Wielding the ZX-calculus, Flexsymmetry, Mixed States, and Scalable Notations

Titouan Carette

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Manier le ZX-calcul

Flexsymétrie, systèmes ouverts et limandes

Wielding the ZX-calculus

Flexsymmetry, Mixed States and Scalable Notations

THÈSE

présentée et soutenue publiquement le 23 Novembre 2021

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Titouan Carette

Composition du jury

Rapporteurs :	Paul-André Melliès	CNRS
	Pawel Sobocinski	Tallinn University of Technology
Examinateurs :	Miriam Backens	University of Birmingham
	Caroline Collange	Inria
	Aleks Kissinger	University of Oxford
Directeurs :	Emmanuel Jeandel	University of Lorraine
	Simon Perdrix	Inria

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

J'ai repoussé cela jusqu'au dernier moment mais au bout d'un moment il faut bien y aller. J'ai pourtant anticipé avec un certain enthousiasme ce que j'écrirais dans ces remerciements, et ce même bien avant de commencer ma thèse. Mais j'ai, comme beaucoup d'autres choses, beaucoup changé ces derniers temps. Alors comme convenu rien ne sera comme prévu.

J'ai vu, parmi mes camarades, certains être terrorisés par l'écriture des remerciements, soucieux de n'oublier personne. J' admets un peu honteusement avoir moi même déjà parcouru distraitement des manuscrits à la recherche de mon nom, expérience humiliante que j'épargnerai à tous ici. Car il n'y a pas de noms dans ces lignes, seulement de la gratitude.

Tout d'abord envers mes parents, mes sœurs et ma famille, qui m'ont soutenu sur la voie que je me suis choisie. Et si tout à ma tâche, je ne les ai pas vu aussi souvent que je l'aurais voulu ces dernières années, j'espère qu'ils ne m'en voudront pas trop. J'imagine que la soutenance sera pour eux un moment un peu étrange mais je suis heureux qu'ils voient ça, car il y a un peu d'eux dans cette thèse.

Ces 26 dernières années de nombreuses et nombreux professeurs m'ont beaucoup appris, parfois à leur insu, et sûrement pas toujours ce qu'ils pensaient m'apprendre, mais il n'empêche que ce travail est tressé de leurs multiples influences.

Mes camarades anarcho-physiciens ont été une réserve d'oxygène académique pour moi. Grace à eux j'ai su très tôt que la recherche c'est des gens qui cherchent. Un avantage considérable.

J'ai toujours trouvé que les chercheurs fonctionnaient comme les Jedis. J'ai eu le privilège d'apprendre de deux maîtres aux styles très différents. J'éprouve à leur égard une admiration grandissante alors que j'emprunte des chemins qu'ils ont déjà suivis. Je n'en dirai pas d'avantage car je pense qu'ils peuvent déjà lire dans ce manuscrit, et sûrement dans tous mes écrits scientifiques futurs, tout ce que je leur dois.

Ce fut un plaisir et un honneur de frayer aux cotés des mocquassins (le nom est libre de droit faites vous plaisir). De ceux que je voyais quotidiennement à ceux avec qui je n'ai eu que de brèves interactions, ils m'ont inondé d'avis, d'histoires et de conseils. J'en suivrai certains, mais pas tous, je ne suis pas fou non plus.

J'improvise aussi une petite phrase pour vous, qui avez l'habitude de ce genre de choses, pour vous aussi les spéciaux, et pour vous tous qui savez ce que tendre fièrement un poireau vers le ciel signifie, et enfin pour vous qui avez tapé-tourné tant de fois. J'en dis bien sûr bien moins que ce que j'en pense.

Enfin à tous ceux qui m'ont un jour accompagné même brièvement et qui suivent maintenant d'autres voies, je me souviens.

Peu ont lu ce manuscrit en entier, un peu plus en ont parcouru des fragments. Bien entendu tout mot correctement orthographié, toute phrase grammaticalement correcte et toute équation mathématiquement raisonnable est de leur entière responsabilité. Le reste est pour moi.

Je ne tombe pas mais je triche, je vis en équipe et ma dernière ligne est à toi.

Introduction

Introduction

What is this all about?

At the time when I write this, we have become pretty good at building and running computers, the physical incarnation of the theoretical Turing machines. The sensation of accomplishment any human being feels by acknowledging this fact (do you?) takes its source in the Church-Turing thesis:

Any reasonable model of computation can be simulated by a Turing machine

Accepting implicitly that a Turing machine is a reasonable model of computation, this states that the most powerful reasonable computational device we can think of can compute as many things as Turing machines.

This is not a formal statement, and the fuzziest point here is the meaning of *reasonable*. A well-accepted interpretation of reasonable is something that a human with pen and paper can emulate. Another one, often considered equivalent, is something that can be implemented as an actual machine in the real world. However, the latter interpretation, by replacing the assumed abilities of humans with pen and paper by the laws of physics, has very different implications. Formalized in those terms, the Church-Turing thesis sounds like an invitation to ~~vampires~~ physicists to come and play with Turing machines.

Any computational device that can be built and run in our physical world can be simulated on a Turing machine

Now if we accept that Turing machines can be built and run in our physical world (and I think that most people agree that the thing I am now typing this text on is a good approximation) then this amounts to say that the most powerful computational devices we can build and run in our universe can compute as many things as Turing machines.

And in fact, when it comes to computability (can I answer the question ?) the Church-Turing thesis is believed to hold (notice this implies that our universe could perfectly be a huge simulation running on a Turing machine, Matrix-style). However when it comes to complexity, (how fast can I answer the question ?), then it seems that digging into the laws of physics can allow us to grasp a little more speed. This leads to the refined Church-Turing thesis:

Any computational device that can be built and ran in our physical world can be ***efficiently*** *simulated on a* ***quantum*** *Turing machine*

The situation requires some shifts in our way of thinking. Computer scientists usually design algorithms on intuitions roughly corresponding to classical physics. Quantum computer scientists have to carve algorithms from quantum physics. Happily, generations of physicists gave us a solid

and accurate abstract mathematical model we can rely on. So the situation is more as follows: quantum computer scientists have to carve algorithms from the abstract mathematical formalism of quantum physics.

This explains why if the first ideas of quantum computers can be traced to conferences of Richard Feynman in 1981 we have to wait until 1992 for the first interesting toy examples of quantum algorithms: the Deutsch-Jozsa algorithm [1]. The fact that it needed two papers, [1] and [2], to arrive at the form that is now taught in less than one hour in an introductory course to quantum computing gives a good idea of how difficult it is to have those "simple" quantum computing ideas.

In 1997, Peter Shor gave us our first champion, an algorithm able to factorize integers into prime numbers [3]. This frightened the cryptographers enough for them to invent a brand new field, post-quantum cryptography, only to put our communications out of reach of the recently sharpened quantum claws. After that, the twenty last years saw slowly but surely increase the examples of potential applications of quantum computing to interesting real-world problems.

As the history of computing meets its first circle, those theoretical games soon shifted from ideas to technologies. Today, companies and governments show vivid interest in quantum technologies and their promises. Here we are, in a delicious mixture of theoretical speculations, engineering challenges, hypes, and business opportunities. The present times are in some ways the quantum analog of the very youth of classical computers when computer scientists used to make (now) very laughable predictions about the future of computing.

Having the chance to learn from their example, I will neither advocate that we are at the dawn of a groundbreaking quantum revolution, nor that the quantum winter, the thermal death of the field, is eventually coming. But I do claim that quantum computing is worth to be investigated, would it only be for the fact that no field before never asked with so much intensity the questions of the physicality of computing and of the computability of physics.

Now, where does the present thesis fit into this story? As previously said the formalism of quantum mechanics is very abstract and then designing quantum algorithms is difficult. So, attempts have been made to re-express the formalism in a more intuitive way that could allow to simplify this task. One of those approaches is the ZX-calculus, which has the particularity to be a graphical language, meaning that quantum computations are represented by diagrams.

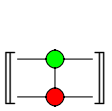
The goal of this thesis is to provide insights and extensions of the ZX-calculus with the aim to make it as operational as possible to handle quantum computations. I see my contribution as a student as an exercise of reformulation, clarification, and notation design. Thus, this thesis contains mainly two kinds of theorems: the ones showing that the notations are nice, and the ones already shown by others but here re-proved using the aforementioned nice notations.

What's in this thesis?

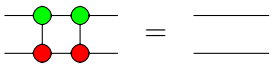
A qubit, the quantum unit of information, is represented as a unit norm vector in $\mathbb{C}^2$. A pair of qubits is represented by a vector in the tensor product $\mathbb{C}^2 \otimes \mathbb{C}^2 = \mathbb{C}^4$. The run of a quantum computer, which is the evolution of a register of a number n of qubits, then corresponds to a unitary map $\mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$. The lack of intuition about such huge unitary transformation combined with the exponentially large size of the matrices representing them led the quantum computer scientists to prefer a graphical notation to matrices: quantum circuits. Those circuits are very similar to boolean ones with the crucial difference that a gate can have more than one output and that sharing of input variables is not allowed. In other words, we work in a linear setting, information can't be copied. All the quantum algorithms and protocols designed so far can then be depicted by combinations of elementary quantum gates.

An example of such gate is the **controlled not gate**, usually called the CNot, graphically denoted as:

The CNot gate has two qubits as inputs and two qubits as outputs, so it corresponds to a complex matrix of size 4×4:

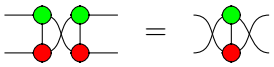

$\overset{\text{def}}{=} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

The elementary concepts of quantum computing are introduced in detail in Chapter 2 of this thesis. The CNot gate, like other quantum gates, satisfies various equations that can be interpreted as rewriting rules for quantum circuits. For example, it is an involution:

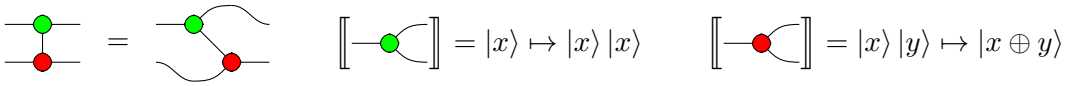


Each boolean circuit corresponds to a logical formula and then such rewriting rules on boolean circuits amount to usual term rewriting on formulas. However quantum circuits are more complicated objects, harder to represent as terms, this explains why, in circuit rewriting, the graphical representation has such an important role. The general theory of graphical languages when we can draw diagrams from generators like the CNot gate and consider diagrammatic equations like the above are introduced in Chapter 1 of this thesis. This presentation relies on the language of category theory that is presented briefly in Chapter 0.

