta_O, \delta_{O'}, \epsilon_{O'})$ be a *bialgebra*, up to scalar factors. However, we can show that bialgebras consisting of \dagger -SCFAs automatically satisfy equation (7.2), so they are Hopf algebras. Before we can show this, we need a couple of lemmas. For the remainder of the section, let $O = (\downarrow, \circlearrowleft, \circlearrowright, \circlearrowup)$ and $O' = (\downarrow, \circlearrowleft, \circlearrowright, \circlearrowup)$ be strongly complementary \dagger -SCFAs in **FHilb**.

Lemma 7.2.4. *Up to a scalar, \circlearrowleft is a monoid over the classical points of \circlearrowright . For all i, j , the following are classical points for O' :*

$$\begin{array}{c} \triangle_{i \cdot j} \\ \downarrow \end{array} = \begin{array}{c} \triangle_i \quad \triangle_j \\ \downarrow \end{array} \quad \begin{array}{c} \triangle_e \\ \downarrow \end{array} = \begin{array}{c} \circ \\ \downarrow \end{array} \tag{7.4}$$

Proof. We can show that the point labelled $i \cdot j$ is copied using the first bialgebra rule.



Deletion follows from the dagger of the second bialgebra rule.

$$\begin{array}{c} \triangle_i \quad \triangle_j \\ \downarrow \end{array} = \begin{array}{c} \triangle_i \quad \triangle_j \\ \downarrow \end{array} = 1$$

We can apply the bialgebra:

Since the scalar $\downarrow \circ \uparrow$ is non-zero, e is a classical point. □

This is the property that Coecke and Duncan refer to as *closure*.

Lemma 7.2.5. For a strongly complementary pair of observables, $(\downarrow \circ \uparrow)^{\circ T} = \downarrow$ and $(\uparrow \circ \downarrow)^{\circ T} = \uparrow$.

Proof. We can use the previous lemma to evaluate over classical points for \downarrow . For multiplication:

And for unit:

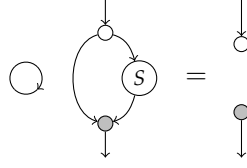
A simple consequence of this lemma is:

Lemma 7.2.6. The antipode map S defined in figure (7.3) is self-adjoint and is an automorphism for both Frobenius algebras.

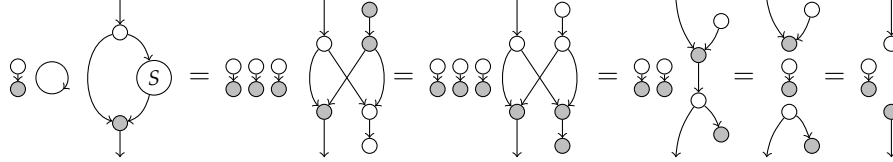
Proof. For S to be self-adjoint it suffices to show we can interchange the caps and cups.

To show S is a Frobenius algebra automorphism, we can use the previous identity and the fact that it is copied by \downarrow .

Theorem 7.2.7. A strongly complementary pair of observables forms a scaled Hopf algebra with antipode S .



Proof. The proof follows straightforwardly from the bialgebra identities and Lemmas 7.2.5 and 7.2.6.



□

Using the results from this section, we can prove a stronger classification result than the ones given in [16] for strongly complementary observables in **FHilb**.

Theorem 7.2.8. *Let (G, \cdot, e) be a finite Abelian group of order D , and $\{|g\rangle : g \in G\}$ be a D -dimensional orthonormal basis. Every strongly complementary pair O, O' of \dagger -SCFAs is of the following form.*

$$\begin{aligned} \delta_O &:: |g\rangle \mapsto |g\rangle \otimes |g\rangle & \epsilon_O &:: |g\rangle \mapsto 1 \\ \mu_{O'} &:: |g\rangle \otimes |h\rangle \mapsto \frac{1}{\sqrt{D}} |g \cdot h\rangle & \eta_{O'} &:: 1 \mapsto \sqrt{D} |e\rangle \end{aligned}$$

Proof. First, we show this is indeed a strongly complementary pair. (δ_O, ϵ_O) copies and deletes an orthonormal basis, so it extends to a \dagger -SCFA. Also, up to a scalar, $(\mu_{O'}, \eta_{O'}, \mu_{O'}^\dagger, \eta_{O'}^\dagger)$ is the induced Frobenius algebra of the group algebra $\mathbb{C}[G]$. It is a routine calculation to show that the factors of $1/\sqrt{D}$ and \sqrt{D} cancel out where necessary in the monoid, comonoid, and Frobenius identities. We can give explicit forms for $\delta_{O'} = \mu_{O'}^\dagger$ and $\epsilon_{O'} = \eta_{O'}^\dagger$.

$$\delta_{O'} :: |g\rangle \mapsto \frac{1}{\sqrt{D}} \sum_{g_1 \cdot g_2 = g} |g_1\rangle \otimes |g_2\rangle \quad \epsilon_{O'} = \sqrt{D} \langle e|$$

To show specialness, evaluate $\mu \circ \delta$ for any $g \in G$:

$$\mu_{O'} \delta_{O'} |g\rangle = \mu_{O'} \left(\frac{1}{\sqrt{D}} \sum_{g_1 \cdot g_2 = g} |g_1\rangle \otimes |g_2\rangle \right) = \frac{1}{D} \sum_{g_1 \cdot g_2 = g} |g\rangle$$

Every element in G has exactly $|G| = D$ distinct factorisations (i.e. pairs (gh, h^{-1}) for all $h \in G$), so $\mu_{O'} \delta_{O'} |g\rangle = |g\rangle$. We can also compute the explicit form for the “cap” $\delta_{O'} \circ \eta_{O'}$.

$$\delta_{O'} \eta_{O'} = \delta_{O'} (\sqrt{D} |e\rangle) = \sqrt{D} \sum_{g_1 \cdot g_2 = e} |g_1\rangle \otimes |g_2\rangle = \sqrt{D} \sum_{g \in G} |g\rangle \otimes |g^{-1}\rangle$$

From this it follows that $S |g\rangle = |g^{-1}\rangle$ and $(\mu_{O'}, \eta_{O'}, \delta_O, \epsilon_O, S)$ is (up to a scalar) the induced Hopf algebra of the group algebra $\mathbb{C}[G]$. Therefore O and O' are strongly complementary.

Conversely, let $(\downarrow, \circlearrowleft, \downarrow, \circlearrowright)$ and $(\downarrow, \circlearrowleft, \downarrow, \circlearrowright)$ be strongly complementary \dagger -SCFAs. From Lemma 7.2.4, \downarrow is a monoid over the classical points of \downarrow . We can then evaluate both sides of the equation from Theorem 7.2.7 over an arbitrary classical point.

Since S is a Frobenius algebra automorphism, it is a permutation of the classical points of \downarrow . Thus the previous equation implies that for all i , there exists i' such that:

In other words, all of the classical points of \downarrow have inverses, so \downarrow is isomorphic to the group algebra $\mathbb{C}[G]$ for some Abelian group G . \square

For a D -dimensional space in **FHilb**, fixing \downarrow as the \dagger -SCFA corresponding to the computational basis $\mathcal{B} = \{|i\rangle\}$, we can find the basis inducing the group algebra $\mathbb{C}[\mathbb{Z}_D]$ corresponding to the order- D cyclic group by applying the D -dimensional Fourier transform F_D to the elements of the computation basis: $\mathcal{B}' = \{F_D |i\rangle\}$. Using the fact that $\mathbb{C}[G \times G'] \cong \mathbb{C}[G] \otimes \mathbb{C}[G']$ for Abelian groups G, G' , it is possible to compute the strongly complementary pair corresponding to *any* finite Abelian group G by decomposing it into its cyclic components.

Example 7.2.9. In \mathbb{C}^4 , let \mathcal{B} be the computational basis. Then, we can compute the strongly complementary basis $\mathcal{B}' = \{|e_i\rangle\}$ corresponding to $\mathbb{Z}_2 \times \mathbb{Z}_2$ as follows. The 2D Fourier transform is just the Hadamard matrix H . So $|e_i\rangle = (H \otimes H) |i\rangle$. Writing these as column vectors in the computational basis, we have:

$$\mathcal{B}' = \left\{ \frac{1}{\sqrt{4}} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \frac{1}{\sqrt{4}} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}, \frac{1}{\sqrt{4}} \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}, \frac{1}{\sqrt{4}} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} \right\}$$

In the case of complementary observables, it is often useful to know how big a *complete set* of mutually unbiased bases is for a given dimension. That is, a maximal set of bases such that is pairwise mutually unbiased. In the case of strongly complementary observables, there can only be two.

Theorem 7.2.10. Let \downarrow be a \dagger -SCFA of dimension $D \geq 2$ and let (\downarrow, \bullet) and $(\downarrow, \circlearrowright)$ be strongly complementary pairs. Then \bullet and \circlearrowright cannot be strongly complementary.

Proof. By contradiction. Suppose $(\downarrow_{\bullet}, \downarrow_{\circ})$ is a strongly complementary pair. The units \bullet and \circ must both be proportional to classical points of \downarrow_{\circ} . We already showed that for any strongly complementary pair, $\bullet \circ \circ = \circ \bullet \bullet = \sqrt{D} \neq 0$, so \bullet and \circ must be proportional to the *same* classical point. Then:

$$\begin{array}{c} \bullet \\ \circ \end{array} \downarrow = \begin{array}{c} \bullet \\ \bullet \end{array} \downarrow = \begin{array}{c} \bullet \\ \bullet \end{array} \downarrow = \begin{array}{c} \circ \\ \circ \end{array} \downarrow$$

This is a contradiction because the LHS is invertible, while the RHS is rank $1 < D$. \square

The classification of strongly complementary observables is much simpler than the general case. Whereas the maximum number of mutually unbiased bases of dimension 6 is still unknown, there is (up to isomorphism) one strongly complementary pair, corresponding to the cyclic group $\mathbb{Z}_6 \cong \mathbb{Z}_2 \times \mathbb{Z}_3$.

Phases have a special status for \dagger -SCFAs in **FHilb**. Recall that phases are maps such that:

$$\begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} = \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} = \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array}$$

From Proposition 3.2.15, we can put any phase in the form of the right multiplication by an arbitrary vector. Suppose a \dagger -SCFA corresponds to a basis $\{|i\rangle\}$, then $\mu = \sum |i\rangle \langle ii|$. For an arbitrary vector $|\psi\rangle = \sum \alpha_i |i\rangle$, this is:

$$\mu(1 \otimes |\psi\rangle) = \sum \alpha_i |i\rangle \langle i|$$

So, phases are precisely the maps that are diagonal in the basis defined by a \dagger -SCFA. *Unitary* phases are precisely the phase gates familiar from quantum computing.

For strongly complementary observables, the phases for \downarrow_{\circ} associated with classical points of \downarrow_{\circ} are Frobenius algebra automorphisms of \downarrow_{\circ} , up to a scalar. This follows from the bialgebra law.

$$\begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} = \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} = \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \quad (7.5)$$

7.3 The Z/X Calculus and Quantum Computation

We have already seen that strongly complementary observables satisfy many graphical identities. This collection of identities is sometimes referred to as the *calculus of complementary observables* to emphasise that it can be used as a computational tool. We will now restrict our attention to the complementary pair Z and X , and show how we can use the Z/X -calculus to perform calculations on quantum circuits.

Definitions 7.3.1. Let $\mathcal{Z} = (\mathbb{C}^2, \delta_Z, \epsilon_Z)$ be the \dagger -SCFA corresponding to the Z -observable and let $\mathcal{X} = (\mathbb{C}^2, \delta_X, \epsilon_X)$ be the \dagger -SCFA corresponding to the X -observable.

$$\begin{aligned} \delta_Z &:: |0\rangle \mapsto |00\rangle, |1\rangle \mapsto |11\rangle & \epsilon_Z &:: |0\rangle \mapsto 1, |1\rangle \mapsto 1 \\ \delta_X &:: |+\rangle \mapsto |++\rangle, |-\rangle \mapsto |--\rangle & \epsilon_X &:: |+\rangle \mapsto 1, |-\rangle \mapsto 1 \end{aligned}$$

For the remainder of this section, we will define:

$$\delta_Z := \begin{array}{c} \downarrow \\ \circ \\ \swarrow \quad \searrow \end{array} \quad \epsilon_Z := \begin{array}{c} \downarrow \\ \circ \end{array} \quad \delta_X := \begin{array}{c} \downarrow \\ \bullet \\ \swarrow \quad \searrow \end{array} \quad \epsilon_X := \begin{array}{c} \downarrow \\ \bullet \end{array}$$

Up a global $e^{i\theta}$ factor, the unitary phases for \mathcal{Z} are the phase gates Z_θ and the unitary phases for \mathcal{X} are the phase gates X_θ . We represent these as dots with a phase angle.

$$Z_\alpha := \begin{array}{c} \downarrow \\ \circ \\ \downarrow \end{array} \quad X_\alpha := \begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array}$$

More generally, we can write arbitrary spiders with phases.

$$\begin{array}{c} \dots \\ \swarrow \quad \searrow \\ \circ \\ \swarrow \quad \searrow \\ \dots \end{array} \quad \alpha := \begin{array}{c} \downarrow \\ \circ \\ \swarrow \quad \searrow \\ \dots \end{array} \quad \begin{array}{c} \dots \\ \swarrow \quad \searrow \\ \bullet \\ \swarrow \quad \searrow \\ \dots \end{array} \quad \alpha := \begin{array}{c} \downarrow \\ \bullet \\ \swarrow \quad \searrow \\ \dots \end{array}$$

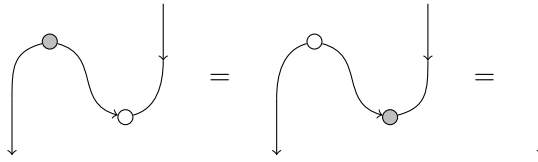
Since the phase commutes with all of the Frobenius structure, it does not matter which leg of the spider we place the phase gate on. For this section, we will ignore (non-zero) scalars, as they will not be important for the calculations. Up to scalar factors, the following equations hold.

$$\begin{array}{c} \dots \\ \swarrow \quad \searrow \\ \circ \\ \swarrow \quad \searrow \\ \dots \end{array} \quad 0 = \begin{array}{c} \downarrow \\ \circ \\ \downarrow \end{array} \quad \begin{array}{c} \dots \\ \swarrow \quad \searrow \\ \circ \\ \swarrow \quad \searrow \\ \dots \end{array} \quad \pi = \begin{array}{c} \downarrow \\ \circ \\ \downarrow \end{array} \quad \begin{array}{c} \dots \\ \swarrow \quad \searrow \\ \bullet \\ \swarrow \quad \searrow \\ \dots \end{array} \quad 0 = \begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array} \quad \begin{array}{c} \dots \\ \swarrow \quad \searrow \\ \bullet \\ \swarrow \quad \searrow \\ \dots \end{array} \quad \pi = \begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array}$$

Since \mathcal{Z} and \mathcal{X} are strongly complementary, the phase gates corresponding to classical points are Frobenius algebra automorphisms. Using equation (7.5) and the fact that $X_\alpha Z_\pi \propto Z_\pi X_{-\alpha}$ for all α , we have:

$$\begin{array}{c} \downarrow \quad \downarrow \\ \pi \quad \pi \\ \dots \\ \downarrow \quad \downarrow \\ \alpha \end{array} = \begin{array}{c} \dots \\ \swarrow \quad \searrow \\ -\alpha \\ \dots \\ \downarrow \quad \downarrow \\ \pi \quad \pi \end{array} \quad \begin{array}{c} \downarrow \quad \downarrow \\ \pi \quad \pi \\ \dots \\ \downarrow \quad \downarrow \\ \alpha \end{array} = \begin{array}{c} \dots \\ \swarrow \quad \searrow \\ -\alpha \\ \dots \\ \downarrow \quad \downarrow \\ \pi \quad \pi \end{array}$$

There is only one Abelian group of order 2, so by Theorem 7.2.8, \mathcal{X} is the group algebra $\mathbb{C}[\mathbb{Z}_2]$ defined over the basis given by \mathcal{Z} . Both elements of \mathbb{Z}_2 are self-inverse, so the antipode of the strongly complementary pair is trivial.



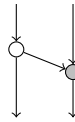
As a consequence, we can freely change the direction of any edge between dots of different colours, and we can delete any two parallel edges between dots of different colours. Finally, we introduce the Hadamard gate, which exchanges the colours of dots.

$$H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

We refer to the bialgebra identities along with these additional rules as the *Z/X calculus*.

7.3.1 Example: Building and Rewriting Circuits

Consider the following map from $\mathbb{C}^2 \rightarrow \mathbb{C}^2$.



By evaluating the first qubit at $|0\rangle$ and $|1\rangle$, we can see that this map selectively applies X_π .



Therefore, it is a CNOT gate. From this, we have a universality theorem.

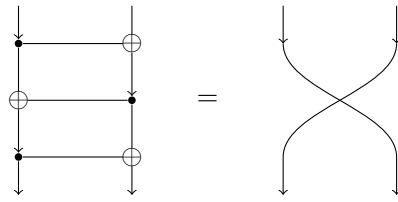
Theorem 7.3.2. *The generators of the Z/X calculus are universal for quantum computation.*

Proof. We have already constructed a CNOT gate, so it suffices to show we can construct an arbitrary 1-qubit unitary. This is possible because every unitary map $\mathbb{C}^2 \rightarrow \mathbb{C}^2$ admits an Euler decomposition $U = Z_\gamma X_\beta Z_\alpha$. That is:

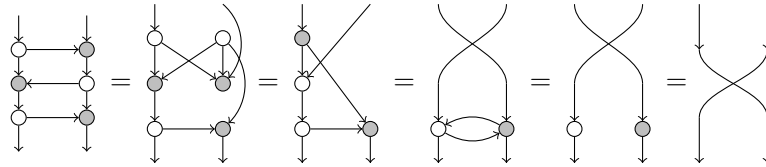
$$U = \begin{array}{c} \downarrow \\ \alpha \\ \downarrow \\ \beta \\ \downarrow \\ \gamma \\ \downarrow \end{array} \quad (7.6)$$

□

Example 7.3.3. It is a basic property of CNOT gates that three alternating applications yields a qubit swap:



We prove this using the Z/X calculus.



More examples like this can be found in [16]. Hillebrand applied to Z/X calculus to a wide variety of security protocols in [31], and Duncan and Perdrix applied it to MBQC in [25].