Now coming back to the CNot gate, it satisfies a more exotic equation involving the swap gate that exchanges two qubits.



If we can compute and see why this rule is sound, it is more difficult to grasp an intuition on what is going on here. However, it happens that we can see a CNot gate as the composition of two more elementary components: copy and addition modulo two:


$= \quad \llbracket \quad \rrbracket = |x\rangle \mapsto |x\rangle\,|x\rangle \qquad \llbracket \quad \rrbracket = |x\rangle\,|y\rangle \mapsto |x \oplus y\rangle$

We see here that this decomposition involves non-unitary linear maps. This is a classic of mathematics, going through a wider place to obtain a clearer understanding of a smaller one. The gain is that the equational theory governing those subcomponents is far more intuitive and simple.

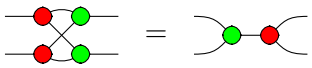
Those two components are in fact special cases of families of n-ary diagrams called spiders:

The interactions and behaviours of those spiders are the core of the ZX-calculus, the main graphical language studied in this thesis, that is introduced in Chapter 3. The ZX-calculus extends quantum circuits in the sense that any quantum gate can be decomposed into diagrams of the ZX-calculus, while some diagrams of ZX-calculus have no circuit counterpart.

Another appeal of ZX-calculus is its nice topological behaviour. Spiders appear to be extremely flexible, we can bend their legs in many ways:

My first contribution is to propose a formalisation of this property through the notion of **flexsymmetry** in Chapter 5, it relies on the notion of **paradigm** introduced in Chapter 4.

Using flexsymmetry, we see that the exotic equation involving CNots and swaps can be restated:



This equation called the bi-algebra rule and making two spiders interact is the key ingredient in the definition of Z^*-algebras in Chapter 6 where it is shown that the red and green spiders are not the only ones following this pattern. The introduction of Z^*-algebras and the classification of all their models in qubits is the second contribution of this thesis. There are essentially three such structures for qubits, corresponding to the ZX-calculus, ZH-calculus, and ZW-calculus. Similar behaviour can also appear in other settings, as in the graphical linear algebra introduced in Chapter 7.

All those calculi can then be used to represent quantum algorithms. But there are two obstacles. First, those algorithms can involve interactions with the non-quantum world that require a broader model of computation allowing classical behaviours alongside quantum ones. This is called mixed state quantum mechanics and a way to extend quantum graphical languages to this setting is presented in Chapter 8. The extension of ZX-calculus to mixed-state quantum mechanics is the third contribution presented in this thesis. A second obstacle is the fact that a quantum algorithm is defined by a uniform family of circuits, one for each possible size of the input. The scalable notations presented in Chapter 9 allow to handle such families, completing the toolbox necessary to calmly approach quantum algorithm graphically. The scalable notations are the fourth and last theoretical contribution of this thesis.

All those methods can be used to graphically prove the correction of quantum algorithms. Examples are provided in Chapter 10, which concludes the thesis.

Chapter 0, 1, 2, 3, 4 and 7 introduces the concepts and necessary notations. If the way of presenting the material can be new, those chapters don't really contain anything that cannot be find elsewhere in the literature. The other chapters (5,6,8,9 and 10) however present original contributions, some of which have led to publications:

- Chapter 5 relies on personal unpublished work. A preliminary but erroneous version is available in [4].
- Chapter 6 is based on an ICALP2020 paper [5] with Emmanuel Jeandel. I have here reformulated the results using flexsymmetry.
- Chapter 8 is based on an ICALP2019 paper [6] with Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. It has been then extended into a journal paper to appear in TQC2020.
- Chapter 9 is based on an MFCS2019 paper [7] with Dominic Horsman and Simon Perdrix. I added here some unpublished work with Simon Perdrix available in [8].
- Chapter 10 is based on a QPL2020 paper [9] with Yohann D'Anello and Simon Perdrix. I added personal work on diagonal maps that can be found in the research report [10].

How does this work relate with others?

I give here a quick survey of the main topics mentioned in the thesis. More detailed discussions of the literature are available in the corresponding Chapters.

The ZX-calculus has been introduced in [11]. It can be considered as a part of a broader research project: categorical quantum mechanics, advocating the use of arrows in dagger symmetric monoidal categories to represent quantum processes with an emphasis on string diagram notations. I would say that the main quest of the ZX-calculus community for a decade was to achieve completeness, meaning finding a set of rewriting rules able to equate any pair of diagrams representing the same quantum computation. This has been solved in [12] and [13]. Instrumental in this result has been the development of other graphical calculus derived from the ZX-calculus, [14] and [15]. The desire to unify those languages led to the work on Z^*-algebras presented in Chapter 6.

Extensions of the ZX-calculus to mixed states were already known using the doubling from [16], an example of applications are the bastard spiders of [17]. However, no complete set of rewriting rules were known.

The development of scalable notations was motivated by the need for nice representations of possibly huge quantum processes. A primitive form of scalable notations has been introduced in [18] for the analysis of quantum error-correcting codes. Similar notations for big wires have also been used informally in numerous sources using string diagrams, a good example is [17].

My emphasis on the prop formalism for graphical languages comes from [19] where it has been used to describe graphical linear algebra.

Graphical linear algebra, that inspired the matrix arrows of Chapter 9, has been developed independently of ZX-calculus in [20]. The first paper I know acknowledging the connection between the two languages is [21]. I personally learned graphical linear algebra through the blog [22].

Numerous quantum protocols have been formulated in ZX-calculus by generations of Oxford students. Sadly the corresponding master thesis and reports are often difficult to find, and I hope one day that all those applications of ZX-calculus will be gathered. I only mention here the works that directly inspired the examples of Chapter 10. Quantum algorithms based on oracles have been investigated in [23]. Graph-states have been investigated in [24]. The diagrammatic Fourier transform has first been introduced in [25] and led to the discovery of the spider nest identities developed in [26] and [27].

Contents

Chapter 0

A Pictorial Introduction to Category Theory

But I emphasize that the notions of category and functor were not formulated or put in print until the idea of a natural transformation was also at hand.

Saunders Mac Lanes [28]

The main goal of this chapter is not really to present category theory to a beginner reader but more to introduce the not-so-well-known graphical notations from [29] and [30] that I use in the thesis. The first section still is a classical presentation of the notions of category, functor, and natural transformation. The second one introduces 2-categorical string diagrams that are a generalisation of the string diagrams introduced in Chapter 1. I then introduce the categorical concepts needed to understand Chapter 2 and Chapter 4 in this graphical framework.

0.1 Basic notions

This section introduces basic definitions and notations.

0.1.1 Categories and their pictures

I start by defining small categories.

Definition 1 (Small category)**.** *A small category $\mathcal{C}$ is defined by the following data:*

- *A set $\mathcal{O}(\mathcal{C})$ of objects.*
- *A family of sets $\mathcal{C}[a,b]$ of arrows indexed by pairs of objects $(a,b) \in \mathcal{C}^2$.*
- *For each object a an identity arrow: $id_a \in \mathcal{C}[a,a]$.*
- *For each triple of objects $(a,b,c) \in \mathcal{O}(\mathcal{C})^3$, a composition map, $\_ \circ \_ : \mathcal{C}[b,c] \times \mathcal{C}[a,b] \to \mathcal{C}[a,c]$.*

The composition and identities are required to satisfy two axioms. Given $a,b,c,d \in \mathcal{O}(\mathcal{C})$, $f \in \mathcal{C}[a,b]$, $g \in \mathcal{C}[b,c]$ and $h \in \mathcal{C}[c,d]$:

$$\underset{\textit{(identity)}}{id_b \circ f = f = f \circ id_a} \qquad \underset{\textit{(associativity)}}{h \circ (g \circ f) = (h \circ g) \circ f}$$

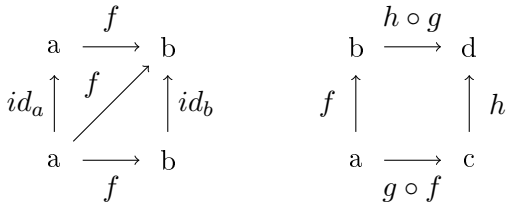
An example that will appear often is the **terminal** category $\mathbf{1}$ which has a unique object $*$ and a unique arrow id_*. An arrow $f \in \mathbf{C}[a, b]$ is said to be:

- A **monomorphism** if $f \circ g = f \circ h \quad \Rightarrow \quad g = h$.
- An **epimorphism** if $g \circ f = h \circ f \quad \Rightarrow \quad g = h$.
- An **isomorphism** if there is f^{-1} such that $f \circ f^{-1} = id_b$ and $f^{-1} \circ f = id_a$.

We say that to objects a and b are **isomorphic** if there is an isomorphism $f \in \mathbf{C}[a, b]$. There are various ways to represent graphically the arrows of a category. The most common consists in seeing categories as oriented graphs. Objects are vertices and arrows are directed edges.

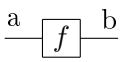
$$\text{a} \xrightarrow{f} \text{b}$$

The axioms allow to see compositions of arrows as paths between objects in the graph. The identities are the trivial paths and concatenation of path is the associative composition. We say that a diagram is **commutative** or **commutes** when any pair of paths with the same start and end represent the same arrow in the category. Commutative diagrams allow a compact presentation of equations between arrows. For example, the identity and associativity axioms can be represented by commutative diagrams.

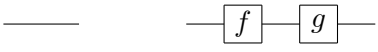


Note that here the reasoning is circular since those two axioms are exactly what makes the graphical representation consistent in the first place. In fact, I cheated a bit in those diagrams. The identity diagram could be collapsed since the identities are already represented by the empty path and the compositions in the associativity diagram could be represented as a concatenation of arrows. But then both diagrams are trivial and this is exactly the point! The defining axioms of categories are natively embedded in the graphical representation thus we won't have to use them explicitly. As it is often stated: we manage to get rid of the bureaucracy. The same phenomenon occurs with string diagrams.