Chapter 8

Monoidal Algebra in Quantum Entanglement Theory

We now turn our attention to a different topic in quantum information theory: multipartite entanglement. In this chapter, we review several major results from the study of multipartite quantum entanglement. We then give an algebraic (i.e. diagrammatic) characterisation of a special class of highly entangled, symmetric states called Frobenius states. Frobenius states always induce commutative Frobenius algebras, and it can be shown that the two canonical maximally-entangled states on qubits, GHZ and W , can be distinguished by a simple property of this induced Frobenius algebra: specialness or anti-specialness.

In studying GHZ and W states abstractly, we introduce the notion of a GW-pair. A GW-pair consists of a special commutative Frobenius algebra and an anti-special commutative Frobenius algebra, and it exhibits an interaction theory characteristic of the algebras induced by the GHZ and W states. We provide a behavioural intuition for the generators of a GW-pair, show that they are universal for quantum computing, and use them to encode arithmetic on the complex projective line.

8.1 Classifying Entanglement

Characterising general N -system entangled states is a very hard open problem in quantum information theory. Before talking about applications of categorical diagrams to the study of entangled states, we will briefly give some background and major results from the field.

Bipartite states, i.e. quantum states consisting of two entangled systems, are fairly well understood. We can characterise bipartite states by “how much” entanglement they have: ranging from product states (which have no entanglement), to perfectly correlated states (which have maximal entanglement).

(8.1)

This characterisation can be formalised using the *majorisation order* on bipartite states. This is done using the *Schmidt decomposition*. For any bipartite state $|\Psi\rangle \in \mathcal{H} \otimes \mathcal{H}$, there exist orthonormal bases $\{|u_i\rangle\}$ and $\{|v_i\rangle\}$ and non-negative real numbers α_i such that:

$$|\Psi\rangle = \sum_i \alpha_i |u_i\rangle \otimes |v_i\rangle$$

The numbers α_i are called the *Schmidt coefficients* of $|\Psi\rangle$ and are uniquely determined, up to permutation, by $|\Psi\rangle$. The number of non-zero Schmidt coefficients is called the *Schmidt rank*. By reordering the associated basis vectors, we can always assume these coefficients are in decreasing order $\alpha_0 \geq \alpha_1 \geq \dots \geq \alpha_{D-1}$. We can define the majorisation ordering on states $|\Psi\rangle, |\Phi\rangle \in \mathcal{H} \otimes \mathcal{H}$ using their associated Schmidt coefficients $\{\alpha_i\}$ and $\{\beta_i\}$.

$$|\Psi\rangle \leq_M |\Phi\rangle \Leftrightarrow \forall k. \left(\sum_{i=0}^k \alpha_i^2 \geq \sum_{i=0}^k \beta_i^2 \right)$$

Intuitively, states whose Schmidt coefficients are more evenly spread are higher in the majorisation order. For instance, the product state $|00\rangle$ has Schmidt coefficients $(1, 0)$, whereas the Bell state $|\text{Bell}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ has Schmidt coefficients $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. Since $\frac{1}{2} \leq 1$ and $\frac{1}{2} + \frac{1}{2} \leq 1 + 0$, $|00\rangle \leq_M |\text{Bell}\rangle$.

This relation is transitive, reflexive, and anti-symmetric up to a change of orthonormal bases $|u_i\rangle$ and $|v_i\rangle$. We call two states that are equal up to a change of orthonormal basis on each subsystem *LU-equivalent*, or equivalent up to *local unitaries*.

Definition 8.1.1. Two states $\Psi, \Phi \in \mathcal{H} \otimes \dots \otimes \mathcal{H}$ are said to be *LU-equivalent* if there exist unitary maps $U_i : \mathcal{H} \rightarrow \mathcal{H}$ such that:

So, \leq_M forms a partial order on LU-classes of bipartite states. For qubits, \leq_M is a total order, but it is not total in general. Consider two states in $\mathbb{C}^3 \otimes \mathbb{C}^3$:

$$|\Psi\rangle = \frac{1}{\sqrt{4}} \left(\sqrt{2} |00\rangle + |11\rangle + |22\rangle \right) \quad |\Phi\rangle = \frac{1}{\sqrt{5}} \left(\sqrt{2} |00\rangle + \sqrt{2} |11\rangle + |22\rangle \right)$$

Then, it is neither the case that $|\Psi\rangle \leq_M |\Phi\rangle$ nor that $|\Phi\rangle \leq_M |\Psi\rangle$. However the minimal and maximal elements of \leq_M are always unique, up to LU-equivalence.

We can get a better feel for what characterises these states by thinking of bipartite states as processes, or *quantum channels*, over which information can flow. We do this by employing the principal of *channel-state duality*.

Fixing an orthonormal basis $\{|i\rangle\} \in \mathcal{H}$ fixes a unitary isomorphism $\chi : \mathcal{H}^* \rightarrow \mathcal{H}$ sending $\langle i|$ to $|i\rangle$. As we have seen in the previous chapter, this is the same as fixing a \dagger -special commutative Frobenius algebra. The map χ is then the following induced isomorphism $\mathcal{H}^* \cong \mathcal{H}$:

$$\chi := \begin{array}{c} \uparrow \\ \circ \\ \downarrow \end{array} = \begin{array}{c} \circ \\ \downarrow \quad \uparrow \\ \downarrow \quad \uparrow \end{array} \quad \chi^\dagger := \begin{array}{c} \downarrow \\ \circ \\ \uparrow \end{array} = \begin{array}{c} \downarrow \quad \uparrow \\ \circ \\ \downarrow \quad \uparrow \end{array}$$

Up to normalisation factors, we can consider linear maps $L : \mathcal{H} \rightarrow \mathcal{H}$ as bipartite states $|\Psi_L\rangle \in L \otimes L$ and bipartite states $|\Psi\rangle \in \mathcal{H} \otimes \mathcal{H}$ as linear maps $L_\Psi : \mathcal{H} \rightarrow \mathcal{H}$.

$$\begin{array}{c} \triangle \\ \Psi_L \\ \downarrow \quad \downarrow \end{array} := \begin{array}{c} \downarrow \quad \uparrow \\ \square \\ \downarrow \quad \downarrow \end{array} \quad \begin{array}{c} \downarrow \\ \square \\ \downarrow \end{array} := \begin{array}{c} \triangle \\ \Psi \\ \downarrow \quad \downarrow \end{array} \quad (8.2)$$

This is the “pure” version of the *Choi-Jamiołkowski isomorphism*. The general statement, which includes mixed states, is as follows. For a finite-dimensional Hilbert space \mathcal{H} , let $\mathcal{L}(\mathcal{H})$ be the vector space of linear maps $\mathcal{H} \rightarrow \mathcal{H}$. Positive operators correspond to mixed states and completely positive maps correspond to (mixed) quantum operations.

Theorem 8.1.2. *Positive operators $L \in \mathcal{L}(\mathcal{H} \otimes \mathcal{H})$ are in one-to-one correspondence with completely positive maps $\Phi : \mathcal{L}(\mathcal{H}) \rightarrow \mathcal{L}(\mathcal{H})$.*

Since pure states (up to a global phase) are the same thing as positive operators of the form $\rho := |\psi\rangle\langle\psi|$ and pure maps $L : \mathcal{H} \rightarrow \mathcal{H}$ are the same thing as CPMs of the form $\Phi(\rho) = L\rho L^\dagger$, equation (8.2) can be thought of as the pure fragment of channel-state duality. Under this correspondence, the Schmidt decomposition of a state is essentially the same as the singular value decomposition of its associated map. Let $\langle v'_i| = \chi^\dagger(|v_i\rangle)$, i.e. the transposition of $|v_i\rangle$ in the basis $\{|i\rangle\}$. This clearly forms an orthonormal basis for \mathcal{H}^* . Using this basis, we can decompose L_Ψ such that its singular values are the same as the Schmidt coefficients of $|\Psi\rangle$.

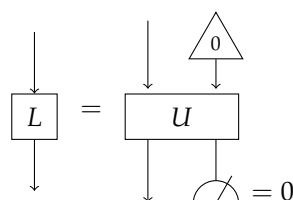
$$|\Psi\rangle = \sum_i \alpha_i |u_i\rangle \otimes |v_i\rangle \quad \leftrightarrow \quad L = \sum_i \alpha_i |u_i\rangle\langle v'_i|$$

In particular, the Schmidt rank of a bipartite state is the same as the rank of the associated map.

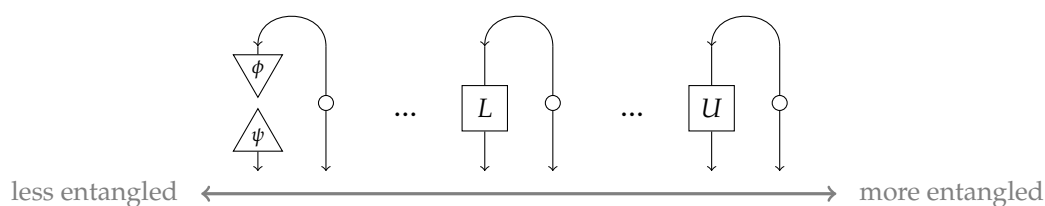
Quantum teleportation is an archetypal example of regarding a bipartite state as a channel. The entangled state that Alice and Bob share provides the “medium” over which the unknown state is teleported from Alice to Bob. Suppose we considered variants of the teleportation protocol over an arbitrary finite-dimensional Hilbert space \mathcal{H} , replacing the (perfectly correlated) Bell state with other kinds of bipartite states from chart (8.1). The states at the left extreme are the worst for teleportation. Regarded as channels, product states correspond to rank-one maps. No matter what

we put in to such a channel, we always get the same output, up to a scalar. Such channels cannot be used to send *any* quantum data. At the other extreme are the perfectly correlated states, which correspond to unitary maps under channel-state duality. Any such state can be used to construct a teleportation protocol which will always succeed. If this unitary U_Ψ is not the identity (as in the case of the Bell state), we simply need to undo it by applying U_Ψ^\dagger to one of the sub-systems.

In between these two extremes are maps L of rank $2 \leq r \leq D$, which can be thought of as noisy channels. Suppose $r = D$, then $|\Psi_L\rangle$ can at least in principal teleport a state, but it might be impossible for Bob to recover Alice's state deterministically. However, in most cases, Bob can at least recover the state with non-zero probability. This is because an arbitrary linear map can be "applied" to a quantum system by first applying a big unitary to the system and an ancilla state, then measuring the ancilla.



If Bob gets outcome 0, then he has successfully applied L , otherwise he has applied some other (unwanted) map L' . Maps L that can be applied with non-zero probability are known as *stochastic maps*. In the case where $r < D$, states can only be teleported in a lossy sense, i.e. they are (non-deterministically) projected on to a subspace of \mathcal{H} before being sent. Under channel-state duality, chart (8.1) becomes:



When we move from *bipartite* entanglement to *multipartite* entanglement, the picture becomes less clear. For one thing, this is no canonical analogue to the majorisation order involving three or more systems. However, we can define an *operational* ordering on states that is equivalent to the majorisation ordering in the bipartite case. Operational orderings relate states by the existence of certain types of quantum protocols that can convert one state into another. The most well-known type of protocol used for this purpose is a LOCC protocol.

Definition 8.1.3. An N -partite state $|\Psi\rangle$ can be converted into $|\Phi\rangle$ by *Local Operations and Classical Communication* (LOCC) if there exists an N -party protocol that can *deterministically* convert $|\Psi\rangle$ into $|\Phi\rangle$ consisting of any number of the following operations:

1. Party p_i performs a local measurement of the i -th subsystem (possibly with an ancillary system which only p_i can access).
2. Party p_j performs a local unitary, which can be conditioned on any previous measurement outcome in the protocol.

In such a case, we write $|\Phi\rangle \leq_{\text{LOCC}} |\Psi\rangle$. If two states are LOCC-interconvertible (i.e. $|\Phi\rangle \leq_{\text{LOCC}} |\Psi\rangle$ and $|\Psi\rangle \leq_{\text{LOCC}} |\Phi\rangle$), we say they are *LOCC-equivalent*, written $|\Psi\rangle \sim_{\text{LOCC}} |\Phi\rangle$.

Nielsen and Vidal showed that the majorisation order is the same as the LOCC order on bipartite states [55]. As a consequence, two states are LOCC-equivalent if and only if they are LU-equivalent. In fact, this relationship between LU-equivalence and LOCC-equivalence is true for any number of systems.

Theorem 8.1.4 ([9]). *Two N -partite quantum states are LOCC-equivalent if and only if they are LU-equivalent.*

Example 8.1.5. The following two bipartite states are LOCC-equivalent:

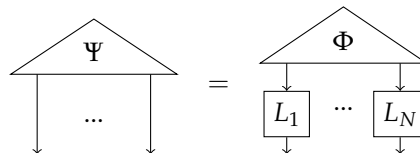
$$|\text{Bell}\rangle := \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \quad |\text{EPR}\rangle := \frac{1}{\sqrt{2}} (|01\rangle + |10\rangle)$$

Even though there are infinitely many LOCC classes in the bipartite case, the majorisation order gives us a straightforward way to characterise LOCC-equivalence, as well as those states that are minimal and maximal with respect to \leq_{LOCC} . However, for three or more systems, the picture becomes much more complicated. For that reason, it is convenient to introduce a course-graining of LOCC called Stochastic LOCC, or SLOCC. For SLOCC, we relax the requirement that the LOCC protocol succeeds deterministically and merely require that it succeed with some non-zero probability.

Definition 8.1.6. An N -partite state $|\Psi\rangle$ can be converted into $|\Phi\rangle$ by *Stochastic Local Operations and Classical Communication* (SLOCC) if there exists an N -party LOCC protocol that converts $|\Psi\rangle$ into $|\Phi\rangle$ with a *non-zero probability*.

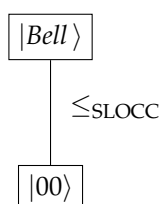
Again, we use \leq_{SLOCC} and \sim_{SLOCC} to represent SLOCC-convertibility and SLOCC-equivalence, respectively. It was shown by Dür, Vidal, and Cirac that SLOCC-equivalence can be characterised in a similar manner to LOCC-equivalence, but replacing *unitary* maps with *invertible* maps.

Definition 8.1.7. Two states $|\Psi\rangle, |\Phi\rangle \in \mathcal{H} \otimes \dots \otimes \mathcal{H}$ are said to be *ILO-equivalent* if there exist invertible local operations $L_i : \mathcal{H} \rightarrow \mathcal{H}$ such that:

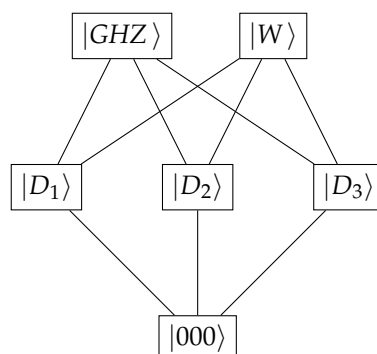


Theorem 8.1.8 ([26]). *Two states $|\Psi\rangle, |\Phi\rangle \in \mathcal{H} \otimes \dots \otimes \mathcal{H}$ are SLOCC-equivalent iff they are ILO-equivalent.*

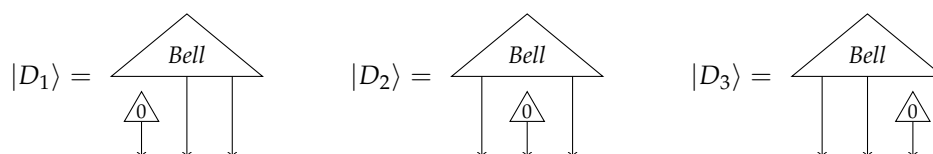
We say a state $|\Psi\rangle$ is *SLOCC-maximal* if its SLOCC-equivalence class is maximal with respect to \leq_{SLOCC} , i.e. $|\Psi\rangle \leq_{\text{SLOCC}} |\Phi\rangle \implies |\Phi\rangle \sim_{\text{SLOCC}} |\Psi\rangle$. We define SLOCC-minimal states similarly. For any number of systems, the unique minimal SLOCC-equivalence class is the class of product states $|00\dots 0\rangle$. For $\mathbb{C}^2 \otimes \mathbb{C}^2$, the only other SLOCC class is generated by the perfectly correlated states. So, the Hasse diagram of SLOCC classes has only two elements.



For three qubits, there are still only six SLOCC classes (or four classes, up to permutations of qubits), there is no longer a unique maximal SLOCC-class. The Hasse diagram of the tripartite SLOCC classes is:



Where $|\text{GHZ}\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$, $|\text{W}\rangle = \frac{1}{\sqrt{3}}(|100\rangle + |010\rangle + |001\rangle)$, and the $|D_i\rangle$ states represent the three separable configurations of a Bell state with $|0\rangle$.



The complete classification for three qubits was given by Dür, Vidal, and Cirac [26]. They distinguished GHZ and W by using an entanglement measure called the *3-tangle*, or *residual tangle*. Intuitively, this is the entanglement “left over” after bipartite correlations are subtracted out. States that are SLOCC-equivalent to $|\text{GHZ}\rangle$ have a non-zero 3-tangle. These states have some true 3-body entanglement, even accounting for bipartite correlations. However, states that are SLOCC-equivalent to $|\text{W}\rangle$ always have a vanishing 3-tangle, which means that all the entanglement present can be accounted for by bipartite correlations. Informally, we might depict the correlations in GHZ and W as:



One could say that W has no “true” tripartite entanglement. One could also state this property positively, by stating that W -like entanglement, unlike GHZ -like entanglement, is (partially) robust to the loss of a one system. That is, if we trace out one of the three parties, the reduced density matrix of W becomes (ignoring normalisation):

$$\rho'_W = (|10\rangle + |01\rangle)(\langle 10| + \langle 01|) + |00\rangle\langle 00| = \sqrt{2} |EPR\rangle\langle EPR| + |00\rangle\langle 00|$$

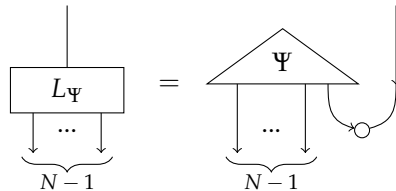
So, ρ'_W corresponds to the probabilistic mixture of the (entangled) Einstein-Podolsky-Rosen state $|EPR\rangle$ and a separable pure state $|00\rangle$. However, if we do the same to the GHZ state, we get:

$$\rho'_{GHZ} = |00\rangle\langle 00| + |11\rangle\langle 11|$$

This is a mixture of two product states. Therefore it contains no entanglement if the third system is disregarded.

There are various schemes for classifying multipartite entanglement which have had some success for small (typically < 6) numbers of systems. These approaches have had mixed results, as the general problem becomes very difficult for four or more systems. For four or more systems, there is necessarily an infinite number of SLOCC classes [26]. This is because the continuous degrees of freedom in the state space of N systems $(2^N - 2)$ quickly overcomes the degrees of freedom available in a tensor product $L_1 \otimes \dots \otimes L_N$ of local invertible maps $(6N + 2)$. Nevertheless, one sometimes give a finite set of SLOCC *super-classes*, i.e. SLOCC classes with some free parameters, which span the state spaces of these larger systems. However, there is no one right choice for a parametrisation, so these super-classes are not uniquely determined and often reflect the methods that were used to obtain them.

Lamata et al introduced an inductive scheme for classifying multipartite states [46] for N partite states in terms of the classification of $N - 1$ partite states. It works by treating N partite states as maps from \mathcal{H} to $\mathcal{H}^{\otimes(N-1)}$.



They then look at the image $V \subseteq \mathcal{H}^{\otimes(N-1)}$ of L_Ψ and classify the vectors that span V (i.e. $N - 1$ partite states). Using their classification scheme, $|GHZ\rangle$ is the unique, tripartite qubit state whose image is spanned by two product states $\{|\psi_1\rangle \otimes |\phi_1\rangle, |\psi_2\rangle \otimes |\phi_2\rangle\}$ where $|\psi_1\rangle \neq \lambda |\psi_2\rangle$ and

$|\phi_1\rangle \neq \lambda' |\phi_2\rangle$. $|W\rangle$ is the unique state whose image is two dimensional, but only contains one product state. The whole classification for tripartite qubit states is as follows.

State	V spanned by
$ GHZ\rangle$	$\{ \psi_1\rangle \otimes \phi_1\rangle, \psi_2\rangle \otimes \phi_2\rangle\}$
$ W\rangle$	$\{ \psi\rangle \otimes \phi\rangle, \Psi\rangle\}$
$ D_1\rangle$	$\{ \psi\rangle \otimes \phi_1\rangle, \psi\rangle \otimes \phi_2\rangle\}$
$ D_2\rangle$	$\{ \psi_1\rangle \otimes \phi\rangle, \psi_2\rangle \otimes \phi\rangle\}$
$ D_3\rangle$	$\{ \Psi\rangle\}$
$ 000\rangle$	$\{ \psi\rangle \otimes \phi\rangle\}$

Since it can be proved that any two-dimensional subspace of $\mathbb{C}^2 \otimes \mathbb{C}^2$ contains at least one product state, this list is exhaustive, and matches the classification given by [26]. It also reflects the fact that tracing out a qubit from $|GHZ\rangle$ yields a mixture of product states, while tracing out a qubit of $|W\rangle$ yields a mixture of a product state and an entangled state.

In [47], the authors went on to use this technique to enumerate a finite set of 4-partite SLOCC super-classes spanning $(\mathbb{C}^2)^{\otimes 4}$. However, the amount of calculations to ensure these super-classes were non-overlapping and spanned the whole space was much greater than in the tripartite case, and it seems likely that enumeration of super-classes with more than four subsystems would be significantly more difficult than the 4-partite case.

8.1.1 Symmetric States

The study of symmetric multipartite states is quite a bit simpler, because they have significantly fewer degrees of freedom. For instance, the space of N -partite symmetric states over qubits is spanned by the N -partite Dicke states:

$$|D_N^{(k)}\rangle = \mathcal{S}_N(|\underbrace{0\dots 0}_{N-k} \underbrace{1\dots 1}_k\rangle)$$

Where \mathcal{S}_N is the N -system symmetrisation map.

$$\mathcal{S}_N :: |i_1, i_2, \dots, i_N\rangle \mapsto \frac{1}{N!} \sum_{\pi \in \text{Perms}(N)} |i_{\pi(1)}, i_{\pi(2)}, \dots, i_{\pi(N)}\rangle$$

So, for qubits, the space of N -system symmetric states is $(N + 1)$ -dimensional. It can also be shown that an arbitrary N -partite state is the symmetrisation of a product state.

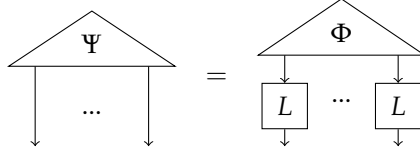
$$|\Psi\rangle = \mathcal{S}_N(|\psi_1, \psi_2, \dots, \psi_N\rangle)$$

The factors $|\psi_i\rangle$ need not be distinct. Since the value of $|\Psi\rangle$ does not depend on the order of factors, $|\Psi\rangle$ is totally defined by the multiset of $M \leq N$ distinct factors $|\phi_i\rangle$:

$$|\Psi\rangle = \mathcal{S}_N(\underbrace{|\phi_1, \dots, \phi_1\rangle}_{d_1} \otimes \underbrace{|\phi_2, \dots, \phi_2\rangle}_{d_2} \otimes \dots \otimes \underbrace{|\phi_M, \dots, \phi_M\rangle}_{d_M})$$

We can also assume that the d_i are in decreasing order. The number M of distinct factors is called the *diversity degree* of a symmetric state, and the list $\mathcal{D}_\Psi = [d_1, d_2, \dots, d_M]$ is called its *degeneracy configuration*.

Definition 8.1.9. Two N -partite states $|\Psi\rangle, |\Phi\rangle \in \mathcal{H}^{\otimes N}$ are called *symmetrically SLOCC-equivalent* if there exists a *single* invertible map $L : \mathcal{H} \rightarrow \mathcal{H}$ such that:



Bastin et al provided a classification result for symmetric qubit states of any size with a small diversity degree [8]. It relies crucially on the following fact about qubit states.

Proposition 8.1.10. *Two N -partite symmetric states are SLOCC-equivalent if and only if they are symmetrically SLOCC-equivalent.*

Using this fact, they showed that for small diversity degree, SLOCC-classes are uniquely fixed by their degeneracy configuration.

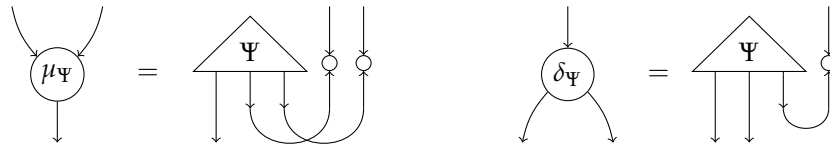
Theorem 8.1.11 ([8]). *Two N -partite symmetric qubit states with diversity degree $M \leq 3$ are SLOCC-equivalent if and only if they have the same degeneracy configuration.*

They proved that any state with diversity degree $M = N$ is SLOCC-equivalent to $|GHZ_N\rangle = |0 \dots 0\rangle + |1 \dots 1\rangle$. As a consequence, the SLOCC classes of N -partite symmetric states are completely determined for $N \leq 4$. They also showed that for $4 \leq M < N$, there are necessarily infinitely-many SLOCC classes of symmetric states.

8.2 Strong SLOCC-maximality and strong symmetry

In an effort to push state classification farther, we will identify the properties of that make GHZ and W states unique. In order to leverage techniques from categorical quantum mechanics and diagrammatic languages, we shall focus on properties that are *compositional* in nature. That is, we shall look at how GHZ and W states, treated as maps via channel-state duality, behave when they are composed with other states or themselves.

Whereas bipartite states can be thought of as quantum channels, tripartite states can be thought of as *algebraic operations* which have two channels of input and one channel of output (or, equivalently, as coalgebraic operations from 1 input to 2 outputs).



The GHZ and W states are characterised as the unique tripartite qubit states that are SLOCC-maximal, so we shall look at how to characterise SLOCC-maximality abstractly. For bipartite states, this condition is equivalent to the a bipartite state forming the “cap” of a self-dual compact structure.

Proposition 8.2.1. *A bipartite state $|\Psi\rangle \in \mathcal{H} \otimes \mathcal{H}$ is SLOCC-maximal iff there exists an effect $\langle\Phi| : \mathcal{H} \otimes \mathcal{H} \rightarrow \mathbb{C}$ such that*

$$\begin{array}{c} \triangle \\ \Psi \\ \downarrow \\ \downarrow \end{array} = \begin{array}{c} \downarrow \\ \downarrow \\ \triangle \\ \Phi \\ \downarrow \end{array} \quad (8.3)$$

Proof. We first show that SLOCC-maximal maps have full Schmidt rank. Suppose $|\Psi\rangle$ has Schmidt decomposition:

$$|\Psi\rangle = \sum_i \alpha_i |u_i\rangle \otimes |v_i\rangle$$

If for any i , $\alpha_i = 0$, then there exists a map S with $|u_i\rangle$ in its null space such that $|\Psi\rangle = (S \otimes 1)|\Psi'\rangle$. Since S is singular, it corresponds to a non-reversible stochastic local operation, so $|\Psi\rangle$ is not SLOCC-maximal.

Since any SLOCC-maximal state must have full Schmidt rank, we can construct $\langle\Phi|$ as follows.

$$\langle\Phi| = \sum_i \frac{1}{\alpha_i} \langle v_i| \otimes \langle u_i|$$

Then equation (8.3) is satisfied. Conversely, if $|\Psi\rangle$ were not SLOCC maximal, then $|\Psi\rangle = (S \otimes 1)|\Psi'\rangle$ for some $|\Psi'\rangle$ and some singular map S . Clearly no bipartite state of this form could satisfy equation (8.3). \square

Characterising SLOCC-maximal tripartite states is trickier. However, GHZ and W satisfy a stronger version of SLOCC-maximality. As in [46], we can study tripartite entangled states by studying the states that span the image of the associated map δ_Ψ .

$$\text{Im} \left(\begin{array}{c} \downarrow \\ \delta_\Psi \\ \downarrow \end{array} \right) = \text{Span} \left\{ \begin{array}{c} \triangle \\ i \\ \downarrow \\ \delta_\Psi \\ \downarrow \end{array} \right\} = \text{Span} \left\{ \begin{array}{c} \triangle \\ \Psi \\ \downarrow \\ \downarrow \\ \triangle \\ i \end{array} \right\}$$

For convenience, we define the 3 bipartite *image spaces* of $|\Psi\rangle$ as follows.

$$\text{Im}_1(|\Psi\rangle) = \text{Span} \{ (\langle i| \otimes 1 \otimes 1) |\Psi\rangle \}$$

$$\text{Im}_2(|\Psi\rangle) = \text{Span} \{ (1 \otimes \langle i| \otimes 1) |\Psi\rangle \}$$

$$\text{Im}_3(|\Psi\rangle) = \text{Span} \{ (1 \otimes 1 \otimes \langle i|) |\Psi\rangle \}$$

For states in $(\mathbb{C}^D)^{\otimes 3}$ to be SLOCC-maximal, these spaces must all be D -dimensional. In fact, this condition is equivalent to SLOCC-maximality.

Proposition 8.2.2. A tripartite state $|\Psi\rangle$ is SLOCC-maximal if and only if $\text{Im}_i(|\Psi\rangle)$ is D -dimensional for $i = 1, 2, 3$.

Proof. Suppose that $\text{Im}_1(|\Psi\rangle)$ is not D -dimensional. Then, there must exist $\langle\psi| \in \mathbb{C}^D$ such that $(\langle\psi| \otimes 1 \otimes 1)|\Psi\rangle = 0$. Fix an orthonormal basis $\langle u_i|$ spanning $\langle\psi|^\perp$. Then define a singular map $S = \sum_i |u_i\rangle\langle u_i|$, and note that there exists $|\Psi'\rangle$ such that $(S \otimes 1 \otimes 1)|\Psi'\rangle = |\Psi\rangle$, so $|\Psi\rangle$ cannot be SLOCC-maximal. The argument follows similarly for $\text{Im}_2(|\Psi\rangle)$ and $\text{Im}_3(|\Psi\rangle)$.

Conversely, let $|\Psi\rangle$ not be SLOCC-maximal. Then it must be in one of the following forms:

$$(S \otimes 1 \otimes 1)|\Psi'\rangle \quad (1 \otimes S \otimes 1)|\Psi'\rangle \quad (1 \otimes 1 \otimes S)|\Psi'\rangle$$

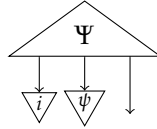
In each of these cases, one of the spaces $\text{Im}_i(|\Psi\rangle)$ must be less than D -dimensional. □

It is always the case for SLOCC-maximal tripartite states that each of the spaces $\text{Im}_i(|\Psi\rangle)$ contain entangled states. However, for $D > 2$, it need not contain a SLOCC-maximal bipartite state. The case of $D = 2$ is degenerate in the sense that *any* entangled state is SLOCC-maximal.

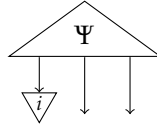
Definition 8.2.3. A tripartite state $|\Psi\rangle$ is called *strongly SLOCC-maximal* if each of its associated image spaces $\text{Im}_i(|\Psi\rangle)$ contain a SLOCC-maximal bipartite state.

Theorem 8.2.4. Strong SLOCC-maximality implies SLOCC-maximality.

Proof. It suffices to show that the image spaces $\text{Im}_i(|\Psi\rangle)$ are all D -dimensional. Let $\langle\psi|$ be a state such that $|\Phi\rangle = (1 \otimes \langle\psi| \otimes 1)|\Psi\rangle$ is a SLOCC-maximal bipartite state. Then, fixing an orthonormal basis $\langle i|$ for $(\mathbb{C}^D)^*$, all states of the form



must be linearly independent for distinct values of i , so they span \mathbb{C}^D . Thus, non-projected states:



must span a D -dimensional subspace of $\mathbb{C}^D \otimes \mathbb{C}^D$. The result for $\text{Im}_2(|\Psi\rangle)$ and $\text{Im}_3(|\Psi\rangle)$ follows similarly. □

Remark 8.2.5. In the case of $D = 2$, any entangled bipartite state is SLOCC-maximal, so strong SLOCC-maximality and “weak” SLOCC-maximality are equivalent. However, for $D > 2$, the implication in Theorem 8.2.4 is strict. To see this, consider the following state in $\mathbb{C}^3 \otimes \mathbb{C}^3 \otimes \mathbb{C}^3$.

$$|\Psi\rangle = |000\rangle + |101\rangle + |110\rangle + |202\rangle + |220\rangle$$

The spaces $\text{Im}_1(|\Psi\rangle)$, $\text{Im}_2(|\Psi\rangle)$, and $\text{Im}_3(|\Psi\rangle)$ are all 3-dimensional, so $|\Psi\rangle$ is SLOCC-maximal. Every state in $\text{Im}_i(|\Psi\rangle)$ is of the form:

$$|\Phi\rangle = a|00\rangle + b(|01\rangle + |10\rangle) + c(|02\rangle + |20\rangle)$$

But $|\Psi\rangle$ cannot be SLOCC-maximal. If $b = c = 0$, then it is a product state, otherwise there exists a non-zero bra $\langle\zeta| := c\langle 1| - b\langle 2|$ such that $(\langle\zeta| \otimes 1)|\Phi\rangle = 0$.

When a state is symmetric we can simplify the strong maximality condition. A state $|\Psi\rangle$ is strongly SLOCC-maximal if there exist effects $\langle\zeta|, \langle\Phi|$ such that:

$$(8.4)$$

The GHZ and W states are strongly SLOCC-maximal because they are SLOCC-maximal qubit states. In particular, ignoring scalar factors, the following equations hold:

$$(8.5)$$

The GHZ and W states have natural N -qubit symmetric analogues.

$$|\text{GHZ}_N\rangle := |00\dots 0\rangle + |11\dots 1\rangle$$

$$|\text{W}_N\rangle := |10\dots 0\rangle + |010\dots 0\rangle + \dots + |0\dots 01\rangle$$

Not only do they have such N -partite versions, they come with a recipe for inductively constructing them. That is, for both of these states, there is a bipartite effect $\langle\Phi|$ that can be used to “glue” a tripartite state on to an N -partite state by projecting out a pair of qubits to make an $(N + 1)$ -partite symmetric state.

$$|\text{GHZ}_{N+1}\rangle = (1 \otimes \langle\text{Bell}| \otimes 1)(|\text{GHZ}_N\rangle \otimes |\text{GHZ}\rangle)$$

$$|\text{W}_{N+1}\rangle = (1 \otimes \langle\text{EPR}| \otimes 1)(|\text{W}_N\rangle \otimes |\text{W}\rangle)$$

To inductively build a symmetric N -partite state, it suffices that the following condition hold.

Definition 8.2.6. A symmetric state is said to be *strongly symmetric* if there exists some bipartite effect $\langle\Phi|$ such that

$$(8.6)$$

8.3 Frobenius States and their Induced Frobenius Algebras

A state that is strongly SLOCC-maximal and strongly symmetric is called a Frobenius state.

Definition 8.3.1. A symmetric tripartite state $|\Psi\rangle \in \mathcal{H} \otimes \mathcal{H} \otimes \mathcal{H}$ is said to be a Frobenius state if there exist effects $\langle\Phi|, \langle\zeta|$ such that:

$$(8.7)$$

Note that $\langle\Phi|$ satisfying two equations in the definition must be the *same* effect. This is a stronger condition than stating that a Frobenius state is both strongly SLOCC-maximal and strongly symmetric (i.e. these equations respectively hold for some possibly distinct effects $\langle\Phi|$ and $\langle\Phi'|$).

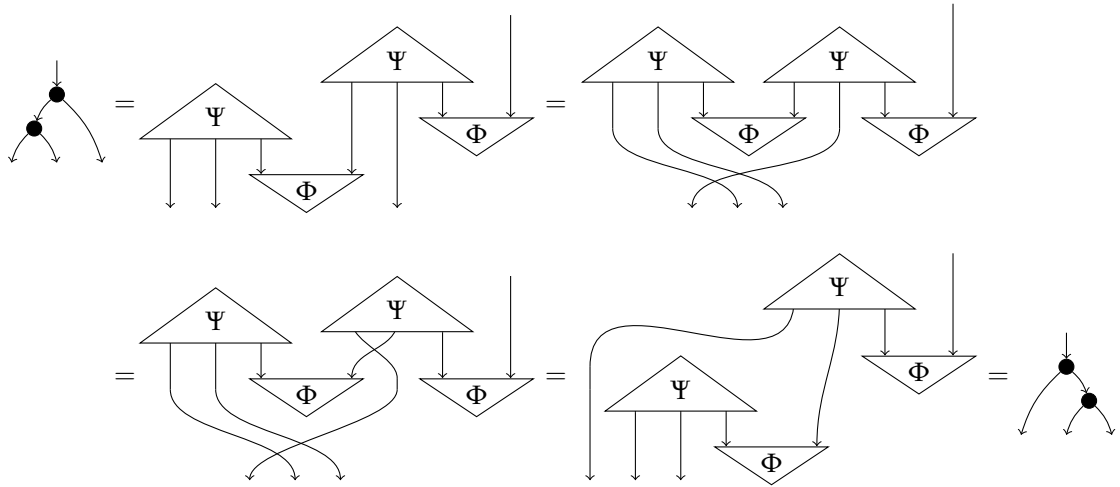
Theorem 8.3.2 (Algebras as states). For any commutative Frobenius algebra $(\downarrow_{\bullet}, \uparrow_{\bullet}, \downarrow_{\bullet}, \uparrow_{\bullet})$, the following is a Frobenius state, with its two associated effects:

Proof. The induced tripartite state is symmetric because \downarrow_{\bullet} is co-commutative. The two Frobenius state equations follow from the spider theorem of commutative Frobenius algebras. \square

The Frobenius state conditions from Definition 8.3.1 hold as a consequence of Theorem 3.2.20. Also, from any Frobenius state we can construct the associated commutative Frobenius algebra.