String diagrams are another graphical representation where objects are represented by strings and arrows by boxes connected by strings. String diagrams are often said to be the dual of graphs diagrams, in the sense that edges become vertices and vertices become edges.

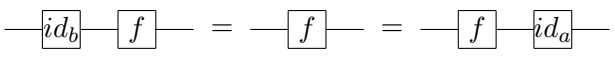


Here the identity is presented by a simple string and the composition by connecting the boxes.

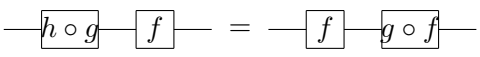


A reading orientation is needed to make sense of string diagrams. In this thesis, I choose to write every diagram from left to right.

String diagrams can be thought of as objectless representations of the category. The role of the objects is played by the identities. This imposes a kind of linearity on the object, strings only have two ends and then we cannot branch numerous arrows on one object like in graph diagrams. Thus there is no notion of commutative string diagrams that would allow to represent equalities in a compact way. In a string diagram, we need to write equality explicitly.



(identity)



(associativity)

Again, those string diagrams are clearly unnatural since the very possibility to write them is based on those axioms. The need to state explicitly the equality in string diagrams is the reason they are less used than graph diagrams which are ubiquitous in category theory. String diagrams seem not different enough from regular symbolic writing to be really useful. Graph diagrams on the contrary can be glued to form larger ones and present a long demonstration in one (often huge) commutative diagram in which we can follow the steps by identifying the paths with common ends. The interest of string diagrams really appears in the presence of additional structure when string diagrams can take care of even more bureaucracy while graph diagrams have reached their limits. Here I particularly think about monoidal categories or 2-categories. Then, the explicit equality even allows a direct connection to graph rewriting.

Like most algebraic structures there is a notion of morphism preserving it (which is on a meta-level a motivation for defining the notion of category).

Definition 2 (Functor)**.** *Given two categories $\mathcal{C}$ and $\mathcal{D}$, a **functor** is given by the following data:*

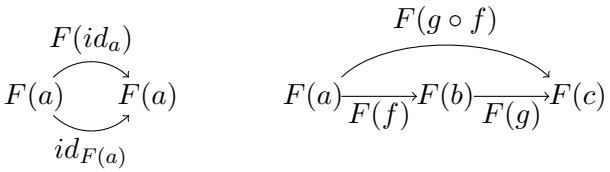
- *A map: $F : \mathcal{O}(\mathcal{C}) \to \mathcal{O}(\mathcal{D})$.*
- *A map: $F : \mathcal{C}[a, b] \to \mathcal{D}[F(a), F(b)]$.*

Those maps are required to satisfy: $F(id_a) = id_{F(a)}$ and $F(f \circ g) = F(f) \circ F(g)$. A functor is said:

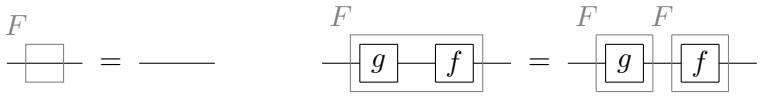
- ***faithful** if $F : \mathcal{C}[a, b] \to \mathcal{D}[F(a), F(b)]$ is injective for all $a, b \in \mathcal{O}(\boldsymbol{C})$.*
- ***full** if $F : \mathcal{C}[a, b] \to \mathcal{D}[F(a), F(b)]$ is surjective for all $a, b \in \mathcal{O}(\boldsymbol{C})$.*
- ***identity on object** (i.o.o.) if $F : \mathcal{O}(\mathcal{C}) \to \mathcal{O}(\mathcal{D})$ is the identity.*
- ***essentially surjective** (i.o.o.) if for each object $b \in \mathcal{O}(\mathcal{D})$ there is an object $a \in \mathcal{O}(\mathcal{C})$ such that b is isomorphic to $F(a)$.*

*A functor is said to be an **equivalence** of category if it is full, faithful, and essentially surjective. Two categories $\mathbf{C}$ and $\mathbf{D}$ are said **equivalent** if there is an equivalence of category between them, we write $\mathbf{C} \simeq \mathbf{D}$.*

In graph diagrams:



In string diagrams we use functor boxes satisfying:



For any small category $\mathcal{C}$ an identity functor is defined by $id_{\mathcal{C}}(a) = a$ and $id_{\mathcal{C}}(f) = f$.

The composition of two functors is again a functor. this composition is associative and the identities also behave as expected. This suggests to define a small category of small categories whose objects are small categories and arrows the functors. However we then have a foundational problem, the set of objects would then contains all sets, a contradiction, this is similar to Russel's paradox. The solutions are then the same. We can use a hierarchy of types, then we define bigger categories by replacing sets in the definition of small categories by collections or classes. Most of the time those size issues are not a problem. Typically in this thesis, I will never use something higher than the category of small categories which has a collection of objects which is not a set but just one step higher in the hierarchy. I'll call such structure **categories** and will say small categories when the collection of objects and arrows are proper sets. Instead of the language of set theory, I will now use type-theoretic notations. I write $a : \mathcal{C}$ to say that a is an object of $\mathcal{C}$ and $f : a \to b$ to say that f is an arrow which goes from a to b. The category of small category and functors is denoted **Cat**. Those size issues are discussed more in-depth in [31].

Now we can provide numerous examples of categories. The category *Set* has sets as objects and functions as arrows. In general, taking sets with an additional structure as objects and morphisms preserving this structure as arrows provides a category. Those category are often called **concrete**. This includes for example the category $\mathbf{Vect}_{\mathbb{K}}$ of vector spaces over a field $\mathbb{K}$ and $\mathbb{K}$-linear maps. Examples of non-concrete categories are the category **Rel** whose objects are sets and arrows are relations or any pre-order seen as a category whose objects are elements and arrows represent the order relation.

An example of functor is $\langle \_ \rangle : \mathbf{Set} \to \mathbf{Set}$, which maps a set A seen as an alphabet to the set of words $\langle A \rangle$ over this alphabet. A function $f : A \to B$ is mapped to a function $\langle f \rangle : \langle A \rangle \to \langle B \rangle$ defined by $\langle f \rangle(a_1, a_2, ..., a_n) \stackrel{\text{def}}{=} (f(a_1), f(a_2), ..., f(a_n))$.

0.1.2 Natural transformations

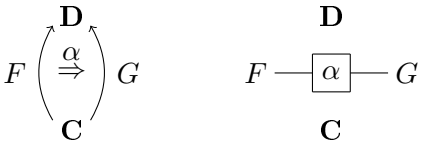
There is also a notion of arrows between functors: natural transformations.

Definition 3 (Natural transformation)**.** *Given two categories and two functors $F, G : \mathcal{C} \to \mathcal{D}$ a natural transformation α from F to G is a family of arrows $\alpha_a : F(a) \to G(a)$, called the **components** of α, indexed by the objects of $\mathcal{C}$ such that the following diagram commutes for all $a, b : \mathcal{C}$ and $f : a \to b$:*

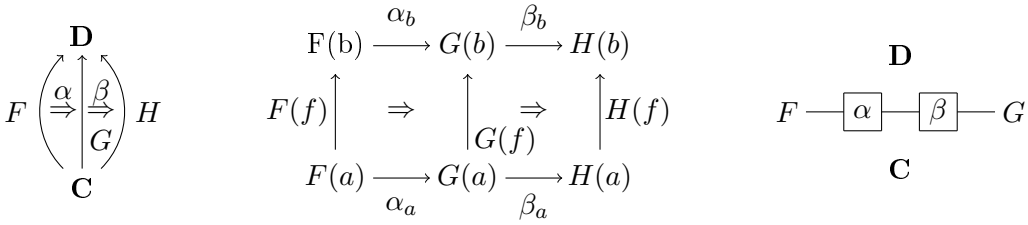
$$\begin{array}{ccc} & \overset{\alpha_b}{} & \\ F(b) & \longrightarrow & G(b) \\ {\scriptstyle F(f)}\uparrow & \Rightarrow & \uparrow{\scriptstyle G(f)} \\ F(a) & \underset{\alpha_a}{\longrightarrow} & G(a) \end{array}$$

(naturality)

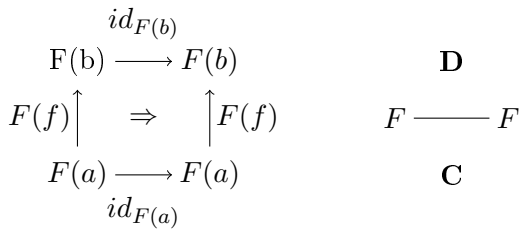
I write $\alpha : F \Rightarrow G$ to say α is a natural transformation from F to G. An example is given by the natural transformation $id_{\mathbf{Set}} \Rightarrow \langle \_ \rangle$ whose components are defined by $a \mapsto (a)$. In graph diagrams, categories are points, functors are arrows and natural transformations are arrows between arrows or regions of the plane. As for arrows, there are also dual string diagrams for natural transformations.



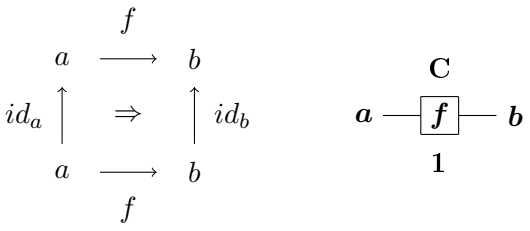
The composition $\beta\alpha : F \Rightarrow H$ of two natural transformations $\alpha : F \Rightarrow G$ and $\beta : G \Rightarrow H$ is defined to have components $(\beta \circ \alpha)_a \overset{\text{def}}{=} \beta_a \circ \alpha_a$. The naturality of the composition follows by gluing the two naturality squares:



This composition is associative. Given a functor $F : \mathbf{C} \to \mathbf{D}$, there is always an **identity natural transformation** $id_F : F \Rightarrow F$ with components $(id_F)_a \overset{\text{def}}{=} id_{F(a)}$ for each object $a : \mathbf{C}$. The naturality is given by the following square:

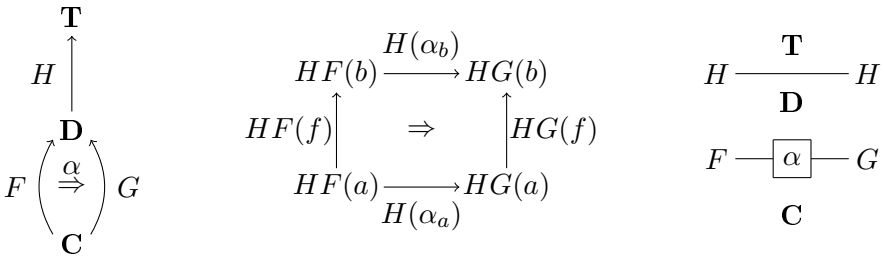


Those identities allow to see $\mathbf{Cat}[\mathbf{C}, \mathbf{D}]$ as a category with objects the functors and arrows the natural transformations. The string diagrams for natural transformations are completely similar to the one for arrows. Indeed the latter can be embedded into the former. Given an object $a : \mathbf{C}$ we can define a functor $\boldsymbol{a} : \mathbf{1} \to \mathbf{C}$, where $\mathbf{1}$ is the terminal category with a unique object denoted $*$, defined as $\boldsymbol{a}(*) = a$ and $\boldsymbol{a}(id_*) = id_a$. Then any arrow $f : a \to b$ defines a natural transformation $\boldsymbol{f} : \boldsymbol{a} \Rightarrow \boldsymbol{b}$ with components $\boldsymbol{f}_* = f$.