Theorem 8.3.3 (States as algebras). For any Frobenius state $|\Psi\rangle$, there exist effects $\langle\Phi|, \langle\zeta|$ such that the following is a commutative Frobenius algebra:

Proof. Cocommutativity of \downarrow_{\bullet} follows from $|\Psi\rangle$ being symmetric. Coassociativity follows from the symmetry of $|\Psi\rangle$ and the strong symmetry equation.

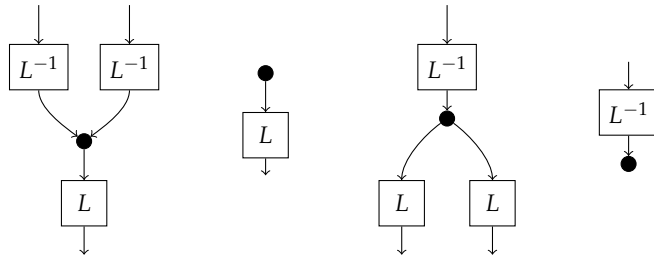


Associativity and the Frobenius law follow similarly. Strong SLOCC-maximality implies that $\langle \zeta |$ is a counit for \downarrow and that $(\langle \zeta | \otimes \langle \zeta | \otimes 1) | \Psi \rangle$ is a unit for \downarrow . \square

We can therefore define a commutative Frobenius algebra using either the usual maps $(\mu, \eta, \delta, \epsilon)$, or a triple $(|\Psi\rangle, \langle \Phi |, \langle \zeta |)$ consisting of a Frobenius state and its two associated effects. Also, note that for a given state $|\Psi\rangle$, there could be multiple induced commutative Frobenius algebras based upon the choice of $\langle \zeta |$. However, once $\langle \zeta |$ is fixed, $\langle \Phi |$ is completely determined. This is analogous to the situation with Frobenius algebras where the maps μ and ϵ completely determine the other two.

Theorem 8.3.4. *If $(|\Psi\rangle, \langle \Phi |, \langle \zeta |)$ is a Frobenius state and $|\Psi'\rangle$ is symmetrically SLOCC-equivalent to $|\Psi\rangle$, then $|\Psi'\rangle$ extends to a Frobenius state in such a way that the induced commutative Frobenius algebras of $|\Psi\rangle$ and $|\Psi'\rangle$ are isomorphic.*

Proof. Let $|\Psi'\rangle = (L \otimes L \otimes L) |\Psi\rangle$ for some invertible map L . Then let $\langle \zeta' | = \langle \zeta | L^{-1}$ and $\langle \Phi' | = \langle \Phi | (L^{-1} \otimes L^{-1})$. It is straightforward to verify the Frobenius state axioms, and constructing the CFA as in Theorem 8.3.3, we have:



\square

8.3.1 Classification of Qubit Frobenius States

In section 3.2.2, we defined special and anti-special commutative Frobenius algebras. These are both defined as CFAs with one additional axiom.



We show in this section that $|GHZ\rangle$ and $|W\rangle$, the two canonical SLOCC-maximal tripartite qubit states, are both Frobenius states. Furthermore, the specialness and anti-specialness condition serve to distinguish the (symmetric) SLOCC-classes of these two states.

It will be convenient to work with the unnormalised versions of $|GHZ\rangle$ and $|W\rangle$.

$$|GHZ\rangle = |000\rangle + |111\rangle \quad |W\rangle = |100\rangle + |010\rangle + |001\rangle$$

For the Frobenius state $|GHZ\rangle$, fixing $\langle \xi | := \langle 0 | + \langle 1 |$ induces the following CFA, which we shall refer to as \mathcal{G} .

$$\begin{aligned} \begin{array}{c} \circlearrowleft \\ \downarrow \end{array} &= |0\rangle \langle 00| + |1\rangle \langle 11| & \begin{array}{c} \circlearrowright \\ \downarrow \end{array} &= \sqrt{2} |+\rangle = |0\rangle + |1\rangle \\ \begin{array}{c} \downarrow \\ \circlearrowright \end{array} &= |00\rangle \langle 0| + |11\rangle \langle 1| & \begin{array}{c} \downarrow \\ \circlearrowleft \end{array} &= \sqrt{2} \langle +| = \langle 0| + \langle 1| \end{aligned} \quad (8.8)$$

We can verify that \mathcal{G} is *special*.

$$\begin{array}{c} \circlearrowleft \\ \downarrow \end{array} \circ \begin{array}{c} \downarrow \\ \circlearrowright \end{array} = |0\rangle \langle 0| + |1\rangle \langle 1| = 1_{\mathbb{C}^2}$$

For the Frobenius state $|W\rangle$, fixing $\langle \xi | := \langle 0 |$ induces the following CFA, called \mathcal{W} .

$$\begin{aligned} \begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array} &= |1\rangle \langle 11| + |0\rangle \langle 01| + |0\rangle \langle 10| & \begin{array}{c} \bullet \\ \downarrow \end{array} &= |1\rangle \\ \begin{array}{c} \bullet \\ \downarrow \\ \bullet \\ \downarrow \end{array} &= |00\rangle \langle 0| + |01\rangle \langle 1| + |10\rangle \langle 1| & \begin{array}{c} \bullet \\ \downarrow \end{array} &= \langle 0| \end{aligned} \quad (8.9)$$

Computing the partial traces of multiplication and comultiplication, we get:

$$\begin{aligned} \begin{array}{c} \circlearrowleft \\ \downarrow \end{array} &= \text{Tr}^{\mathbb{C}^2} \left(\begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array} \right) = \sum_i (1 \otimes \langle i |) (|00\rangle \langle 0| + |01\rangle \langle 1| + |10\rangle \langle 1|) |i\rangle = 2 |0\rangle \\ \begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array} &= \text{Tr}^{\mathbb{C}^2} \left(\begin{array}{c} \circlearrowleft \\ \downarrow \end{array} \right) = \sum_i \langle i | (|1\rangle \langle 11| + |0\rangle \langle 01| + |0\rangle \langle 10|) (1 \otimes |i\rangle) = 2 \langle 1| \end{aligned}$$

We can then verify that \mathcal{W} is *anti-special*.

$$\begin{array}{c} \circlearrowright \end{array} \otimes \left(\begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array} \circ \begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array} \right) = 2(|0\rangle \langle 1| + |0\rangle \langle 1|) = (2|0\rangle)(2\langle 1|) = \begin{array}{c} \circlearrowleft \\ \downarrow \end{array} \circ \begin{array}{c} \bullet \\ \downarrow \end{array}$$

The next theorem is a straightforward consequence of the classification of tripartite qubit states and Theorem 8.3.4.

Theorem 8.3.5. *Any SLOCC-maximal, symmetric state $|\Psi\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2$ is a Frobenius state. It is SLOCC-equivalent to $|GHZ\rangle$ (resp. $|W\rangle$) if and only if the associated commutative Frobenius algebra is special (resp. anti-special).*

Proof. We first show $|\Psi\rangle$ is a Frobenius state. Since it is SLOCC-maximal, it must be SLOCC-equivalent to GHZ or W. Since it is a symmetric qubit state, it must be *symmetrically* SLOCC-equivalent to GHZ or W by Proposition 8.1.10. Therefore it is a Frobenius state.

If $|\Psi\rangle$ is SLOCC-equivalent to GHZ, its induced CFA is isomorphic to \mathcal{G} by Theorem 8.3.4, so it must be special. If $|\Psi\rangle$ is SLOCC-equivalent to W, its induced CFA is isomorphic to \mathcal{W} , so it must be anti-special.

Conversely, let $|\Psi\rangle$ be the induced Frobenius state of a special commutative Frobenius algebra \mathcal{S} . It must be SLOCC-equivalent to GHZ or W. Suppose it were to be SLOCC-equivalent to W, then \mathcal{S} is isomorphic to an anti-special commutative Frobenius algebra, which is a contradiction for dimensions > 1 . So, $|\Psi\rangle$ must be SLOCC-equivalent to GHZ. The result follows similarly for ACFA's. \square

8.3.2 Classification of Frobenius States for Higher Dimensions

We can push this classification a bit farther, but less is known in higher dimensions.

Example 8.3.6. We can produce D -dimensional analogues of the GHZ state and the W state.

$$\begin{aligned} |\text{GHZ}^{(D)}\rangle &= \sum_i |iii\rangle \\ |\text{W}^{(D)}\rangle &= |100\rangle + |010\rangle + |001\rangle + \sum_i (|0ii\rangle + |i0i\rangle + |ii0\rangle) \end{aligned}$$

Theorem 8.3.7. $|\text{GHZ}^{(D)}\rangle$ and $|\text{W}^{(D)}\rangle$ are both Frobenius states for any dimension. The induced CFA for $|\text{GHZ}^{(D)}\rangle$ is always special and the induced CFA for $|\text{W}^{(D)}\rangle$ is always anti-special.

Proof. For $|\text{GHZ}^{(D)}\rangle$, the associated effect in $(\mathbb{C}^D)^*$ is $\sum_i \langle i|$ and for $|\text{W}^{(D)}\rangle$, it is $\langle 0|$. The rest of the structure is uniquely determined, and the verification of SCFA and ACFA axioms is straightforward. \square

Theorem 8.3.8. For any D -dimensional SCFA, the induced Frobenius state is SLOCC-equivalent to $|\text{GHZ}^{(D)}\rangle$.

Proof. Let $(\mathcal{H}, \mu, \eta, \delta, \epsilon)$ be an SCFA. Then so too is $(\mathcal{H}^*, \delta^*, \epsilon^*, \mu^*, \eta^*)$. Any special Frobenius algebra over \mathbb{C} is semisimple, and any commutative semisimple algebra over \mathbb{C} is isomorphic to the direct sum of D copies of (\mathbb{C}, \cdot) . So, for a (not necessarily orthonormal) basis $\{|u_i\rangle\}$, μ has the form:

$$\delta :: |u_i\rangle \mapsto |u_i, u_i\rangle$$

For some arbitrary $\eta = \sum_i \alpha_i |u_i\rangle$, the tripartite state is:

$$|\Psi\rangle = (1 \otimes \delta) \circ \delta \circ \eta = \sum_i \alpha_i |u_i, u_i, u_i\rangle$$

For $|\Psi\rangle$ to be SLOCC maximal, all of the scalars α_i must be non-zero. Let L be defined as:

$$L :: |u_i\rangle \mapsto \frac{1}{\alpha_i} |u_i\rangle$$

Then, applying L to any of the subsystems yields

$$|\Psi'\rangle = \sum_i |u_i, u_i, u_i\rangle$$

which is clearly SLOCC-equivalent to $|\text{GHZ}^{(D)}\rangle$. \square

A complete classification of ACFA is not yet complete. For dimensions $D < 6$, there are relatively few commutative algebras, up to isomorphism, so it is feasible to enumerate them and check which extend to ACFA. For $D < 4$, there is only one ACFA, and its Frobenius state is SLOCC-equivalent to $|\text{W}^{(D)}\rangle$. However, for $D = 4$, there are already two non-isomorphic ACFA, so the classification of these types of Frobenius states may be more difficult in general.¹ A classification of *all* Frobenius states for $D < 6$ is in progress, and will be included in a forthcoming sequel to [19].

8.4 A Graphical Theory for Entanglement

We now look at some of the attributes of the pair of commutative Frobenius algebras corresponding to the GHZ and W states. $\mathcal{G} = (\downarrow, \uparrow, \downarrow, \uparrow)$ is a \dagger -special commutative Frobenius algebra, so it has an orthonormal basis of classical points. For $\mathcal{W} = (\bullet, \bullet, \bullet, \bullet)$, note that \bullet defines a partial monoid over the classical points of \mathcal{G} , and \bullet a partial monoid over their adjoints. In other words, $|i\blacktriangle j\rangle$ and $\langle i\blacktriangledown j|$ defined as follows must either be classical points for \mathcal{G} or 0.

To handle the case where $i\blacktriangle j = \perp$ or $i\blacktriangledown j = \perp$, we use the following notation for “undefined points”:

Since \mathcal{W} is *not* a \dagger -CFA, it is not necessarily true that $(|i\blacktriangle j\rangle)^\dagger = \langle i\blacktriangledown j|$. We will shortly see that these two partial monoids are isomorphic, but they are never identical for non-trivial ACFA. The other thing to note is that both the unit and the anti-unit are proportional to classical points of \mathcal{G} .

¹Thanks to Alex Merry for performing these calculations.

We can build up these features abstractly. First, we generalise several results from section 7.2 to the case where one of the Frobenius algebras is not \dagger -special. The next definition provides a condition for CFA to form a partial monoid over an orthonormal basis.

Definition 8.4.1. An arbitrary commutative Frobenius algebra $(\blacktriangledown, \blacktriangle, \blacktriangleright, \blacktriangleleft)$ is said to be *closed* on a \dagger -special commutative Frobenius algebra $(\circlearrowleft, \circlearrowright, \circlearrowup, \circlearrowdown)$ if the following equations hold:

For the rest of this section, let $(\blacktriangledown, \blacktriangle, \blacktriangleright, \blacktriangleleft)$ be a commutative Frobenius algebra closed on a \dagger -SCFA $(\circlearrowleft, \circlearrowright, \circlearrowup, \circlearrowdown)$.

Theorem 8.4.2. For a commutative Frobenius algebra $(\blacktriangledown, \blacktriangle, \blacktriangleright, \blacktriangleleft)$, closed on $(\circlearrowleft, \circlearrowright, \circlearrowup, \circlearrowdown)$, the dualiser is unitary and self-adjoint. Furthermore, it is a permutation of the classical points of \circlearrowup .

Proof. We can compute caps and cups in terms of \blacktriangle and \blacktriangledown . It follows from Definition 8.4.1 that $\blacktriangle = |u\rangle$ is a classical point for \circlearrowup and $\blacktriangledown = \langle c|$ is the adjoint of a classical point. So:

$$\blacktriangle = \sum_{i \blacktriangledown j = u} |i, j\rangle \quad \blacktriangledown = \sum_{i \blacktriangle j = c} \langle i, j|$$

From these, we can compute the dualiser and its inverse.

$$d = \text{cup with tick} = \sum_{i \blacktriangledown j = u} |i\rangle \langle j| \quad d^{-1} = \text{cup} = \sum_{i \blacktriangle j = c} |i\rangle \langle j|$$

Written in the basis given by \circlearrowup , these are both binary matrices (i.e. matrices whose entries are all either 0 or 1). But then, the only binary matrices whose inverses are also binary matrices are the permutations, so d must be a permutation, and $d^{-1} = d^\dagger$ its inverse permutation. The fact that this map is self-adjoint follows from the symmetry of caps and cups. Evaluating the cup at classical points, we have:

□

For the remainder of this chapter, we will represent the dualiser by placing a tick on an edge.

$$\dagger = \text{loop with dot} = \text{loop with dot}$$

The dualiser generates a monoid isomorphism from \blacktriangle to \blacktriangledown :

$$\text{triangle } i, j \text{ with dot} = \text{triangle } i, j = \text{triangle } i, j \text{ with dot} = \text{triangle } i, j \text{ with dot} = \text{triangle } i, j \text{ with dot} \quad (8.10)$$

Definition 8.4.3. Let $\mathcal{S} = (\text{cup}, \text{cap}, \text{cup with dot}, \text{cap with dot})$ be a \dagger -SCFA and let $\mathcal{A} = (\text{dot with cup}, \text{dot with cap}, \text{dot with cup with dot}, \text{dot with cap with dot})$ be an ACFA. Then $(\mathcal{S}, \mathcal{A})$ is called a *GW-pair* if \mathcal{A} is closed on \mathcal{S} and the following equations hold:

$$\text{cup with dot} = \text{dot with cap} \quad \text{cup} = \text{dot with cap with dot}$$

Technically, we only have to require that dot with cap be *proportional* to a classical point. Then, the fact that $\text{dot with cap} \circ \text{dot with cup} = \text{cup} = D$ uniquely fixes the scalar factor. Furthermore, we can prove that the dualiser interchanges the unit and anti-unit.

Lemma 8.4.4. *The following equations hold for any GW-pair:*

$$\text{cup} (\text{dot with cap})^\dagger = \text{dot with cap} \quad \text{cup} (\text{dot with cup})^\dagger = \text{dot with cup} \quad \text{cup} \text{dot with cap} = \text{dot with cap}$$

Proof. By Definition 8.4.3, dot with cap and dot with cup correspond to classical points for cup with dot , up to scalar. By closure, dot with cap and dot with cup are also classical points. Since $\text{dot with cap} \circ \text{dot with cup} = \text{cup} \neq 0$, then $(\text{dot with cap})^\dagger$ and dot with cup must be proportional. By closure, dot with cap is *equal* to a classical point, hence it is normalised. So $\text{cup} (\text{dot with cap})^\dagger = \text{dot with cap}$. The second equation holds similarly. We can also show that $\text{dot with cap} = (\text{dot with cup})^{\circ T}$:

$$\text{dot with cap} = \text{loop with dot} = \text{loop with dot}$$

Since dot with cup is the adjoint of a classical point, $(\text{dot with cup})^\dagger = (\text{dot with cap})^{\circ T} = \text{dot with cap}$. The final equation then follows. \square

A consequence of this lemma is that the monoids \blacktriangle and \blacktriangledown cannot be equal for dimensions $D \geq 2$. By anti-specialness and equation (8.10), that would imply the identity was rank 1. In [19], Coecke and Kissinger identified four axioms for interacting GHZ- and W-like states.

$$\begin{aligned} \text{(i.)} \quad \dagger &:= \text{loop with dot} = \text{loop with dot} & \text{(ii.)} \quad \dagger &= \text{cup with dot} \\ \text{(iii.)} \quad \text{cup with dot} &= \text{dot with cap} & \text{(iv.)} \quad \text{cup} &= \text{dot with cap with dot} \end{aligned}$$

Theorem 8.4.5. *Any GW-pair satisfies axioms (i.)-(iv.).*

Proof. Axioms (i.) and (ii.) are consequences of Theorem 8.4.2. Axiom (iii.) is by definition, and axiom (iv.) is part of Lemma 8.4.4. \square

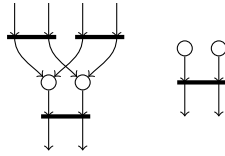
It is known that not all SCFA/ACFA pairs in **Rel** satisfy the closure identities, so it is likely that the GW-pair conditions are strictly stronger than axioms (i.)-(iv.). However, the Frobenius states $\left| \text{GHZ}^{(D)} \right\rangle$ and $\left| W^{(D)} \right\rangle$ provide at least one example of a GW-pair for any dimension in **FHilb**.

8.4.1 Symmetric Modules of an SCFA and Distributivity

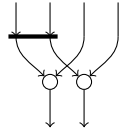
For any Hilbert space \mathcal{H} , we can form the space $\mathcal{H} \otimes_S \mathcal{H}$ of *symmetric* vectors in $\mathcal{H} \otimes \mathcal{H}$. There is a canonical projection from an arbitrary vector in $\mathcal{H} \otimes \mathcal{H}$ onto a symmetric vector in $\mathcal{H} \otimes_S \mathcal{H}$, called the *symmetriser map* S_2 :

$$\text{---} := \frac{1}{2} \left(\begin{array}{c} \downarrow \downarrow \\ \downarrow \downarrow \end{array} + \begin{array}{c} \downarrow \downarrow \\ \downarrow \downarrow \end{array} \right)$$

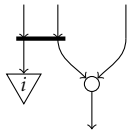
Any monoid (\mathcal{H}, μ, η) can be extended to a monoid $(\mathcal{H} \otimes_S \mathcal{H}, \mu_S, \eta_S)$ by using the symmetriser.



There is a canonical μ_S -module over the whole space $\mathcal{H} \otimes \mathcal{H}$. It is essentially the regular module of μ_S , but without the symmetriser maps on the right.



We shall refer to this as the *extended regular module* x_μ . For any SCFA, there is also a μ_S -module $k_{\langle u_i \rangle}$ over \mathcal{H} for every classical point $|u_i\rangle$.



For a GW-pair, $\langle c| := \blacklozenge$ is the adjoint of a classical point, so it induces a μ_S -module on \mathcal{H} . We call a GW-pair *distributive* if the multiplication $\blacklozenge : \mathcal{H} \otimes \mathcal{H} \rightarrow \mathcal{H}$ is a μ_S -module homomorphism from $(\mathcal{H} \otimes \mathcal{H}, x_\mu)$ to $(\mathcal{H}, k_{\langle c|})$.

Definition 8.4.6. A special commutative Frobenius algebra $(\blacklozenge, \circlearrowleft, \circlearrowright, \circlearrowright, \circlearrowleft)$ and an anti-special commutative Frobenius algebra $(\blacklozenge, \bullet, \blacklozenge, \bullet)$ form a *distributive GW-pair* if they are a GW-pair and:

$$\begin{array}{c} \text{---} \bullet \\ \downarrow \downarrow \\ \downarrow \downarrow \end{array} = \begin{array}{c} \text{---} \\ \downarrow \downarrow \\ \downarrow \downarrow \end{array} \bullet \tag{8.11}$$

We refer to this condition as distributivity, because it resembles the distributive law for rings. To see this most clearly, consider arbitrary vectors $|a\rangle, |b\rangle, |c\rangle \in \mathcal{H}$. Noting that $|a\rangle \otimes |a\rangle$ is symmetric, we can prove the following equation:

Ignoring scalars, and writing “+” for \bullet and “ \cdot ” for \circ , this equation becomes:

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

It reduces to the distributive law familiar from arithmetic. A slightly different, but equivalent way to see this is \bullet copies phases for \circ , up to a scalar.

This is similar to the case for strongly complementary observables, as in equation (7.5) from section 7.2. However, the condition here is much stronger, because \bullet copies *arbitrary* phases for \circ , rather than just those corresponding to classical points.

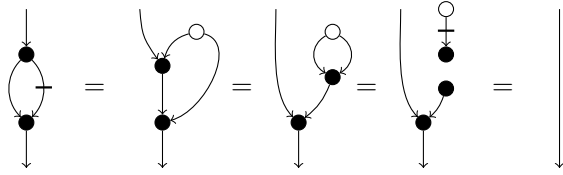
Example 8.4.7. The Frobenius algebras \mathcal{G}, \mathcal{W} defined in section 8.3.1 form a distributive GW-pair.

The distributive law implies many identities between \bullet and \circ . In the next lemma are two equations that we shall find useful in the next section.

Lemma 8.4.8. For any distributive GW-pair, the following equations hold:

Proof. The first equation follows from distributivity, noting that \circ is symmetric.

The second equation then follows from the first one.



□

8.4.2 Universality

Returning to the example of the GHZ and W Frobenius algebras, we treat $(\downarrow, \uparrow, \downarrow, \uparrow)$ and $(\downarrow, \uparrow, \downarrow, \uparrow)$ as generators for quantum circuits and look at their computational power. While they are not gates themselves, as they are not unitary, we shall soon see that we can think of certain gates as being composed from these generators, as we did for the Z/X calculus in section 7.3.1. Alternatively, one can think of the GW-pair maps as stochastic gates, prepared using post-selection or some more sophisticated measurement-based scheme.

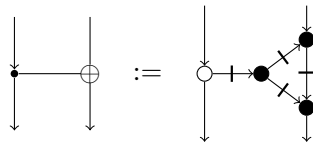
First, note that the dualiser interchanges $\downarrow = |1\rangle$ and $\uparrow = |0\rangle$, so it serves as a NOT (i.e. Pauli X) gate.

$$\uparrow \circ \downarrow = \uparrow \quad \uparrow \circ \uparrow = \downarrow$$

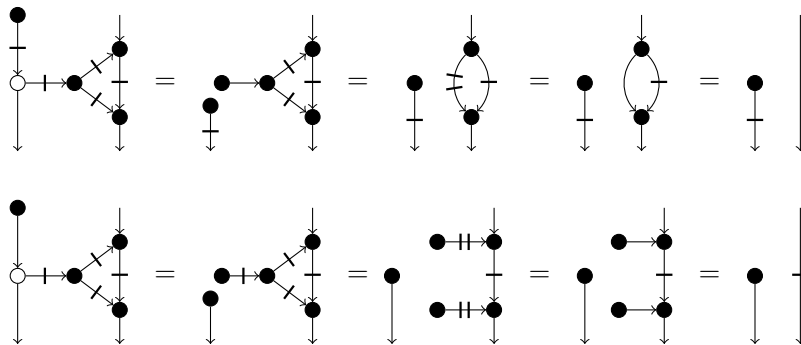
While the GHZ dot copies both \downarrow and \uparrow , the W dot acts like a “controlled” copy.



With this behaviour in mind, we can use these generators to build a CNOT gate.



We can verify that this is indeed a CNOT gate by rewriting.



Theorem 8.4.9. *The generators of \mathcal{G} and \mathcal{W} , along with single-qubit states, are universal for quantum computation.*

Proof. We have already illustrated the construction of a CNOT gate. To complete the proof, it suffices to show that we can apply arbitrary single-qubit unitaries. We can actually do better than this by showing we can apply arbitrary single-qubit linear maps. We can write a general single-qubit diagonal matrix as a GHZ phase. Let $|\psi_1\rangle = a|0\rangle + b|1\rangle$.

$$\begin{array}{c} \triangle \psi_1 \\ \curvearrowright \\ \circ \\ \downarrow \end{array} = a|0\rangle\langle 0| + b|1\rangle\langle 1| = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \quad (8.13)$$

For $|\psi_2\rangle = c|0\rangle + |1\rangle$, we can construct an arbitrary unit-diagonal upper triangular matrix as a W phase.

$$\begin{array}{c} \triangle \psi_2 \\ \curvearrowright \\ \bullet \\ \downarrow \end{array} = |0\rangle\langle 0| + c|0\rangle\langle 1| + |1\rangle\langle 1| = \begin{pmatrix} 1 & c \\ 0 & 1 \end{pmatrix} \quad (8.14)$$

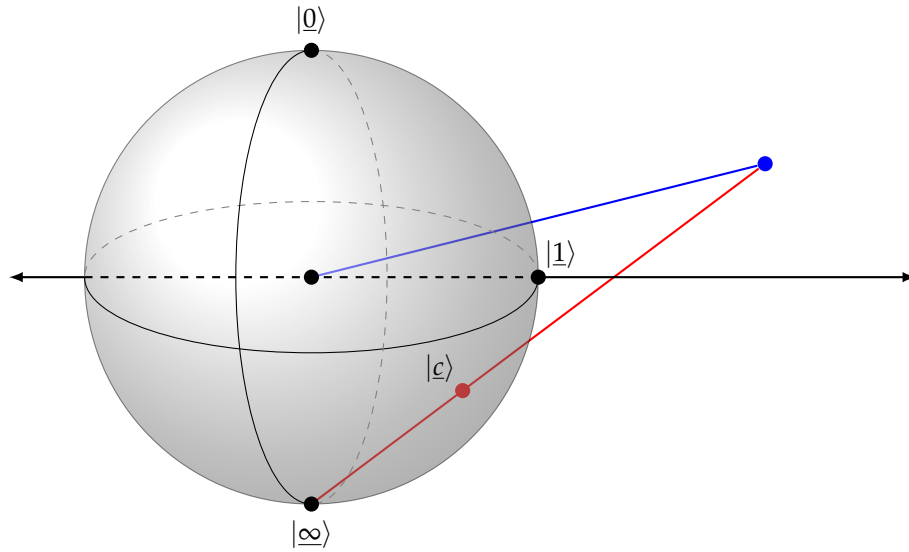
For $|\psi_3\rangle = d|0\rangle + |1\rangle$, we can construct an arbitrary unit-diagonal lower triangular matrix by applying the dualiser.

$$\begin{array}{c} \triangle \psi_3 \\ \curvearrowright \\ \bullet \\ \dagger \end{array} = |0\rangle\langle 0| + d|1\rangle\langle 0| + |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ d & 1 \end{pmatrix} \quad (8.15)$$

Any linear map decomposes as $M = PLDU$, where P is a permutation, L is a unit-diagonal lower triangular matrix, D is diagonal, and U is a unit-diagonal upper triangular matrix. Since the only permutations on \mathbb{C}^2 are the identity and \dagger , we can construct any single-qubit map using \dagger and the maps above. \square

8.4.3 Arithmetic on the Complex Projective Line

It is a well-known fact that points on the Bloch sphere correspond to points on the complex projective line. Any state $a|0\rangle + b|1\rangle$ where $b \neq 0$ can be represented, up to a scalar by the quotient $\frac{b}{a}$. Defining $|\frac{b}{a}\rangle := |0\rangle + \frac{b}{a}|1\rangle$ and $|\infty\rangle := |1\rangle$, we can cover the entire Bloch sphere. Then, the usual projection of a sphere on to \mathbb{CP}^1 takes these states to their corresponding points in \mathbb{CP}^1 .



Under this correspondence, we can see that the algebra induced by the W state is addition on \mathbf{CP}^1 and the GHZ algebra is multiplication. Before we illustrate this, we define addition and multiplication on \mathbf{CP}^1 as commutative partial monoids. For $k_1, k_2 \in \mathbf{C}$, addition and multiplication are defined as usual. For the rest, let $k \in \mathbf{C}$ be a non-zero complex number, and let \perp represent undefined.

$$\begin{array}{ll}
 k \cdot \infty = \infty & k + \infty = \infty \\
 0 \cdot \infty = \perp & 0 + \infty = \infty \\
 \infty \cdot \infty = \infty & \infty + \infty = \perp
 \end{array} \tag{8.16}$$

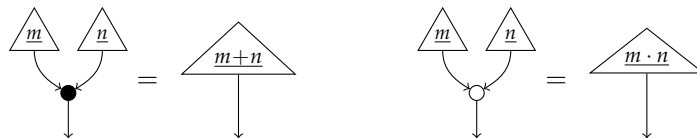
Intuitively, these are addition and multiplication operations for “formal fractions” over \mathbf{C} . That is, equivalence classes of pairs of complex numbers:

$$|(d, n)| = \{(d, n) \sim (\lambda d, \lambda n) : \lambda \in \mathbf{C} - \{0\}\}$$

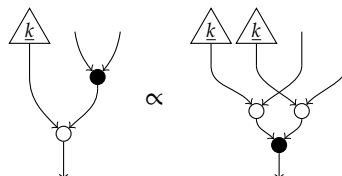
Letting $\infty := (0, 1)$ and $\perp := (0, 0)$, we can reproduce the above multiplication tables with:

$$|(d_1, n_1)| + |(d_2, n_2)| = |(d_1 d_2, n_1 d_2 + n_2 d_1)| \quad |(d_1, n_1)| \cdot |(d_2, n_2)| = |(d_1 d_2, n_1 n_2)|$$

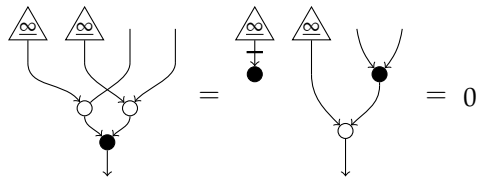
Using the convention that $|\perp\rangle = 0$, it is straightforward to verify the following equations.



It follows from equation 8.11 that the finitary points in \mathbf{CP}^1 distribute over addition. That is, for $k \in \mathbf{C}$:



Distributivity fails for ∞ , as is usually the case when formally introducing points at infinity.



The relationship between GHZ states, W states and the arithmetic of fractions is explored in detail by Coecke, Kissinger, Merry, Roy in [20].

Part III

Automation

Chapter 9

Automating String Graph Rewriting: Quantomatic

The Quantomatic Project [40] provides a set of tools for working with string graphs and string graph rewrite systems. It is divided into three parts: QuantoCore, QuantoGUI, and QuantoCoSy. The first is called QuantoCore, which is an ML library that does most of the work in representing, manipulating, and rewriting string graphs. QuantoCore uses three basic kinds of files.

1. ***.graph** files store string graphs.
2. ***.theory** files store graphical theories. A graphical theory contains information about what kinds of vertices can occur in a string graph, what kinds of data can occur on vertices, and how that data should be matched.
3. ***.rules** files contain sets of string graph rewrite rules.

On top of QuantoCore, we have developed a graphical user interface called QuantoGUI. Currently, QuantoGUI can:

- create and edit string graphs (Figure 9.1) and string graph rewrite systems (Figure 9.2),
- search for rewrites in a selected subgraph and apply them manually (Figure 9.3),
- display animated normalisations of string graphs with respect to a rewrite system,
- do “fast-normalisation” of string graphs and only display the output, and
- interact with computer algebra systems to perform concrete calculations of string graphs as linear maps (Figure 9.4, see [39] for details).

While it is currently quite minimal, we intend to make QuantoGUI into the graphical analogue of a proof assistant. Just as Isabelle [56] or Coq [10] exposes a variety of techniques for constructing formal proofs with respect to a term rewrite system (i.e. an algebraic theory), Quantomatic aims

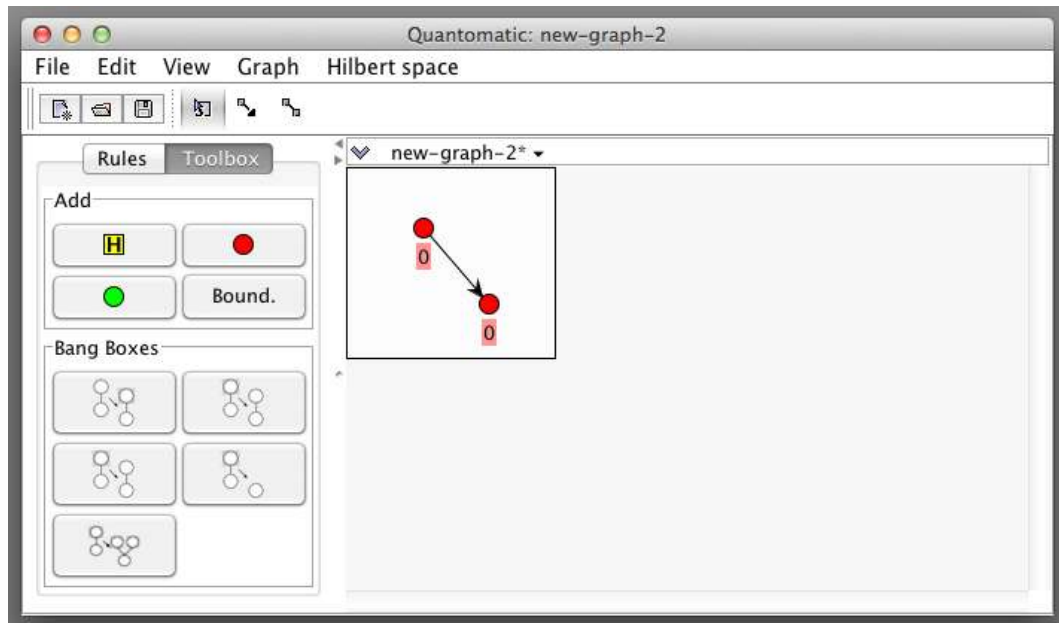


Figure 9.1: A string graph in Quantomatic

to do the same for string graph rewrite systems. The real power of systems like Isabelle and Coq comes not just from rewriting, but from the application of inductive reasoning techniques to prove theorems in first- or higher-order logic. These techniques do not always extend straightforwardly from terms to graphs. However graphs with quantifiers (defined by Rensink et al in [60]), pattern graphs, and graphs defined by grammars can give us the ability to reason about an infinite set of graphical equations simultaneously. We discuss how some of these more advanced techniques might work in section 10.1.5.

For more details about the Quantomatic project and to download the software itself, visit the project's web page at <http://sites.google.com/site/quantomatic>.

9.1 Conjecture Synthesis and QuantoCoSy

In this section, we will discuss QuantoCoSy, the third component to the Quantomatic project. It performs automated theory creation using a technique called *conjecture synthesis*.