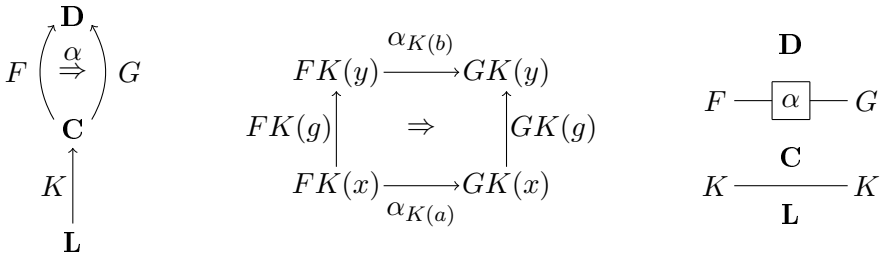


This define a faithful functor $\mathbf{C} \to \mathbf{Cat}[\mathbf{1}, \mathbf{C}]$. This trick will be use extensively in this thesis and is extremely useful to represent arrows obtained by composing numerous functors and natural transformations. If string diagrams for natural transformations generalizes the ones for

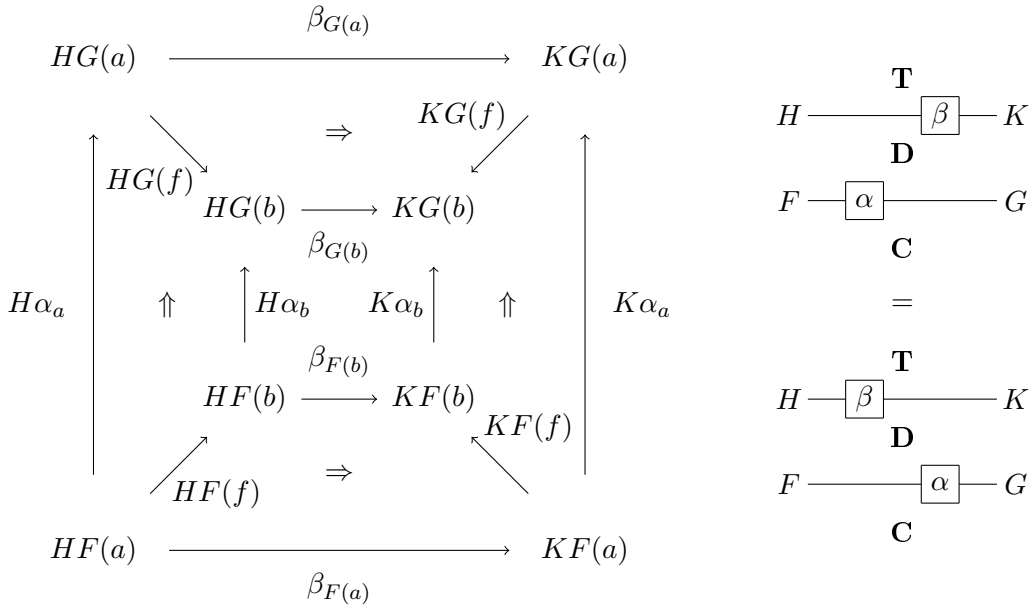
arrows, they can do a lot more. We will now define how we can make sense of juxtaposition of string diagrams. Given a natural transformation $\alpha : F \Rightarrow G$ and a functor $H : \mathbf{D} \to \mathbf{T}$ the **whiskering** $H\alpha : H \circ F \Rightarrow H \circ G$ has components $H\alpha_a \stackrel{\text{def}}{=} H(\alpha_a)$ for each objects $a : \mathbf{C}$.



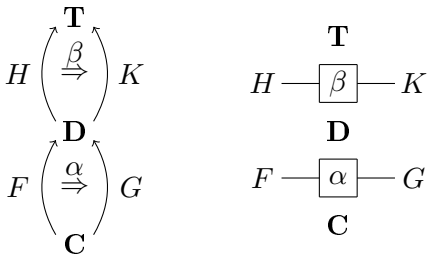
Here we just apply the functor H to the commutative diagram representing the naturality of α. Functoriality directly gives us a new commutative diagram and then, a new natural transformation. The name whiskering comes from the graph diagram where the extension functor is like a whisker to the diagram. Given a functor $K : \mathbf{L} \to \mathbf{C}$ another whiskering $\alpha K : F \circ K \Rightarrow G \circ K$ can also be defined with components $(\alpha K)_x \stackrel{\text{def}}{=} \alpha_{K(x)}$ for each objects $x : \mathbf{L}$.



The commutative diagram is basically the naturality of α applied on the image of the functor K. The two whiskering allows to define the **vertical composition** of natural transformations: $\beta \bullet \alpha : H \circ F \Rightarrow K \circ G$. There are two ways to define such a composition: as $K\alpha \circ \beta F$ or as $\beta G \circ H\alpha$. The two happen to be the same.



The naturality of α makes the Inner square commute and the naturality of β makes the outer square commutes. Thus the whole diagram is commutative and we can define $\beta \bullet \alpha \overset{\text{def}}{=} K\alpha \circ \beta F = \beta G \circ H\alpha$.



We see that the string diagrams for natural transformations allow us to represent vertical composition by the juxtaposition of diagrams. This composition is associative so no vertical parentheses are needed. Note that this forces us to represent the transformation $id_{id_{\mathbf{C}}}$ as an empty diagram, in other words, we do not write anything. Those natural transformations act as identities for the vertical composition. Yes, there is again a category with objects functors and arrows the natural transformations between functors. But we already have enough abstraction for everything that will occur in this thesis, it is wiser to stop here.

0.1.3 Strict monoidal categories

We give here the definition of strict monoidal categories that is used through the thesis. In the literature, we generally find the most subtle notion of non-strict monoidal category. However, I choose to restrict the definitions for the various categories used in the thesis to avoid the discussion of the non-strict notion. Any monoidal category is equivalent to a strict one anyway, see the famous coherence theorem in [32] for more details.

Definition 4 (strict monoidal category)**.** *A small category* $\mathbf{C}$ *is said* ***strict monoidal*** *if it is equipped with:*

- *An associative binary operation* $\_ \otimes \_ : \mathcal{O}(\mathbf{C}) \times \mathcal{O}(\mathbf{C}) \to \mathcal{O}(\mathbf{C})$ *called the* ***tensor product****.*
- *An identity object* $I \in \mathcal{O}(\mathbf{C})$ *satisfying* $I \otimes A = A \otimes I = A$ *for all object* $A \in \mathcal{O}(\mathbf{C})$*.*
- *For each objects* $A, B \in \mathcal{O}(\mathbf{C})$*, an associative binary operation* $\otimes : \mathbf{C}[A, B] \times \mathbf{C}[C, D] \to \mathbf{C}[A \otimes C, B \otimes D]$ *such that* $f \otimes id_I = f \otimes id_I = f$ *and* $\sigma_{A,I} = \sigma_{I,A} = id_A$ *.*

Furthermore, we require $A \otimes \_ : f \mapsto id_A \otimes f$ *and* $\_ \otimes B : f \mapsto f \otimes id_B$ *to be functors.*

There is a notion of strict monoidal functor preserving the tensor product:

Definition 5 (strict monoidal functor)**.** *A functor* $F : \mathbf{C} \to \mathbf{D}$ *between two strict monoidal categories is said* ***strict monoidal*** *if* $F(A \otimes B) = F(A) \otimes F(B)$*,* $F(I) = I$ *and* $F(f \otimes g) = F(f \otimes F(g))$*.*

This gives us a non-small category **MonCat** of small strict monoidal categories and strict monoidal functors.

A strict monoidal category often comes with an additional structure allowing to exchange the two components of a tensor product $A \otimes B$.

Definition 6 (symmetric strict monoidal category)**.** *A strict monoidal category* $\mathbf{C}$ *is said* ***symmetric*** *if there is a natural transformation with invertible components* $\sigma_{A,B} : A \otimes B \to B \otimes A$ *and* $\sigma_{I,I} = id_I$ *such that* $\sigma^{-1}_{A,B} = \sigma_{B,A}$.

There is also a corresponding notion of functors.

Definition 7 (symmetric strict monoidal functor)**.** *A strict monoidal functor* $F : \mathbf{C} \to \mathbf{D}$ *between two strict monoidal categories is said* ***symmetric*** *if* $F(\sigma_{A,B}) = \sigma_{F(A),F(B)}$.

This gives us a non-small category **SymMonCat** of small symmetric strict monoidal categories and symmetric strict monoidal functors.

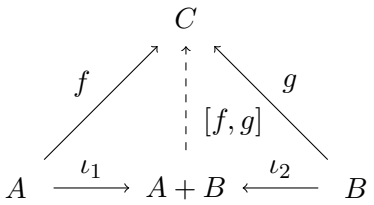
Symmetric strict monoidal categories admit a dedicated graphical notation with string diagrams that will not be developed here since it is explained in detail in Chapter 1. **Set** is not a symmetric strict monoidal category but we can obtain one if we identify isomorphic sets. Then a tensor product is given by the Cartesian product and the symmetry map is defined by $\sigma_{A,B} : (a, b) \mapsto (b, a)$.