One of the main goals of automated reasoning is to reproduce as much as possible on a machine the way a human mathematician thinks and works. Consider a situation where a mathematician has the following:

1. A set of generators for a new algebraic object X .
2. A concrete model or set of models for X .

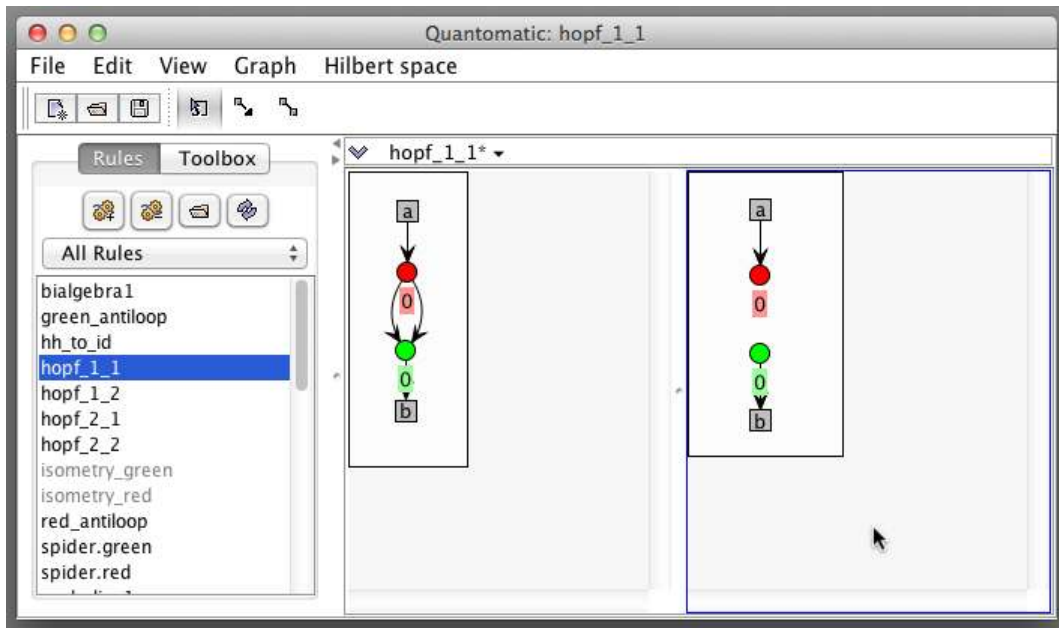


Figure 9.2: Editing a rule

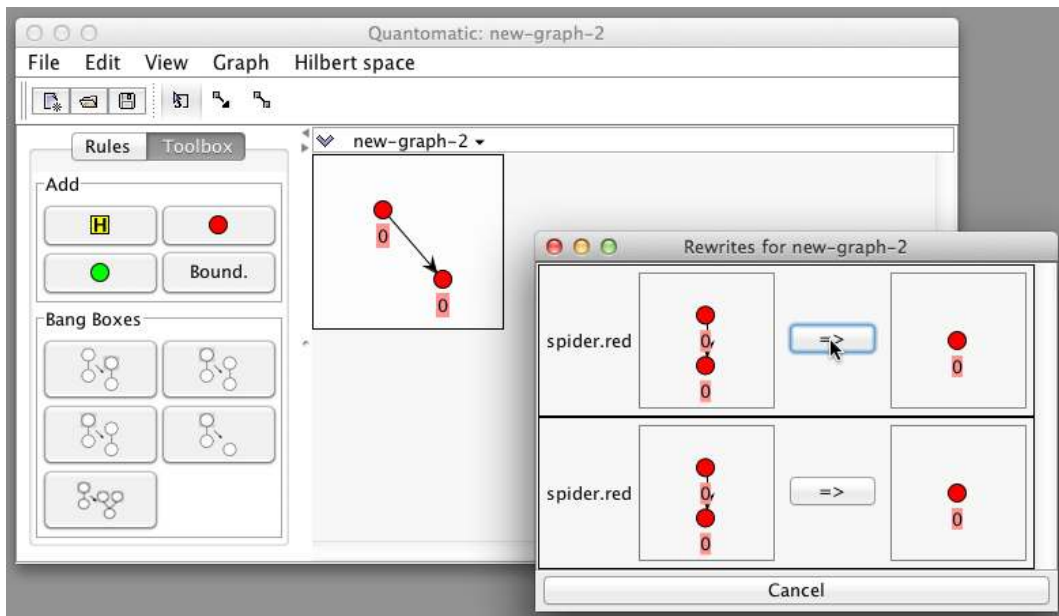
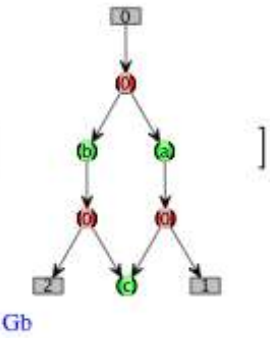


Figure 9.3: Applying a rewrite

```

In[554]:= term = Hilb [
    
]

Out[554]= T[zsp[c, 2, 0], id2[1], id2[1]].
          sig[0, 2, 1, 3].T[xsp[0, 1, 2], xsp[0, 1, 2]].
          T[zsp[a, 1, 1], zsp[b, 1, 1]].xsp[0, 1, 2].id2[1]

In[555]:= QuantoKill[]

```

Figure 9.4: Invoking Quantomatic from a Mathematica [65] notebook

Though X is not defined yet, the models that the mathematician has in hand are things that “morally” should be X ’s. For instance, if the mathematician were trying to develop an algebra for studying maximally entangled states, as in chapter 8, these would be a set of known maximally entangled states. From this data, the mathematician now seeks to *axiomatise* X .

One way he or she could start this process is to “plug-and-chug”. That is, the mathematician could plug these generators together randomly and see which compositions equal other compositions. In actuality, this process is not totally random, as the mathematician calls upon experience and a handful of helpful heuristics for seeking out likely equations.

Heuristic 1: seek familiarity. the mathematician may discover that these generators are actually satisfying some properties of a known algebraic object, such as a Hopf algebra. From this, the mathematician deduces that the generators are more likely to satisfy *the rest* of the identities of a Hopf algebra than they are to satisfy some other, randomly chosen identities.

Heuristic 2: avoid redundancy. Once the mathematician has a handful of identities, then while searching for new identities, he or she will avoid those which are trivially derivable from those already known. For instance, if the mathematician already knows $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$, it is redundant to consider the terms $a \cdot (b + c + d)$ and $(a \cdot b) + (a \cdot c) + (a \cdot d)$.

Heuristic 3: elegant identities are essential. Though the generators in question may exhibit large, complex, and asymmetric identities, the mathematician is treating the generators as merely one example of a more abstract mathematical object. His experience tells him that simpler identities tend to be the most crucial in characterising abstract mathematical objects.

Though there is theoretically no end to this procedure, the mathematician may hold a conviction, as per heuristic 3, that he or she will eventually find no more “interesting” identities above a

certain size. At this point, the procedure is effectively complete.

This section is about reproducing this process of (graphical) theory generation with a program called QuantoCoSy.

9.1.1 Conjecture Synthesis for Terms

Conjecture synthesis is a technique that automatically generates “reasonable” conjectures to test for an algebraic theory. This procedure for term-based theories was introduced by Johansson, Dixon, and Bundy in 2010 [33]. The tool that implements their technique is called IsaCoSy (for Isabelle COnjecture SYnthesis). A single round of their algorithm proceeds as follows:

1. Initialise a conjecture as an expression with *holes*, e.g. “ $(*) = (*)$ ”. Holes mark places where there is more expansion to do. They also come with certain constraints on what terms can be instantiated in them. Initially these are maximum size or depth constraints to guarantee termination, but they will get updated later.
2. Substitute holes with all possible terms-with-holes. This is done by a depth-first enumeration of possible substitutions, respecting the constraints on holes.
3. Once there are no holes, save the expression as a possible conjecture.
4. Perform (fast) post-filtering of conjectures that are obviously not true. IsaCoSy does this by using Isabelle’s fast counter-example search.
5. Try to prove the remaining conjectures using an automated proof search routine. When a proof is found, save the conjecture as a new rewrite rule.

When viewed as a single round, this looks like the naïve technique that one would use to go about searching for conjectures. However, the interesting part is how constraints are updated between rounds. A given round of the synthesis procedure will produce a set of true equations, E . We can turn these equations into rewrite rules by putting a reduction ordering on terms (see Definition 4.1.7 in section 4.1).

$$S = \{t_1 \rightarrow t_2 : ((t_1, t_2) \in E \vee (t_2, t_1) \in E) \wedge \omega(t_1) > \omega(t_2)\}$$

Let R be the set of all terms occurring on the LHS of rules in S . These are called *reducible expressions*, or *redexes*. For a rewrite rule $r = (t_1 \rightarrow t_2)$, any equation containing t_1 can already be proved using r and an equation containing t_2 . As conjectures involving redexes are redundant, they are never considered. We can cut them out of the search space by updating the constraints on holes, such that the search performed in step (2.) above never generates terms containing redexes.

While this is a fairly simple procedure, in practice, this can exponentially reduce the search space.

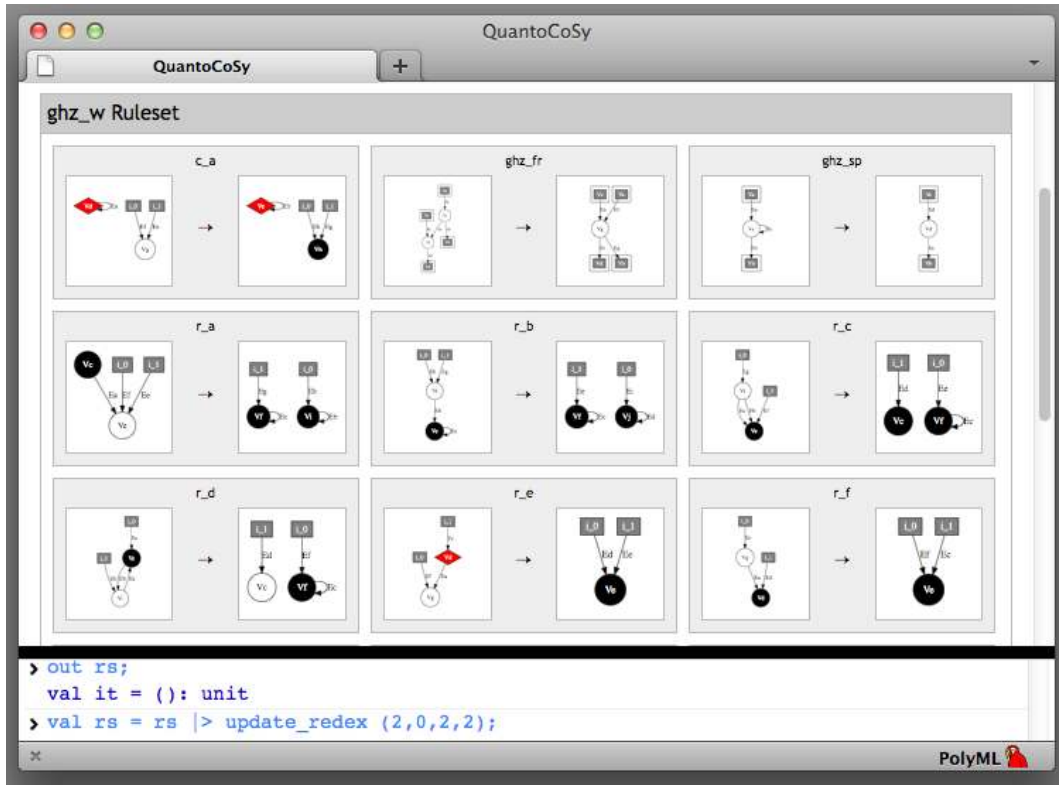
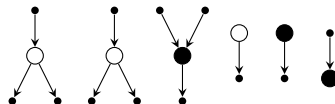


Figure 9.5: QuantoCoSy runs in Firefox using PolyChome [53], an extension for running Poly/ML in a web browser.

9.1.2 Adapting Conjecture Synthesis to String Graphs

The synthesis procedure for string graph identities is similar to that for terms. The procedure described in this section is implemented on top of QuantoCore with a program called QuantoCoSy (Figure 9.5).

In the term case, a single round of synthesis is parametrised by two natural numbers: the maximum term size (or term depth) and the maximum number of free variables occurring in the term. For string graphs, we parametrise a run with four natural numbers: the number of inputs M , the number of outputs N , the maximum number of box-vertices B , and the maximum number of pluggings P . We enumerate string graphs by starting with disconnected string graphs, i.e. graphs only containing box-vertices and their adjacent edges and wire-vertices. For instance:



The synthesis procedure for string graphs takes as input:

1. A string-graph signature T . For simplicity, we will focus on string graph signatures with only a single wire type.

2. An (m, n) -tensor for every box in T with m inputs and n outputs.
3. A function ω from string graphs to a well-ordered poset (P, \leq) . This will serve as a (candidate) reduction ordering.

Throughout the synthesis, we maintain a rewrite system S , and a set R of reducible string graphs. A single run given by natural numbers (M, N, B, P) consists of the following steps:

1. For all p such that $0 \leq p \leq P$, generate all disconnected string graphs with $M + p$ inputs and $N + p$ outputs, up to isomorphism.
2. For a disconnected string graph with $M + p$ inputs and $N + p$ outputs, there are $(M + p) \cdot (N + p)$ input/output pairs. Choose p of them to plug together. These are chosen so that the enumeration is exhaustive and minimises the occurrences of isomorphic string graphs (any remaining isomorphic graphs will be filtered out later). After each plugging, if a string graph contains an element of R as a subgraph, terminate that branch of the enumeration.
3. Evaluate the string graphs as tensors, performing a tensor contraction for every edge (c.f. the construction of \hat{F} in Theorem 5.5.10). Organise them into equivalence classes, up to scalar factors and permutations of inputs and outputs (which are stored with the string graphs). Filter out any remaining isomorphic graphs.
4. For each equivalence class C , identify a set $C' \subseteq C$ of minimal elements with respect to ω . Add any string graph in $C - C'$ to the set of reducible graphs R . Choose a string graph $s \in C'$ at random and add rules $t \rightarrow s$ to the rewrite system S for all $t \in C - C'$. Add rules in both directions $(s \rightarrow s', s' \rightarrow s)$ for the other minimal graphs $s' \in C' - \{s\}$.

We postpone filtering out isomorphic graphs until step (3.) because tensor contraction is fast, and two graphs will not be isomorphic unless they are in the same equivalence class. We choose ω such that step (4.) picks out as few graphs in $C' \subseteq C$ as possible. If C' is a singleton, all of the rewrites respect the reduction order. Rewrites that do not strictly decrease $\omega(G)$ (i.e. rewrites from an element of C' to another element of C') are called *congruences*. We can retain a terminating rewrite system if we throw out all of the congruences, but not without losing some information about the model. Therefore, a large portion of the refinement of this technique has to do with eliminating congruences or handling them in smarter ways (i.e. building them in to the graph representation).

We applied QuantoCoSy to generators of the GW-pair given in section 8.4. We preloaded the synthesis procedure with rewrites that merge two vertices of the same colour (i.e. the spider laws), and synthesised graphical identities for $B = 3$, $P = 3$, and $M + N \leq 3$. This yielded 223 rewrite rules, most of which were versions of the four axioms given in Theorem 8.4.5.

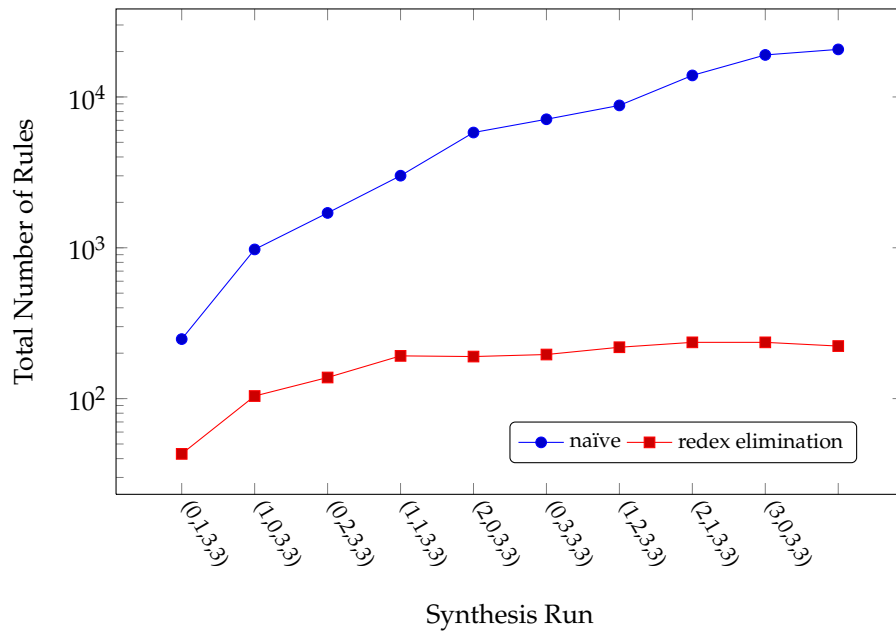


Figure 9.6: Rewrite rules synthesised from the generators of the GW-pair defined in section 8.4.

As was the case for terms, filtering out redexes has a huge impact on the number of string graphs that need to be checked. The naïve synthesis procedure with the same parameters yielded over 20,000 rewrite rules. In Figure 9.1.2, we plot the number of rewrite rules generated using a naïve graph enumeration algorithm against the number generated using the procedure above.

While the redex-elimination procedure yielded a much more manageable number of rewrite rules, a quick look at the rules by a human will show that many are still trivially consequences of each other. Therefore, there is still much work to be done in eliminating redundancy in rules and building more of the symmetries of a rewrite theory into the graphical representation itself. We discuss some of the ways in which we hope to accomplish this in sections 10.1.4 and 10.1.5.

Chapter 10

Conclusion

The main contributions of this dissertation fall under three categories: (1) the definition and properties of string graphs, (2) the application of string diagrams/string graphs to quantum computation, and (3) the automated generation and manipulation of string graph rewrite systems.

First, we defined string graphs and string graph rewrite systems using double-pushout graph rewriting. We also introduced the notion of composition for string graphs via certain pushouts called *pluggings* and defined a category whose morphisms are string graphs modulo a rewrite system, where categorical composition is defined using plugging. Using these rewrite categories, we constructed the free traced symmetric and the free compact closed categories over a monoidal signature. These results allow us to prove identities in an arbitrary symmetric traced category using graph rewriting.

In [24], Dixon and Kissinger proved that rewrite categories are equivalent to their topological analogues, as defined by Joyal and Street in [35]. Since free categories are defined up to equivalence, the results in this dissertation imply two of the missing “GTC-II” theorems. Namely, a category defined using string diagrams forms the free symmetric traced (or compact closed) category over a monoidal signature.

Next, we illustrated the application of diagrammatic languages to the study of quantum phenomena: namely complementary observables and multipartite entanglement. After reviewing Coecke and Duncan’s characterisation of complementarity using interacting Frobenius algebras, we proved several new results, including a complete classification of pairs of strongly complementary observables in a finite dimensional Hilbert space. This classification theorem showed a 1-to-1 correspondence between D -dimensional strongly complementary pairs of observables and Abelian groups of order D .

After this, we showed how certain kinds of highly entangled states, called Frobenius states, induce Frobenius algebras. Furthermore, the two canonical tripartite qubit states—GHZ and W —are distinguished by a simple condition on these Frobenius algebras: specialness or anti-specialness.




A Frobenius state in $\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2$ is SLOCC-equivalent to GHZ if and only if its induced commutative Frobenius algebra is special, and a Frobenius state is SLOCC-equivalent to W if and only if its induced CFA is anti-special.

Drawing on the interaction properties of these canonical qubit states, we introduced the theory of *GW-pairs*. These abstract the interaction properties of the Frobenius algebras associated with GHZ and W, and can be used to study arbitrary multipartite entangled states. Focusing on the specific case of GHZ and W, we showed that the generators of the GW-pair are universal for quantum computing and characterised the behaviour of the GHZ- and W-algebras in terms of (partial) arithmetic operations defined on the complex projective line $\mathbb{C}P^1$.

Finally, in part III, we introduced the Quantomatic project, which consists of tools for the automatic construction and manipulation of string graph rewrite theories. We illustrated how the process of conjecture synthesis introduced by Johansson, Dixon, and Bundy [33] can be adapted to the setting of string graphs, where each graphical generator is given a concrete valuation as a linear operator. Using this technique, it becomes practical to enumerate all of the graphical identities exhibited by a set of generators under composition for small- to medium-sized string graphs. These tools, along with the methods they employ, show great potential for changing the way we formulate and interact with a wide variety of theories involving interacting components.

10.1 Future Work

10.1.1 Classifying Frobenius states

In section 8.3.2, we gave a classification result for anti-special commutative Frobenius algebras of dimension 2 and for special Frobenius algebras of *any* dimension. The natural next step is to ask if a reasonable classification result can be constructed for ACFA of dimension $D \geq 3$. This classification is likely to be more difficult in the case of SCFA, where classification follows from the fact that there are not very many semisimple, commutative K -algebras. When K is an algebraically closed field, the only semisimple algebras are direct sums of K itself. In the case of ACFA of dimension $D > 1$, the vector  always generates a non-trivial nilpotent ideal, so non-trivial ACFA are never semisimple. However, it is our hope that anti-specialness will prove a strong enough condition on a Frobenius algebra to yield a straightforward classification.

We also intend to expand the classification results for Frobenius states on a different front: the classification of *arbitrary* Frobenius states for low dimensions (e.g. $D \leq 5$). This problem is tractable because there are relatively few commutative, unital algebras of dimension 5 or less. There is one

such algebra for dimension 1, two for $D = 2$, four for $D = 3$, nine for $D = 4$, and 20 for $D = 5$. For dimensions 3 and above, only some of these algebras extend to Frobenius algebras, and it becomes a straightforward task to enumerate those that are. The classification of Frobenius states for dimension 3 was completed this year by Honda [32]. As in the case for two dimensions, a Frobenius state up to SLOCC is uniquely determined by the rank of its loop map. Thus there is only Frobenius state corresponding to an SCFA (full rank), one corresponding to an ACFA (rank 1), and one corresponding to what Honda calls an *intermediate special commutative Frobenius algebra* (ICFA), which has a loop map of rank 2. The author, along with Coecke and Merry intend to incorporate this result into a complete classification of Frobenius states up to dimension 5 in the sequel to [19].

10.1.2 Super-qubits and the W state bialgebra

The W state exhibits some interesting properties that we have not yet fully explored. In section 3.2, we introduced the operation $(-)^{\circ T}$ of transposition relative to a particular Frobenius algebra. In the case of the GW-pair corresponding to the GHZ and W states, let $\Downarrow = \left(\begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array} \right)^{\circ T}$ and let $\Updownarrow = \left(\begin{array}{c} \downarrow \\ \bullet \\ \uparrow \end{array} \right)^{\circ T}$. The following then forms a bialgebra:

$$\widehat{\mathcal{W}} := (\mathbb{C} \oplus \mathbb{C}, \Downarrow, \Updownarrow, \begin{array}{c} \downarrow \\ \bullet \\ \downarrow \end{array}, \begin{array}{c} \downarrow \\ \bullet \\ \uparrow \end{array})$$

But there is a catch: it does not form a bialgebra in **FHilb**, but rather the category **SuperHilb** of *super-Hilbert spaces and even maps*.¹ This is the category where Hilbert spaces are graded into a “bosonic” and a “fermionic” part. Objects are \mathbb{Z}_2 -graded Hilbert spaces $H_0 \oplus H_1$ and morphisms are linear maps $f : H_0 \oplus H_1 \rightarrow H'_0 \oplus H'_1$ that respect the grading. This category is monoidally equivalent to the category of unitary representations of \mathbb{Z}_2 , but we use a different symmetry map that introduces a -1 factor when a “fermionic” element crosses another one:

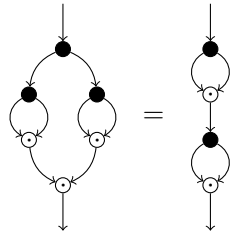
$$\sigma_{H_0 \oplus H_1, H'_0 \oplus H'_1}(|i\rangle \otimes |j\rangle) = \begin{cases} -|j\rangle \otimes |i\rangle & \text{if } |i\rangle \in H_1, |j\rangle \in H'_1 \\ |j\rangle \otimes |i\rangle & \text{otherwise} \end{cases}$$

For instance, over the graded space $\mathbb{C} \oplus \mathbb{C}$ of *super-qubits*, the swap map is defined by the following matrix:

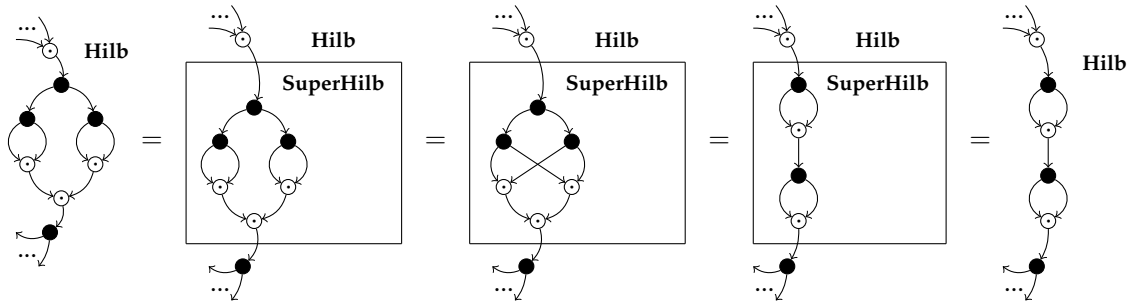
$$\sigma_{\mathbb{C} \oplus \mathbb{C}, \mathbb{C} \oplus \mathbb{C}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

Why is it interesting that $\widehat{\mathcal{W}}$ forms a bialgebra in **SuperHilb**, especially if **Hilb** is our primary category of interest? There is a faithful forgetful functor $U : \mathbf{SuperHilb} \rightarrow \mathbf{Hilb}$ that is strongly monoidal, but does not preserve symmetries. As a result, any *planar* diagrammatic identity we can prove in **SuperHilb** also holds in **Hilb**. For instance, we can prove the following equation for any commutative bialgebra.

¹Thanks to Jamie Vicary for pointing this out.

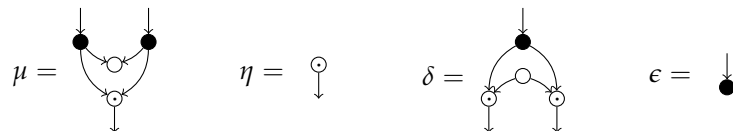


This is a bit of a contrived example, but it's a special case of a general result for commutative bialgebras. A commutative bialgebra diagram is uniquely determined by the number of forward-directed paths from each input to each output. In the above equation, there are exactly 4 distinct paths connecting the input to the output on both the LHS and the RHS. When morphisms exhibit certain properties in one “categorical context” but not in another, we express this graphically, using the functorial boxes defined by Melliès in [50]. Starting with a diagram in **Hilb**, we can draw a box around a sub-diagram as long as the diagram (1) is planar and (2) contains only morphisms in the image of U . We can then rewrite the elements inside the box as if they are in **SuperHilb**, possibly breaking planarity along the way. Then, as long as the diagram is ultimately planar, we can erase the box.



10.1.3 GW-pairs and strongly complementary observables

Aside from contrived examples we have given, one might wonder if there are *useful* planar equations satisfied by bialgebras. In 2008, Melliès showed that any commutative bialgebra whose comonoid is the transpose of its monoid, the following maps form a Frobenius algebra [49]:

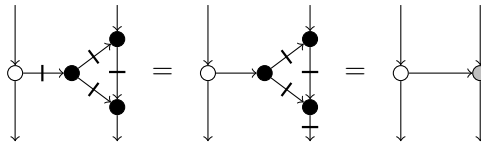


Recall that the monoid, comonoid, and Frobenius identities are all planar, so the above yield a Frobenius algebra in both **SuperHilb** and **Hilb**. Furthermore, in the case of the W state algebra, this Frobenius algebra is precisely the \dagger -SCFA for the X observable, defined in section 7.3. Expanding \downarrow and \circlearrowleft using the dualiser, we obtain the following expression for the Frobenius algebra

$(\mathbb{C}^2, \downarrow, \circlearrowleft, \circlearrowright, \circlearrowup, \circlearrowdown)$.

(10.1)

Under this encoding, we can relate the two constructions of the CNOT gate from sections 7.3.1 and 8.4.2.



So, we know that in the particular case of the GHZ/W Frobenius algebra pair, we can construct the Frobenius algebras for the strongly complementary observables Z and X . However the general relationship between GW-pairs and strongly complementary pairs is still unknown. However, we conjecture that the axioms of a distributive GW-pair subsume those of a strongly complementary pair.

Conjecture 10.1.1. Let $(\downarrow, \circlearrowleft, \circlearrowright)$ be a distributive GW-pair. Then, for a third Frobenius algebra $(\downarrow, \circlearrowleft, \circlearrowright, \circlearrowup, \circlearrowdown)$ defined as in (10.1) above, $(\downarrow, \circlearrowleft, \circlearrowright)$ forms a strongly complementary pair of \dagger -SCFAs.

Another area for future work is the conceptual understanding of strong complementarity. In section 7.2, we provided a complete classification for pairs of strongly complementary observables. While we have a clear idea of what strong complementary means mathematically, a physical interpretation of strong complementary is still missing. The fact that a particular pair of strongly complementary observables (Pauli Z and X) play such a central role in the study of complementarity in finite dimensions suggests that such an interpretation exists. As a first step toward finding this interpretation, we are looking for quantum protocols and theorems that rely crucially on certain forms of the bialgebra equations given in Definition 7.2.3.

10.1.4 Knuth-Bendix completion for string graphs

Knuth-Bendix completion is a procedure for turning terminating, non-confluent rewrite systems into terminating, confluent rewrite systems. It works by identifying *critical pairs* for a rewrite system. For a finite set of term rewrite rules, we can always identify a finite set of terms $s \in S$ that represent all of the “possible ways” in which the left-hand sides of two rewrite rules can overlap. A critical pair is then a pair of distinct, normalised terms t_1, t_2 that were both rewritten from some

such term s . A rewrite system is confluent precisely when it has no critical pairs. Knuth-Bendix completion takes a rewrite system R and a strict, total reduction ordering ω on terms, and operates as follows:

1. Compute all of the critical pairs for R .
2. For each critical pair (t_1, t_2) , add $t_1 \rightarrow t_2$ to R if $\omega(t_1) > \omega(t_2)$. Add $t_2 \rightarrow t_1$ otherwise.
3. Repeat until there are no critical pairs.

There are two possible outcomes for this procedure: (1) it halts and produces a confluent, terminating rewrite system, or (2) it keeps producing more and more rewrite rules forever. Since arbitrary word problems can be encoded as terminating term rewrite systems, there must exist some rewrite systems for which the Knuth-Bendix procedure does not halt. However, for many useful classes of rewrite systems, this procedure always halts, yielding a terminating, confluent rewrite system.

As an example, let $J \trianglelefteq K[X_1, \dots, X_n]$ be some ideal of a polynomial ring. It is a well-known fact that the ideal membership problem is decidable precisely when one can find a special set of polynomials generating J called a *Gröbner basis*. There is a natural way to consider a particular polynomial J_i as a rewrite rule on other polynomials.

$$X_1^2 X_2 + 4X_2 + 2 \quad \longrightarrow \quad (X_1^2 X_2 \rightarrow -4X_2 - 2)$$

This rule then rewrites certain polynomials P that are “matched” by J_i to $P - J_i$. Gröbner bases are then exactly those sets of polynomials generating J that, considered as rewrite systems, are terminating and confluent. A crucial tool for computer algebra systems is Buchberger’s algorithm, which derives Gröbner bases from arbitrary finite sets of polynomials. This algorithm is precisely Knuth-Bendix completion applied to polynomial rewrite systems [6].

Confluence for diagrammatic rewrite systems can be more subtle than term rewrite systems. For example, it was shown by Lafont that for general diagrammatic rewriting, finite diagrammatic rewrite systems could lead to infinite families of critical pairs exhibiting what he calls *global conflicts* [45]. However, this problem can be overcome by performing critical pair analysis on diagrams with “gaps”, as Mimram did in his thesis [51] to demonstrate a locally confluent presentation of $\mathbf{Mat}(\mathbb{N})$.

The source of the subtlety here lies in the fact that the diagrams considered by Lafont et al exist in arbitrary monoidal categories. Any additional categorical structure (like symmetries and duals) are treated “opaquely” as morphisms in a 2D grid. String graph rewrite systems rely crucially on the fact that the traced symmetric structure of the category is *built in* to the graphical representation of the string graph. Therefore much of the subtlety of the Lafont-style approach falls away (or more precisely, is absorbed in graph isomorphism) at the expense of generality.

In 2008, Kissinger defined a Knuth-Bendix procedure for diagrams of interacting commutative Frobenius algebras and applied it by hand to derive a confluent fragment of the Z/X -calculus [38]. Extending this to general string graph theories is straightforward, and its implementation in Quantomatic will be a useful tool both on its own and as a component of more sophisticated procedures.

10.1.5 Pattern graphs and graphical inductive reasoning

We often wish to work with graphical generators that have commutative inputs and outputs with variable arities. We can encode this into the usual string graph formalism by adding a box type for every possible arity and adding rewrite rules for commutativity. However, experience has showed us that there is much to be gained by encoding as much symmetry into the *representation* of an algebraic system as possible. For instance, in term rewrite theories, once a function is assumed to be commutative, its arguments are treated as a multiset, rather than an ordered list. We incorporate this into the theory of string graphs by introducing *string graphs with arities*.

Monoidal signatures are replaced by *signatures with arities* $\widehat{T} = (O, M, \text{dom}, \text{cod})$. For $P\mathbb{N}$ the powerset of \mathbb{N} , the functions:

$$\text{dom} : M \rightarrow w(O \times P\mathbb{N}) \quad \text{cod} : M \rightarrow w(O \times P\mathbb{N})$$

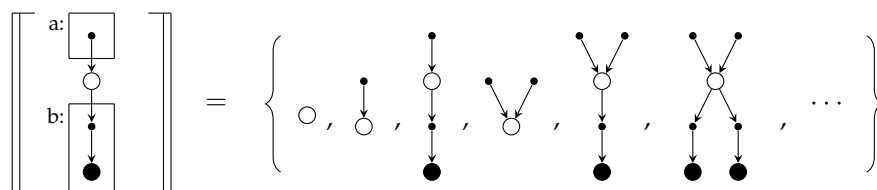
assign a morphism to a list of pairs $(o \in O, A \subseteq \mathbb{N})$. The element o is the type of the input or output as before, and the set A is the set of allowed arities. The requirement that the typing maps $\tau_G : G \rightarrow G_T$ for string graphs must be local isomorphisms around box vertices is replaced with the requirement that these maps respect arities. For an edge e in the typegraph, we require that e occurs with an allowed arity around every box-vertex $v \in B(G)$. Formally, for all box-vertices $v \in B(V)$ where $\tau_G(v) = f$ and all edge types $\text{in}_{f,i}, \text{out}_{f,j}$ such that $\text{dom}(f)[i] = (o, A)$ and $\text{cod}(f)[j] = (o', A')$ the following equations must hold:

$$\begin{aligned} |(\tau_G^v)^{-1}(\text{in}_{f,i})| &\in A \\ |(\tau_G^v)^{-1}(\text{out}_{f,j})| &\in A' \end{aligned} \tag{10.2}$$

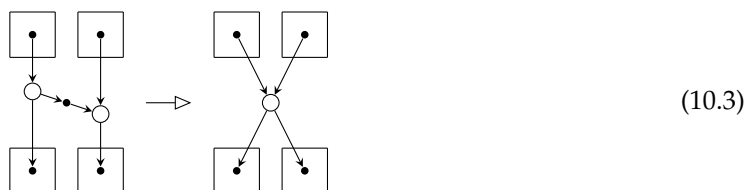
Using arities admits a great deal of flexibility in string graph signatures. The usual notion of string graphs without arities is recovered by letting all of the sets of allowed arities be $\{1\}$, because τ_G is a local isomorphism iff the inverse images defined in (10.2) are always of cardinality 1. The other extreme is where all of the sets $A = \mathbb{N}$, where any arity is allowed. This is incredibly useful when working with spiders, as defined in section 3.2.1. In fact, this is default mode for Quantomatic, as it was originally designed to work with “spider-based” graphical languages like those defined in part I. In between these two extremes, one could define optional (non-commutative) inputs and outputs by setting $A = \{0, 1\}$ and (fixed arity) commutative inputs and outputs by setting $A = \{k\}$.

One of the most useful things about having generators with variable arities (as opposed to just introducing new generators for every arity) is that we can define *pattern graphs*. These are graphs where certain portions of a graph (and their incident edges) can be duplicated any number of times. We define pattern graphs by introducing *!-boxes* (pronounced “bang boxes”) around string subgraphs. Intuitively, pattern graphs represent a set of concrete string graphs where each of the *!-boxes* can occur 0 or more times.

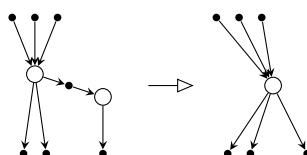
Example 10.1.2. A pattern graph with two *!-boxes*, and the set of concrete graphs it represents.