0.2 Limits and co-limits

In this section, we set the notations for families of categorical constructions that will be used through the thesis. They are all very similar. Indeed, they are a particular case of the more general notions of limits and co-limits. More details can be found in [32] and [33].

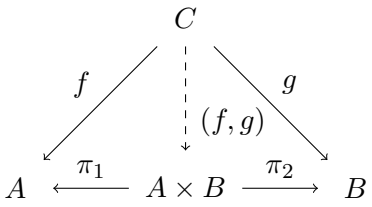
0.2.1 Products and Co-products

Given two objects A and B in a category $\mathbf{C}$ we say that A and B have a **Co-product** if there is an object $A + B$, called the co-product of A and B, and two arrows $\iota_1 : A \to A + B$ and $\iota_2 : B \to A+B$ called the **injections** satisfying the following universal property. Given any object C and a couple of arrows $f : A \to C$ and $g : B \to C$, there is a unique arrow $[f, g] : A + B \to C$ such that $[f, g] \circ \iota_1 = f$ and $[f, g] \circ \iota_2 = g$.



Such object $A + B$ needs not to be unique but it is unique up to isomorphism. So when we say $A + B$ we implicitly assume that we choose one of the possible isomorphic objects. The sum of two arrows $f : A \to B$ and $g : C \to D$ is defined as $f + g \stackrel{\text{def}}{=} [\iota_1 \circ f, \iota_2 \circ g]$. We have $f + g : A + C \to B + C$. A typical example of co-product is the disjoint union in **Set**.

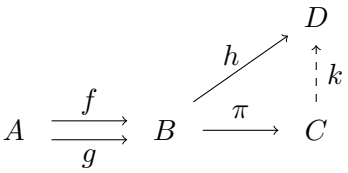
Dually a product $A \times B$ is an object with two arrows $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$ called the **projections** satisfying the following universal property. Given any object C and a couple of arrows $f : C \to A$ and $g : C \to B$, there is a unique arrow $(f, g) : C \to A \times B$ such that $\pi_1 \circ (f, g) = f$ and $\pi_2 \circ (f, g) = g$.



The Cartesian product is an example of a product in **Set**.

0.2.2 Co-equalizers

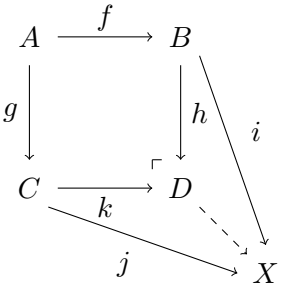
Given two maps $f, g : A \to B$ in a category $\mathbf{C}$ we say that f and g have a **co-equalizer** if there is an object C, called the co-equalizer of f and g, and an arrow $\pi : B \to C$ called the **projection** satisfying $\pi \circ f = \pi \circ g$ and the following universal property. Given any object D and any arrow $h : B \to D$ such that $h \circ f = h \circ g$, there is a unique arrow $k : C \to D$ such that $h = k \circ \pi$.



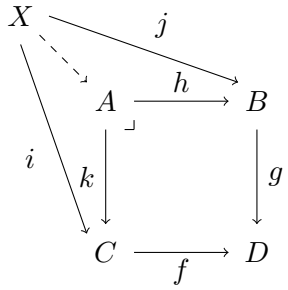
Intuitively the co-equalizer corresponds to a quotient. In **Set**, co-equalizers are obtained by identifying the images by f and g of the elements in A.

0.2.3 Pull-back and Push-outs

Given two maps $f : A \to B$ and $g : A \to C$ in a category $\mathbf{C}$ we say that f and g have a **push-out** if there is an object D, called the push-out of f and g, and two arrows $h : B \to D$ and $k : C \to D$ satisfying $h \circ f = k \circ g$ and the following universal property. Given any object X and two arrows $i : B \to X$ and $j : C \to X$ such that $i \circ f = j \circ g$, there is a unique arrow $t : D \to X$ such that $i = t \circ h$ and $j = t \circ k$.



A push-out can be seen as a combination of co-product and co-equalizers, the dual notion is called pull-back Given two maps $f : C \to D$ and $g : B \to D$ in a category $\mathbf{C}$, we say that f and g have a **pull-back** if there is an object A, called the pull-back of f and g, and two arrows $h : A \to B$ and $k : A \to C$ satisfying $g \circ h = f \circ k$ and the following universal property. Given any object X and any arrows $i : X \to B$ and $j : X \to C$ such that $g \circ j = f \circ i$, there is a unique arrow $t : X \to A$ such that $i = k \circ t$ and $j = k \circ t$.



We add a little wedge in the corner of commutative squares to indicate that they are push-out or pull-back

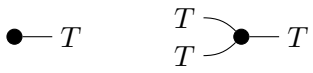
0.3 Monads and Adjunctions

Natural transformations are the key to define various concepts of category theory. I introduce in this section two of them, monads and adjunctions.

0.3.1 Monad

Monads are ubiquitous in mathematics, in particular, behind most algebraic structures there is a monad.

Definition 8. *A **Monad** is a functor $T : \mathbf{C} \to \mathbf{C}$ together with two natural transformations:*



unit *multiplication*

Furthermore those transformations are required to satisfy:

(unit laws) *(associativity)*

A possible intuition is that a monad represents a construction process that builds structures inductively. In this interpretation, TA is an object of constructions from the object A. The unit morphism $A \to TA$ can be interpreted as the inclusion of building blocks in A as trivial constructions in TA. The multiplication morphism $T^2A \to TA$ is interpreted as an inclusion stating that all possible constructions are in TA since all constructions obtained from building blocks in TA can already be seen as constructions in TA.

A typical example is the **free monoid** monad over **Set**. $\langle \_ \rangle : \mathbf{Set} \to \mathbf{Set}$ maps a set to the free monoid over this set. The unit corresponds to seeing an element of the set as a one-letter word. The multiplication is seeing a word of words as just one big word.

0.3.2 Adjunctions

Adjunctions are a kind of generalized inverse for functors. They assert a weaker notion than equivalence of category. There are different equivalent definitions of adjunction. The one I use here is not the most common but is better suited to a graphical representation.

Definition 9 (adjunction). *Two functors $F : \boldsymbol{C} \to \boldsymbol{D}$ and $U : \boldsymbol{D} \to \boldsymbol{C}$ are said to be adjoints to each others if there are two natural transformations $\eta : id_{\boldsymbol{C}} \Rightarrow U \circ F$, the **unit**, and $\epsilon : F \circ U \Rightarrow id_{\boldsymbol{D}}$, the **co-unit**, such that $\epsilon U \circ U\eta = id_U$ and $F\epsilon \circ \eta F = id_F$. We write $F \dashv U$ and say that F is the **left adjoint** of U and that U is the **right adjoint** of F.*

This definition is in fact easier to memorize with string diagrams. Let the unit and the co-unit be depicted as:

co-unit ϵ *unit η*

Then the conditions to be an adjunction are:

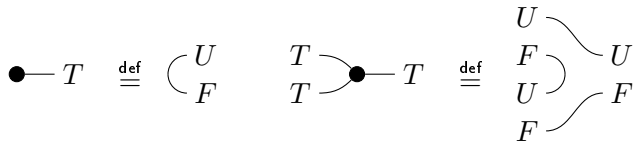
They are numerous interpretations of adjunctions depending on which specific example we have in mind. I invite the reader to look at [32] for more examples.

The unit and co-unit provide a way to map in a bijective way any arrow $f : A \to UB$ to an arrow $FA \to B$ as follows:

We see here the objects A and B as functors $\mathbf{1} \to \mathbf{C}$ and $\mathbf{1} \to \mathbf{D}$. Conversely a bijection $\varphi_{A,B} : \mathbf{D}[FA, B] \to \mathbf{C}[A, UB]$ which is natural in A an B provides a unit and a co-unit by taking as components $\varphi_{A,FA}(id_F A) : A \to UFA$ and $\varphi^{-1}_{UB,B}(id_{UB}) : FUB \to B$ respectively.

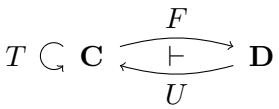
A very important property to keep in mind is that limits are preserved by right adjoints and co-limits by left adjoints. For example for the co-product we have $F(A + B) \simeq FA + FB$. Furthermore $F[f, g] = [Ff, FG]$, $F(\iota_1) = \iota_1$ and $F(\iota_2) = \iota_2$. So $F(f + g) = F(f) + F(g)$.

Given an adjunction we can always define a monad by setting:

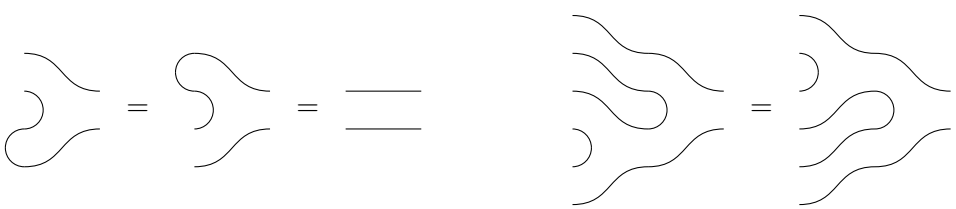


unit multiplication

We then have a factorization $T = U \circ F$.

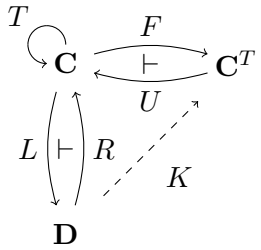


We can check graphically that the monad laws hold:

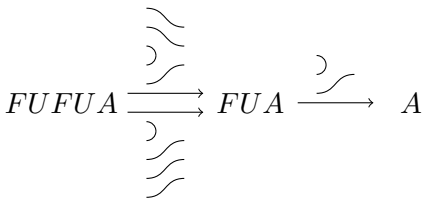


0.3.3 Monadic adjunctions

The notion we define here will be used mostly in Chapter 4. If each adjunction provides a monad the converse also holds. Given any monad $T : \mathbf{C} \to \mathbf{C}$, we can factorize it into an adjunction $F \dashv U$ with $F : \mathbf{C} \to \mathbf{C}^T$ and $U : \mathbf{C}^T \to \mathbf{C}$ where $\mathbf{C}^T$ is called the **Eilenberg-Moore** category of the monad T. This factorization is not unique but satisfies a universal property. Given any adjunction such that there is a unique functor K called the **comparison** functor such that this diagram commute:



See [32] for more details on the construction of $\mathbf{C}^T$. An adjunction $F \dashv U$ is said to be **monadic** when $\mathbf{D}$ is equivalent to $\mathbf{C}^T$, in other words, if and only if the comparison functor K is an equivalence of category. In this situation, all objects in $\mathbf{D}$ can be seen as co-equalizer of free objects, i.e., objects of the form FX.

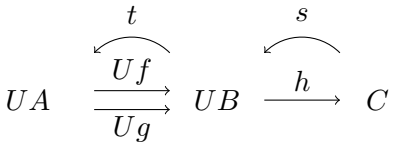


A proof of this can be found in [34]. Intuitively all objects in $\mathbf{D}$ can be presented as free construction over objects of $\mathbf{C}$ quotiented by equations. There is a very useful theorem to prove such fact:

Theorem 1 (Beck's monadicity theorem)**.** *A functor $U : \mathbf{D} \to \mathbf{C}$ is **monadic** if:*

- *U has a left adjoint.*
- *U is conservative, i.e., if Uf is an isomorphism then f is an isomorphism.*
- *$\mathbf{D}$ has and U preserves co-equalizers of U-split pairs*

A U-split pair in this theorem is a pair $f, g : A \to B$ such that there is an $h : UB \to C$, $s : C \to UB$ and $t : UB \to UA$ such that: $h \circ Ug = h \circ Uf$, $Uf \circ t = id_{UB}$, $h \circ s = id_C$ and $Ug \circ t = s \circ h$.



Proofs of this theorem can be found in [32] or [34].

Part I

Prologue

Chapter 1

Props and Graphical Languages

I think that my model has much more actually to do with mime. But there ought to be something worth seeing not just worth hearing. I want to see props and stuff.

Keith Johnstone, interviewed by Hugh Tebby, [35]

Drawing ideas is not a new trend. The history of science is filled with diagrams, schemata, and visual representations of all sorts. Most of them are considered as illustrations or analogies that allow to quickly build intuition on a problem. But some of them come with more restrictions and structures. That's the case of Circuit Diagrams [36] or control flow graphs [37]. At some point, such representation becomes formalized enough to not only convey ideas but rigorous proofs. Then graphics are not illustrations anymore but a language in itself.

Formal graphical representations are far too diverse to dare hope for a satisfying unifying theory. However, we can restrict to families of diagrams that share common ways to be drawn, composed, and interpreted. This is the case of string diagrams representing processes as boxes with inputs and outputs linked together with strings. They happen to be all unified through the notion of prop.

I spent a lot of time thinking about the right formalism to unify my favourite graphical languages. I am still not completely satisfied by the result in some fringe cases but at least the formalism is expressive enough to encompass all the material I choose to include in this thesis. This formalism relies on category theory and categorical universal algebra. The graphical formalism for natural transformations introduced in Chapter 0 will be used extensively.

1.1 Props

Props have first been introduced in [38] as an acronym. PROP stands for PROduct and Permutation category [1]. PROPs being ubiquitous in this thesis I just write prop, following the remark of Baez, Coya, and Rebro in [36] that props should be considered as ordinary mathematical citizens like groups or monoids.

[1] We might never know if this contrived acronym was chosen naively or designed on purpose with in mind to the numerous puns allowed by the polysemy of the word prop.

1.1.1 Combinatorial definition

I start with a combinatorial definition of props. The set of lists over a set C is denoted $\langle C \rangle$. A list is denoted $\overrightarrow{a}$ and its elements a_i. Concatenation is denoted additively $\overrightarrow{a} + \overrightarrow{b}$ and the empty list is denoted 0. A list with one element will just be denoted by this element.

Definition 10 (Props combinatorially)**.** *A **prop** **P** is a set of **colours** C together with sets of **arrows** indexed by pair of lists of colours denoted $\boldsymbol{P}[\overrightarrow{a}, \overrightarrow{b}]$. Furthermore we require:*

- *An associative horizontal composition operation* $\circ : \boldsymbol{P}[\overrightarrow{b}, \overrightarrow{c}] \times \boldsymbol{P}[\overrightarrow{a}, \overrightarrow{b}] \to \boldsymbol{P}[\overrightarrow{a}, \overrightarrow{c}]$.
- *An associative vertical composition operation* $\otimes : \boldsymbol{P}[\overrightarrow{a}, \overrightarrow{b}] \times \boldsymbol{P}[\overrightarrow{c}, \overrightarrow{d}] \to \boldsymbol{P}[\overrightarrow{a} + \overrightarrow{c}, \overrightarrow{b} + \overrightarrow{d}]$.
- *For each colour* $a \in C$ *identities* $id_a \in \boldsymbol{P}[a, a]$.
- *An identity for the empty list* $id_0 \in \boldsymbol{P}[0, 0]$.
- *For each pair of colours* $a, b \in C$ *a swap* $\sigma_{a,b} \in \boldsymbol{P}[a + b, b + a]$.

Defining inductively:

- $id_0 \stackrel{\text{def}}{=} id_0$.
- $id_{\overrightarrow{a}+\overrightarrow{b}} \stackrel{\text{def}}{=} id_{\overrightarrow{a}} \otimes id_{\overrightarrow{b}}$.
- $\sigma_{0,\overrightarrow{a}} \stackrel{\text{def}}{=} \sigma_{\overrightarrow{a},0} \stackrel{\text{def}}{=} id_{\overrightarrow{a}}$.
- $\sigma_{\overrightarrow{a}+\overrightarrow{b},\overrightarrow{c}+\overrightarrow{d}} \stackrel{\text{def}}{=} (id_{\overrightarrow{c}} \otimes \sigma_{\overrightarrow{a},\overrightarrow{d}} \otimes id_{\overrightarrow{b}}) \circ (\sigma_{\overrightarrow{a}+\overrightarrow{c}} \otimes \sigma_{\overrightarrow{b}+\overrightarrow{d}}) \circ (id_{\overrightarrow{a}} \otimes \sigma_{\overrightarrow{b},\overrightarrow{c}} \otimes id_{\overrightarrow{d}})$.

The following equations must hold:

- *(Interchange law)* $(f \circ h) \otimes (g \circ k) = (f \otimes g) \circ (h \otimes k)$.
- *(Horizontal identity)* $f \circ id_{\overrightarrow{a}} = id_{\overrightarrow{b}} \circ f = f$.
- *(Vertical identity)* $f \otimes id_0 = id_0 \otimes f = f$.
- *(Involution)* $\sigma_{b,a} \circ \sigma_{a,b} = id_{a+b}$.
- *(Naturality)* $(id_{\overrightarrow{c}} \otimes \sigma_{\overrightarrow{c},\overrightarrow{d}}) \circ (f \otimes id_{\overrightarrow{d}}) = (id_{\overrightarrow{d}} \otimes f) \circ \sigma_{\overrightarrow{a}+\overrightarrow{b},\overrightarrow{d}}$.

All these abstract combinatorial definitions will become clear once the string diagrams will be introduced. Note this definition corresponds to the concategories in [39].

1.1.2 Categorical definition

Equivalently props can be defined as a particular kind of category.

Definition 11 (Props categorically)**.** *A **prop** is a symmetric strict monoidal category whose monoid of objects is freely spanned by a set C of **colours**.*

I follow [40] in using the word prop for what is usually called coloured prop in the literature. Usually, prop is used to describe what I call **monochromatic props**, *i.e.*, a prop whose set of colours is a singleton. In the monochromatic case, the unique colour is denoted 1. Then each object can be uniquely designed by a natural number $n \in \mathbb{N}$.

Example 1. *The monochromatic prop* ***Fun*** *as for arrows $n \to m$ the functions $\mathbf{n} \to \mathbf{m}$ where* $\mathbf{n}$ *and* $\mathbf{m}$ *denote the finite sets $\{1, \dots, n\}$ and $\{1, , m\}$. The tensor product $\otimes$ corresponds to the disjoint union of sets and functions. The composition $\circ$ is the usual composition of functions.* 0 *is the empty set. The swap is the unique non trivial bijection $\mathbf{2} \to \mathbf{2}$.*

The main difference between props and strict monoidal category is that strict monoidal categories can have non-trivial relation between tensors of objects which is not the case of props. This specificity is in fact why props are more suited to graphical representation.

1.1.3 Categories of props

A **prop morphism** is a strict monoidal functor that maps colours to colours. Combinatorially, given two props $\mathbf{P}$ and $\mathbf{Q}$ with colours $C_{\mathbf{P}}$ and $C_{\mathbf{Q}}$, a prop morphism $F : \mathbf{P} \to \mathbf{Q}$ is defined by a function $F : C_{\mathbf{P}} \to C_{\mathbf{Q}}$ and a family of functions $F_{\overrightarrow{a},\overrightarrow{b}} : \mathbf{P}[\overrightarrow{a}, \overrightarrow{b}] \to \mathbf{Q}[\langle F \rangle(\overrightarrow{a}), \langle F \rangle(\overrightarrow{b})]$ such that:

- $F_{a,a}(id_a) = id_a$
- $F_{\overrightarrow{a},\overrightarrow{c}}(g \circ f) = F_{\overrightarrow{b},\overrightarrow{c}}(g) \circ F_{\overrightarrow{a},\overrightarrow{b}}(f)$
- $F_{\overrightarrow{a}+\overrightarrow{b},\overrightarrow{c}+\overrightarrow{d}}(f \otimes) = F_{\overrightarrow{a},\overrightarrow{c}}(f) \otimes F_{\overrightarrow{b},\overrightarrow{d}}(g)$
- $F_{\overrightarrow{a}+\overrightarrow{b},\overrightarrow{b}+\overrightarrow{a}}(\sigma_{\overrightarrow{a},\overrightarrow{b}}) = \sigma_{F(\overrightarrow{a}),F(\overrightarrow{b})}$

We clearly have a category **Props** with props as objects and prop morphisms as arrows. **Prop** is a subcategory of **SymMonCat** but not a full one. Given a set of colours C, the category C-**Prop** has C-coloured props as objects and identity on objects prop morphisms as arrows. Again C-**Prop** is then a non-full subcategory of **Prop**. Note that 1-**Prop** is the category of monochromatic props.

$$C\text{-}\mathbf{Prop} \hookrightarrow \mathbf{Prop} \hookrightarrow \mathbf{SymMonCat}$$

Prop has been studied in [40] where it is shown that it is complete and co-complete This is also the case of the category C-**Prop** of C-colored prop. I give some examples of constructions in this category that we will encounter later.