The real power of pattern graphs comes from the ability to define *pattern graph rewrite rules*. These consist of a pair of pattern graphs, and a suitable bijection between the *!-boxes* on the LHS and RHS such that we can represent an infinite set of valid string graph rewrite rules. For instance, a rewrite rule to merge two vertices of any arity can be expressed as:



A pattern graph rule can be *instantiated* into a concrete rule by replacing a single *!-box* with N copies of that *!-box* on the LHS and replacing its corresponding *!-box* with N copies on the RHS. For example, the following is an instance of the rule above:



Although a preliminary implementation of *!-boxes* already exists in Quantomatic, to assure its validity, we need to formalise string graphs with arities and pattern graph matching and rewriting within the framework of partial adhesive categories. In many ways this is a straight-forward task, but care must be taken when defining the correct notions of matching and rewriting.

We can also do rewriting on pattern graphs themselves. That is, we can use infinite sets of rules (i.e. pattern graph rewrite rules) to reason about infinite sets of graphs (i.e. pattern graphs).

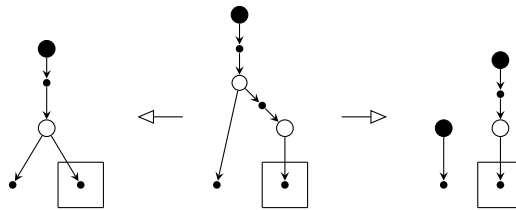
One of the most interesting applications for pattern graph rewriting is combining the conjecture synthesis procedure with Knuth-Bendix completion to automatically generate new pattern graph rewrite rules. IsaScheme, a tool for “scheme-based” conjecture synthesis [52], has already had some success in this area for term-based theories.

As it stands, the synthesis procedure can only discover new concrete rewrite rules, however, using Knuth-Bendix, it could automatically *combine* its pre-existing knowledge (in the form of an initial set of pattern graph rewrite rules) with its findings.

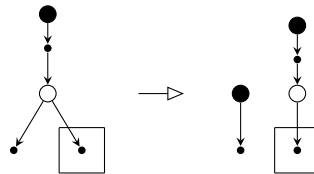
Consider a case where we initiate the synthesis procedure with the rewrite rule given by (10.3). At some point, it discovers a new identity:



If we perform Knuth-Bendix on a rewrite system, we obtain a critical pair.

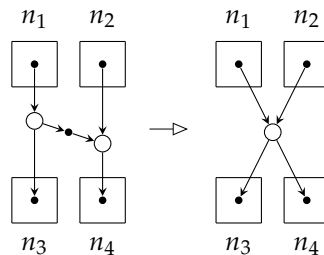


Then, under a suitable ordering on string graphs, we can consider the string graph on the right to be more reduced than the string graph on the left, so we introduce a new pattern-graph rewrite rule.



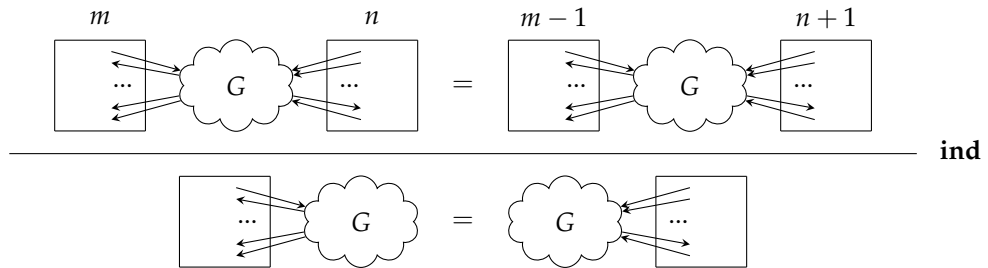
This rule is stronger than (10.4), and can be thought of as the “spiderised” version of that rule. Since this rule is stronger than the previous one, there are more reducible expressions in the rewrite system being synthesised, and hence a smaller search space for string graph enumeration. This suggests that incorporating a Knuth-Bendix step into the conjecture synthesis procedure could vastly improve its performance as well as generate fewer, more powerful graphical identities.

In addition to pure equational reasoning (i.e. rewriting proofs) with !-boxes, we can do some inductive reasoning as well. Suppose we extend the language of !-boxes, allowing them to be bound to expressions over natural numbers, possibly containing free variables.



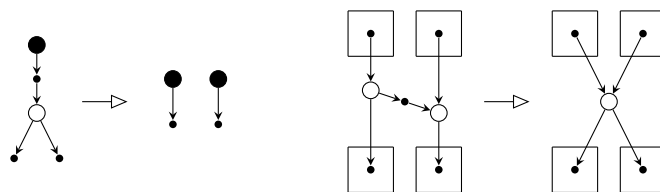
We interpret this rule as the set of all concrete rules where the i -th !-box is duplicated n_i times. For non-atomic expressions (i.e. expressions that are not just a single free variable or constant), it

could become tricky to prove that any substitution of free variables yields valid concrete rewrite rule without ambiguity. However, if this can be done even in limited cases, such a language allows one to define a notion of induction, in the form of an inference rule on graphical identities.

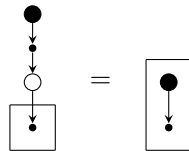


This rule says, “If one can convert a single copy of a !-box to a single copy of another !-box, then one can convert k copies of that !-box for any k .” Note that the base case is trivially satisfied: if we kill both !-boxes, we are left with $G = G$.

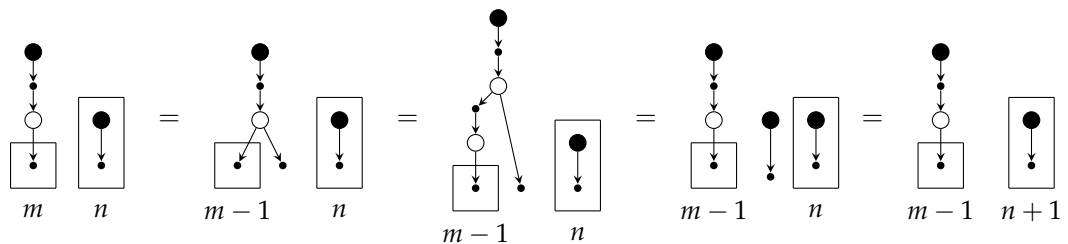
Example 10.1.3. Using the **ind** inference rule to prove a new pattern graph identity. Take the following rules as given:



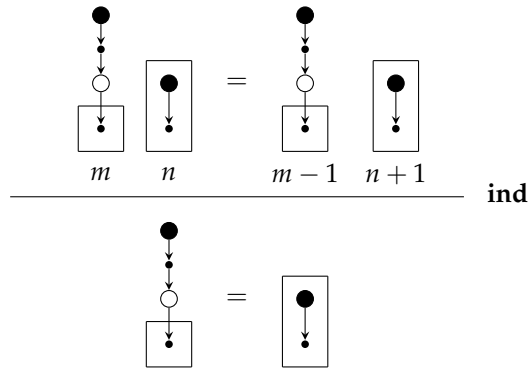
Then, we will use the induction principal to prove:



First, we derive the hypothesis for **ind** using rewriting.



The general rule is then constructed with an application of **ind**.



Note that this identity could not be produced using purely-equational means (e.g. using Knuth-Bendix completion), as none of the equations that we assumed contain the vertex \bullet inside of a $\boxed{}$ -box.

The proof above is quite similar in form to the types of inductive proofs checked by automated proof assistants. In principal, it would be straightforward to verify this proof in Quantomatic. Perhaps more advanced techniques like those employed by the proof-planning tool IsaPlanner [23] (e.g. rippling) could be adapted to string graphs to automatically *search* for such proofs as well.

Of course, $\boxed{}$ -boxes are not the only way to compactly represent infinite sets of string graphs. We could also describe sets of graphs by introducing a “meta” rewrite system, where certain types of rewrite rules are treated as *productions* in a graph grammar. In fact, this was the terminology originally used by the graph rewriting literature in the 1970s [27]. These rules would not need to respect inputs and outputs to a string graph, but some provision would need to be made when applying a meta-rule to a normal string graph rewrite rule to ensure that both the LHS and RHS are expanded in the same way. One could think of pattern graphs, as defined in this section, as something akin to regular languages, whereas sets of graphs described by graph grammars are richer (e.g. context-free or context-sensitive) languages. It is the hope that increasingly sophisticated graphical languages will lead to increasingly elegant and powerful graphical theories with applications in physics, linguistics, logic, and beyond.

Bibliography

- [1] S. Abramsky. No-Cloning In Categorical Quantum Mechanics. arXiv:0910.2401v1 [quant-ph], 2009.
- [2] S. Abramsky and B. Coecke. Physical traces: Quantum vs. classical information processing. *Electronic Notes in Theoretical Computer Science*, 69:1 – 22, 2003. CTCS'02, Category Theory and Computer Science.
- [3] S. Abramsky and B. Coecke. A categorical semantics of quantum protocols. In *Proceedings from LiCS*, arXiv:quant-ph/0402130v5, 2004.
- [4] J. Adamek, H. Herrlich, and G. E. Strecker. *Abstract and Concrete Categories. The Joy of Cats*. Dover Publications, 2009.
- [5] M. Atiyah. Topological quantum field theories. *Publications Mathématiques de L'IHÉS*, 68:175–186, 1988. 10.1007/BF02698547.
- [6] F. Baader and T. Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.
- [7] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. Elementary gates for quantum computation. arXiv:quant-ph/9503016v1, 1995.
- [8] T. Bastin, S. Krins, P. Mathonet, M. Godefroid, L. Lamata, and E. Solano. Operational families of entanglement classes for symmetric n -qubit states. *Phys. Rev. Lett.*, 103:070503, 2009.
- [9] C. H. Bennett, S. Popescu, D. Rohrlich, J. A. Smolin, and A. V. Thapliyal. Exact and Asymptotic Measures of Multipartite Pure State Entanglement. arXiv:quant-ph/9908073v3, 1999.
- [10] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer Verlag, 2004.
- [11] R. Brauer and C. Nesbitt. On the Regular Representations of Algebras. *Proceedings of the National Academy of Sciences of the United States of America*, 23(4):236, 1937.

- [12] D. E. Browne, E. Kashefi, M. Mhalla, and S. Perdrix. Generalized flow and determinism in measurement-based quantum computation. *New Journal of Physics*, 9(250), 2007.
- [13] A. Cabello. Bell’s theorem with and without inequalities for the three-qubit Greenberger-Horne-Zeilinger and W states. *Physical Review A*, 65(3), 2002.
- [14] A. Carboni and R. Walters. Cartesian bicategories I. *Journal of Pure and Applied Algebra*, 49(1–2):11 – 32, 1987.
- [15] B. Coecke and R. Duncan. Interacting quantum observables. In *Proceedings from ICALP*, pages 298–310, 2008.
- [16] B. Coecke and R. Duncan. Interacting quantum observables: Categorical algebra and diagrammatics. arXiv:0906.4725v1 [quant-ph], 2009.
- [17] B. Coecke and B. Edwards. Spekkens’s toy theory as a category of processes. arXiv:1108.1978v1 [quant-ph], 2011.
- [18] B. Coecke, B. Edwards, and R. W. Spekkens. Phase Groups and the Origin of Non-locality for Qubits. *Electronic Notes in Theoretical Computer Science*, 270(2):15–36, 2011.
- [19] B. Coecke and A. Kissinger. The compositional structure of multipartite quantum entanglement. arXiv:1002.2540v2 [quant-ph], 2010.
- [20] B. Coecke, A. Kissinger, A. Merry, and S. Roy. The GHZ/W-calculus contains rational arithmetic. arXiv:1103.2812 [cs.LO], 2011.
- [21] B. Coecke, D. Pavlovic, and J. Vicary. A new description of orthogonal bases. arXiv:0810.0812v1 [quant-ph], 2008.
- [22] L. Dixon and R. Duncan. Extending graphical representations for compact closed categories with applications to symbolic quantum computation. *AISC/MKM/Calculus*, pages 77–92, 2008.
- [23] L. Dixon and M. Johansson. Isaplanner. <http://dream.inf.ed.ac.uk/projects/isaplanner/>, 2011.
- [24] L. Dixon and A. Kissinger. Open Graphs and Monoidal Theories. arXiv:1007.3794v1 [cs.LO], 2010.
- [25] R. Duncan and S. Perdrix. Rewriting measurement-based quantum computations with generalised flow. In *Proceedings of the 37th international colloquium conference on Automata, languages and programming: Part II, ICALP’10*, pages 285–296, Berlin, Heidelberg, 2010. Springer-Verlag.

- [26] W. Dür, G. Vidal, and J. I. Cirac. Three qubits can be entangled in two inequivalent ways. *Phys. Rev. A*, 62(6), 2000.
- [27] H. Ehrig, M. Pfender, and H. J. Schneider. Graph-grammars: An algebraic approach. In *14th Annual Symposium on Switching and Automata Theory*, pages 167–180. IEEE, 1973.
- [28] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. Parallelism and concurrency in high-level replacement systems. *Mathematical Structures in Computer Science*, 1(03):361–404, 1991.
- [29] H. Ehrig and H.-J. Kreowski. Pushout-properties: An analysis of gluing constructions for graphs. *Mathematische Nachrichten*, 91(1):135–149, 1979.
- [30] M. Herrmann. Models of multipartite entanglement. MSc thesis, Oxford University, 2010.
- [31] A. Hillebrand. Superdense coding and quantum direct communication with ghz in the zx-calculus. In *Proceedings from QPL*, To be published in EPTCS. Preprint: <http://qpl.science.ru.nl/QPL.pdf>, pages 135–152, 2011.
- [32] K. Honda. Graphical classification of entangled qutrits. In *Proceedings from QPL*, To be published in EPTCS. Preprint: <http://qpl.science.ru.nl/QPL.pdf>, pages 153–171, 2011.
- [33] M. Johansson, L. Dixon, and A. Bundy. Conjecture Synthesis for Inductive Theories. *Journal of Automated Reasoning*, 2010.
- [34] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Math. Proc. Camb. Phil. Soc.*, 119(3):447–468, 1996.
- [35] A. Joyal and R. Street. The geometry of tensor calculus I. *Advances in Mathematics*, 88:55–113, 1991.
- [36] D. Kaiser. Physics and Feynman’s Diagrams. *American Scientist*, 93:156–165, 2005.
- [37] M. Kelly and M. L. Laplaza. Coherence for compact closed categories. *Journal of Pure and Applied Algebra*, 19:193–213, 1980.
- [38] A. Kissinger. Graph rewrite systems for classical structures in dagger-symmetric monoidal categories. Master’s thesis, Oxford University, 2008.
- [39] A. Kissinger. Exploring a quantum theory with graph rewriting and computer algebra. In *Intelligent Computer Mathematics*, volume 5625 of *Lecture Notes in Computer Science*, pages 90–105. Springer, 2009.
- [40] A. Kissinger, A. Merry, L. Dixon, R. Duncan, M. Soloviev, and B. Frot. Quantomatic. <https://sites.google.com/site/quantomatic/>, 2011.

- [41] J. Kock. *Frobenius Algebras and 2D Topological Quantum Field Theories*. Cambridge Univ. Press, 2003.
- [42] S. Lack. An embedding theorem for adhesive categories. arXiv:1103.0600v1 [math.CT], 2011.
- [43] S. Lack and P. Sobocinski. Adhesive Categories. *Basic Research in Computer Science*, pages 1–28, 2003.
- [44] S. Lack and P. Sobocinski. Adhesive and quasiadhesive categories. *Theoretical Informatics and Applications*, 39(2):522–546, 2005.
- [45] Y. Lafont and P. Rannou. Diagram rewriting for orthogonal matrices: A study of critical peaks. In *RTA'08*, pages 232–245, Berlin, Heidelberg, 2008. Springer-Verlag.
- [46] L. Lamata, J. Leon, D. Salgado, and E. Solano. Inductive Classification of Multipartite Entanglement under SLOCC. *Phys. Rev. A*, 74(052336), 2006.
- [47] L. Lamata, J. Leon, D. Salgado, and E. Solano. Inductive Entanglement Classification of Four Qubits under SLOCC. *Phys. Rev. A*, 75(022318), 2007.
- [48] S. Mac Lane. *Categories for the working mathematician*. Springer Verlag, 1998.
- [49] P.-A. Mellies. Towards an algebra of negation. Unpublished. Talk given at FMCS 2008. Slides to be made available at: <http://www.pps.jussieu.fr/~mellies/talks.html>.
- [50] P.-A. Mellies. Functorial boxes in string diagrams. In Z. Ésik, editor, *Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 1–30. Springer Berlin / Heidelberg, 2006.
- [51] S. Mimram. *Sémantique des jeux asynchrones et réécriture 2-dimensionnelle*. PhD thesis, Paris 7, 2008.
- [52] O. Montano-Rivas, R. McCasland, L. Dixon, and A. Bundy. Scheme-based synthesis of inductive theories. *Advances in Artificial Intelligence*, pages 348–361, 2010.
- [53] K. Narkevicius and L. Dixon. Polychrome: A firefox extension for poly/ml. <http://kidkarolis.github.com/PolyChrome>.
- [54] J. V. Neumann. Letter to Garrett Birkhoff. In M. Rédei, editor, *John von Neumann: selected letters*, volume 27 of *History of Mathematics*. American Mathematical Society, 2005.
- [55] M. A. Nielsen and G. Vidal. Majorization and the Interconversion of Bipartite States. *Quantum Information and Computation*, 1(1):76–93, 2001.

- [56] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.
- [57] E. O. Paquette. *Categorical quantum computation*. PhD thesis, Université de Montréal, 2008.
- [58] R. Penrose. Applications of negative dimensional tensors. In *Combinatorial Mathematics and its Applications*, pages 221–244. Academic Press, 1971.
- [59] U. Prange, H. Ehrig, and L. Lambers. Construction and properties of adhesive and weak adhesive high-level replacement categories. *Applications of Categorical Structures*, 16:365–388, 2008.
- [60] A. Rensink. Nested quantification in graph transformation rules. In *Graph Transformations (ICGT), Natal, Brazil*, volume 4178 of *Lecture Notes in Computer Science*. Springer Verlag, 2006.
- [61] V. Sassone and P. Sobociński. Reactive systems over cospans. In *Proceedings of 20th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 311–320, 2005.
- [62] P. Selinger. A survey of graphical languages for monoidal categories. *New Structures for Physics*, pages 275–337, 2009.
- [63] M. Shum. Tortile tensor categories. *Journal of Pure and Applied Algebra*, 93(1):57–110, 1994.
- [64] P. Sobociński. *Deriving process congruences from reaction rules*. PhD thesis, BRICS, University of Aarhus, 2004.
- [65] Wolfram Research. *Mathematica*, 2007.
- [66] A. Zeilinger, M. A. Horne, and D. M. Greenberger. Higher-Order Quantum Entanglement. In *NASA Conference Publication: Workshop on Squeezed States and Uncertainty Relations*, pages 73–81, 1991.