Let $\mathbf{P}$ and $\mathbf{Q}$ be C-coloured props. Then the product $\mathbf{P} \times \mathbf{Q}$ is defined by $(\mathbf{P} \times \mathbf{Q})[\overrightarrow{a}, \overrightarrow{b}] \stackrel{\text{def}}{=} \mathbf{P}[\overrightarrow{a}, \overrightarrow{b}] \times \mathbf{Q}[\overrightarrow{a}, \overrightarrow{b}]$. More generally, all limits can be computed pointwise in a similar way. Colimits are more subtle, informally the co-product of $\mathbf{P} + \mathbf{Q}$ has for arrows the constructions made from composition and tensors of swaps and arrows from $\mathbf{P}$ and $\mathbf{Q}$, quotiented by all the expected relations concerning swaps, tensors, composition and of course all the relations that were already true in $\mathbf{P}$ and $\mathbf{Q}$. In the same idea co-equalizers are obtained by identifying the necessary arrows. More concrete examples of such colimits will appear later together with explicit constructions with generators and equations.

1.2 String diagrams

String diagrams are a graphical notation to represent arrows in props. They are generalisations of Penrose notations for tensors to morphisms in any monoidal category or to be more exact, in any prop.

1.2.1 Composing Boxes

An arrow $f : \overrightarrow{a} \to \overrightarrow{b}$ in a C-coloured prop $\mathbf{P}$ is denoted by a diagram whose inputs are wires corresponding to the colors of $\overrightarrow{a}$ and outputs are wires corresponding to the colors in $\overrightarrow{b}$. Arrows and identities are depicted:

$f : \overrightarrow{a} \to \overrightarrow{b}$ $\qquad$ $id_a : a \to a$ $\qquad$ $id_0 : 0 \to 0$

Note that I will only add additional information on the wires when it is not clearly conveyed by the types of boxes. id_0 is depicted as an empty diagram and id_a by a wire of type a. The horizontal and vertical compositions are depicted:

$f \otimes g : \overrightarrow{a} + \overrightarrow{c} \to \overrightarrow{b} + \overrightarrow{d}$ $\qquad$ $f \circ g : \overrightarrow{a} \to \overrightarrow{c}$

According to its inductive definition, $id_{\overrightarrow{a}}$ is denoted by wires of type a_i in parallel:

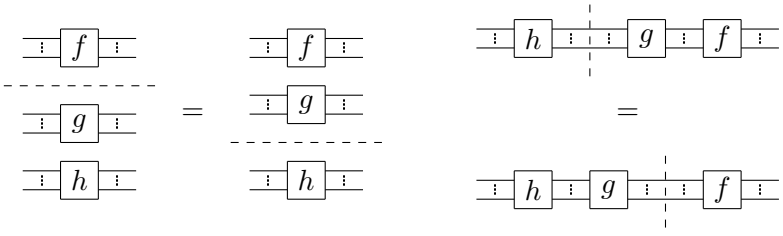
$id_{\overrightarrow{a}} : \overrightarrow{a} \to \overrightarrow{a}$

By construction string diagrams cannot handle equalities between tensors of objects. Indeed if $a \otimes b = c$ we are in a weird situation in which we don't know if we need two wires of type a and b or one wire of type c. String diagrams are not for strict monoidal categories but for props. Props are exactly the strict monoidal categories where such equalities are never a problem since every object can be uniquely decomposed into colours. So we only need one type of wire for each colour to represent any object.

1.2.2 Picturing tautologies

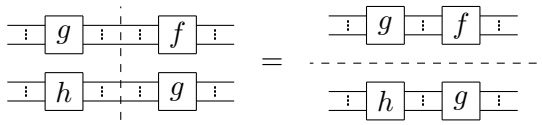
The key interest of graphical notations compared to usual formulas is that most axioms of props are directly embedded into the formalism.

The notation natively lacks the parenthesis necessary to differentiate the two hand sides of the associativity of both compositions.



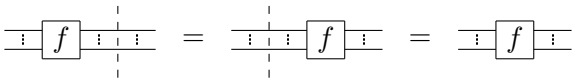
(vertical associativity) $\qquad$ *(horizontal associativity)*

But the same economy can also be reached in formulas by dropping the parenthesis. The real advantage of graphical notation concerns the interchange law.



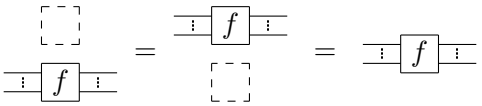
(interchange law)

Here we avoid a huge amount of bureaucratic parenthesis manipulation. A similar phenomenon occurs with identities.



(Horizontal identity)

These notations hide identities inside wires and thus diagrams natively satisfy the horizontal identity axiom.



(vertical identity)

The same happens with the vertical identity axiom, id_0 being the empty diagram we can arbitrarily consider it to be there or not.

1.2.3 Swaps

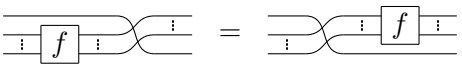
The last ingredient of the axiomatisation of props missing is the swap. The swaps are depicted by crossings of wires:

$$\sigma_{a,b} : a + b \to b + a$$

Swaps being involutive have then a direct topological interpretation:

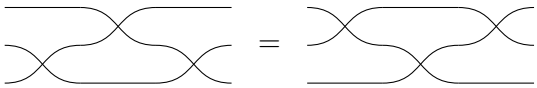
(Involution)

The naturality of the swap corresponds to the fact that any arrow can go through the wires:



(Naturality)

Note that when f itself is taken to be a swap maps we recover the Yang-Baxter equation from knot theory [41]:



Yang-Baxter

In fact, string diagrams are a model of one-dimensional strings evolving in a four-dimensional space. Indeed, the different wires can pass through each other which is not possible in three spatial dimensions where a notion of braiding would be necessary.

There are two different stances concerning the string diagrams notation. The first is to think of it as a convenient way to represent things quite informally knowing that if really needed we can easily translate back from topological manipulations to the combinatorial axioms. The second is to rely on the work of [42] or [43] where it is shown that with a good topological definition of what are string diagrams it can be rigorously shown that there are sound. In other words: we can't obtain equalities by topological manipulations of string diagrams if they don't follow from the axioms of props.

1.3 Graphical languages

In this section, we fix a given set of colours C. Identifying props with string diagrams our goal is to study the presentations of props by generators and equations. The idea is to fit the presentation of props into the general framework of categorical universal algebra [34]. We will define a category of signatures corresponding to families of generators. Then we will define a free functor F that maps a signature to the free prop generated by those generators. An equation can then be seen as a co-equalizer in the category of props which identify some diagrams to others. A graphical language will then be defined as a signature together with maps whose co-equalizer corresponds to the equations.

1.3.1 Definition

In universal algebra, the **signature** of a C-coloured prop is given by a set of **generators** $|\Sigma|$ together with an arity function $\mathfrak{a} : \Sigma \to \langle C \rangle^2$ where $\langle C \rangle$ is the free monoid spanned by the set of colours C. Given a generator $f \in |\Sigma|$, $\mathfrak{a}(f) = (a, b)$ is called the **type** of f and we often write $f : a \to b$. In a categorical setting it is more convenient to organize those data in a more type oriented structure.

Definition 12 (Signature)**.** *A C-coloured **signature** is a functor $\Sigma : \langle C \rangle^2 \to \mathbf{Set}$ where $\langle C \rangle^2$ is seen as a discrete category.*

This functor matches each possible type to a set of generators of this type. Note that this set can be empty. Defining $|\Sigma| \stackrel{\text{def}}{=} \biguplus_{(a,b)\in\langle C\rangle^2} \Sigma(a, b)$ and $\mathfrak{a} : G \mapsto (a, b)$ if $G \in \Sigma(a, b)$ we recover the usual definition. Conversely defining $\Sigma(a, b) \stackrel{\text{def}}{=} \mathfrak{a}^{-1}\{(a, b)\}$ gives a functor in $\mathbf{Set}^{\langle C\rangle^2}$. The two descriptions are completely equivalent.

Example 2. *In practice, to define a signature we use string diagrams to denote the generators and use a set inspired notation. For example $\mathbb{M}_s \stackrel{\text{def}}{=} \{$●–, ◗●–$\}$ is a monochromatic signature with two generators of respective types $0 \to 1$ and $2 \to 1$. Hence we have $\mathbb{M}_s[0, 1] = \{$●–$\}$, $\mathbb{M}_s[2, 1] = \{$◗●–$\}$ and $\mathbb{M}_s[n, m] = \emptyset$ for all other n and m.*

The functor point of view allows to define a signature map as a natural transformation. A **signature map** $\alpha : \Sigma \to \Sigma'$ is a set of functions $\alpha_{a,b} : \Sigma(a,b) \to \Sigma'(a,b)$. There is a category of C-coloured signatures which is exactly the functor category $\mathbf{Set}^{\langle C \rangle^2}$. It will be denoted C-**Sig**.

This category is a pre-sheaf category over **Set**, and then it inherits most of its properties. C-**Sig** is complete and co-complete, in fact all limits and colimits can be computed point-wise. In particular the co-product of two signatures Σ and Σ' satisfies $(\Sigma + \Sigma')(\overrightarrow{a}, \overrightarrow{b}) = \Sigma(\overrightarrow{a}, \overrightarrow{b}) \biguplus \Sigma'(\overrightarrow{a}, \overrightarrow{b})$. There is also an empty signature defined by $\emptyset(\overrightarrow{a}, \overrightarrow{b}) \stackrel{\text{def}}{=} \emptyset$ which is an initial object. C-**Sig** satisfies the external axiom of choice, so every epimorphism splits, *i.e.*, a signature map whose components are surjective has a left inverse.

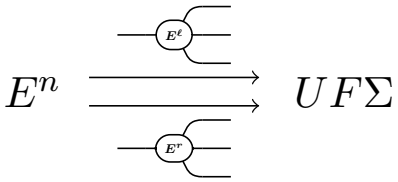
Given a C-coloured prop $\mathbf{P}$ it is always possible to gather its arrows into a signature $U(\mathbf{P})$ defined as $U(\mathbf{P})(\overrightarrow{a}, \overrightarrow{b}) \stackrel{\text{def}}{=} \mathbf{P}[\overrightarrow{a}, \overrightarrow{b}]$. Then by definition any morphisms of C-coloured props $f : \mathbf{P} \to \mathbf{Q}$ gives a family of functions $f_{\overrightarrow{a},\overrightarrow{b}} : \mathbf{P}[\overrightarrow{a}, \overrightarrow{b}] \to \mathbf{Q}[\overrightarrow{a}, \overrightarrow{b}]$. So f defines a natural transformation $U(f) : U(\mathbf{P}) \to U(\mathbf{Q})$. This defines a functor $U : C\text{-}\mathbf{Prop} \to C\text{-}\mathbf{Sig}$.

It is shown in [36] that the functor U has a left adjoint $F : C\text{-}\mathbf{Sig} \to C\text{-}\mathbf{Prop}$. Intuitively this functor maps a signature to a prop whose arrows are the string diagrams built from generators and swaps using composition and tensor product. Those diagrams are quotiented by the prop axioms.

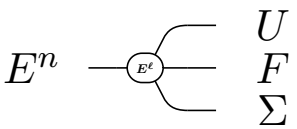
The adjunction $F \dashv U$ is monadic, meaning that the category C-**Prop** is equivalent to the Eilenberg-Moore category of $U \circ F$-algebras, see [34] for more on this. We define a unit $\eta : id_{C\text{-}\mathbf{Sig}} \Rightarrow U \circ F$ and a co-unit $\epsilon : id_{C\text{-}\mathbf{Sig}} \Rightarrow U \circ F$. The unit has components $\eta_\Sigma : \Sigma \to UF\Sigma$ which map a generator to the diagram composed of this generator only. The co-unit has components $\eta_\Sigma : FU\mathbf{P} \to \mathbf{P}$ which map a diagram composed of arrows in $\mathbf{P}$ to the arrow of $\mathbf{P}$ obtained by interpreting the formal composition and tensor product of diagram by the composition and tensor product in $\mathbf{P}$. Graphically, those two natural transformations will be denoted:

η ϵ

Definition 13 (Family of equations)**.** *A* ***family of equations*** *over a C-coloured signature Σ is a tuple $\left(E^n, E^\ell, E^r\right)$ where E^n is a C-coloured signature of* ***names****, $E^\ell : E^n \to UF(\Sigma)$ is the* ***left hand side*** *signature map and $E^r : E^n \to UF(\Sigma)$ is the* ***right hand side*** *signature map.*



Here we have denoted the signature map E^ℓ as a natural transformation between two constant functors.



The input corresponds to the functor $E^n : \mathbf{1} \to C\text{-}\mathbf{Sig}$. The three outputs of the diagram correspond to the functors $\Sigma : \mathbf{1} \to C\text{-}\mathbf{Sig}$, $F : C\text{-}\mathbf{Sig} \to C\text{-}\mathbf{Prop}$ and $U : C\text{-}\mathbf{Prop} \to C\text{-}\mathbf{Sig}$.

An equation of C-coloured prop is then obtained by choosing a type $(\overrightarrow{a}, \overrightarrow{b}) \in \langle C \rangle^2$ and a name $n \in E^n(\overrightarrow{a}, \overrightarrow{b})$. The corresponding equation is $l^{\mathcal{E}}_{(\overrightarrow{a},\overrightarrow{b})}(n) \overset{(n)}{\equiv} r^{\mathcal{E}}_{(\overrightarrow{a},\overrightarrow{b})}(n)$ where $\ell^{\mathcal{E}}_{(\overrightarrow{a},\overrightarrow{b})}(n), r^{\mathcal{E}}_{(\overrightarrow{a},\overrightarrow{b})}(n) \in UF\Sigma[\overrightarrow{a}, \overrightarrow{b}]$ are diagrams built from generators of Σ. So families of equations are a way to gather by types a set of equations.

The empty family of equation is defined by $\emptyset^n \overset{\text{def}}{=} \emptyset$, $\emptyset^\ell$ and $\emptyset^r$ are both the unique map $\emptyset \to UF\Sigma$ from the initial signature $\emptyset$.

Example 3. *In practice we define a family of equations as a set of equation between string diagrams. For example using the monochromatic signature* $\mathbb{M}_s \overset{\text{def}}{=} \{$  $\}$ *previously defined, we introduce the family of equations:*

$\mathbb{M}_e \overset{\text{def}}{=} \Big\{$ $\Big\}$

The set of names is:

$$\mathbb{M}_e^n \overset{\text{def}}{=} \{(associativity), (left\ unit), (right\ unit), (commutativity)\}$$

Considering more precisely the case of the $(commutativity)$ *equation we have:*

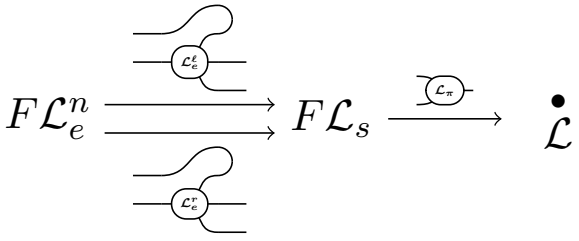
$\mathbb{M}_e^\ell : (commutativity) \mapsto$  *and* $\mathbb{M}_e^r : (commutativity) \mapsto$

Sometimes we might by a slight abuse of notation fuse some equations together and write:

$\mathbb{M}_e \overset{\text{def}}{=} \Big\{$  $\Big\}$

Definition 14 (Graphical Language)**.** *A* C*-coloured graphical language* $\mathcal{L}$ *is a pair* $(\mathcal{L}_s, \mathcal{L}_e)$ *where* $\mathcal{L}_s$ *is a* C*-coloured signature and* $\mathcal{L}_e$ *is a* C*-coloured family of equations over the signature* $\mathcal{L}_s$*.*

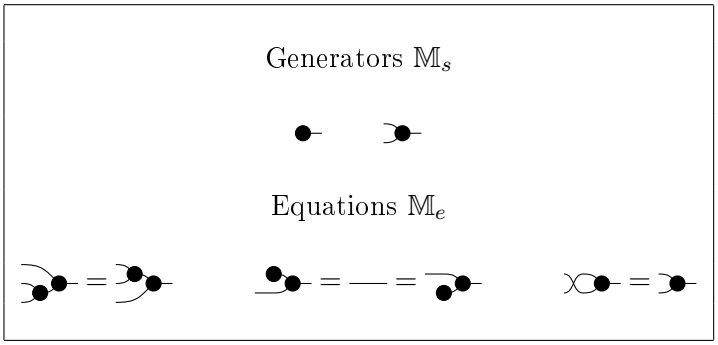
To each graphical language $\mathcal{L}$ corresponds a prop $\dot{\mathcal{L}}$ defined as the co-equalizer:



To be rigorous $\dot{\mathcal{L}}$ is only defined up to prop isomorphisms. However, we will consider that we choose one of the isomorphic props. This choice will have absolutely no consequences.

Example 4. *The monochromatic signature* $\mathbb{M}_s$ *and the family of equations* $\mathbb{M}_e$ *introduced before describe a graphical language* $\mathbb{M}$ *called the graphical languages of monoids. In practice we present graphical languages as follows:*

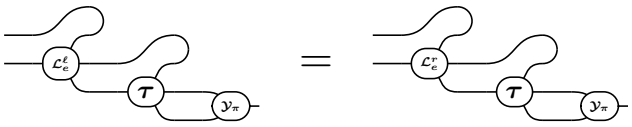
The language $\mathbb{M}$ *of monoids*



1.3.2 Translations

In this section, we define a category of graphical languages whose arrows are translations.

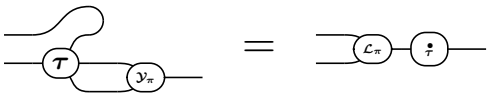
Definition 15 (Translation)**.** *A **translation** between two graphical languages $\mathcal{L}$ and $\mathcal{Y}$ is a signature map $\tau : \mathcal{L}_s \to UF\mathcal{Y}_s$ satisfying the **soundness condition**:*



Intuitively, the soundness condition ensures that equality is preserved by the translation, in other words, two equivalent diagrams in $\mathcal{L}$ are translated into two equivalent diagrams in $\mathcal{Y}$.

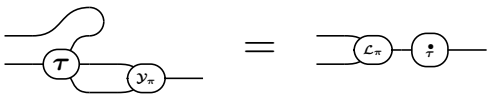
The soundness condition admits an equivalent definition that will be very useful to us.

Proposition 1 (Alternative soundness condition)**.** *A signature map $\tau : \mathcal{L}_s \to UF\mathcal{Y}_s$ satisfies the soundness condition if and only if there exists a prop morphism $\overset{\bullet}{\tau} : \overset{\bullet}{\mathcal{L}} \to \overset{\bullet}{\mathcal{Y}}$ such that:*

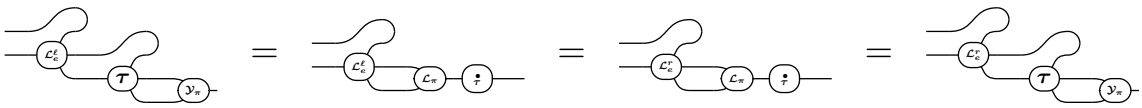


Furthermore, if this morphism exists it is unique.

Proof. Given a signature map $\tau : \mathcal{L}_s \to UF\mathcal{Y}_s$ satisfying the soundness condition, it follows directly from the universal property of the co-equalizer that there is a unique C-coloured prop morphism $\overset{\bullet}{\tau} : \overset{\bullet}{\mathcal{L}} \to \overset{\bullet}{\mathcal{Y}}$ such that:



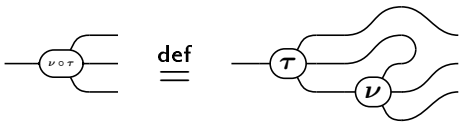
Conversely if such morphism exists then the soundness condition is satisfied:



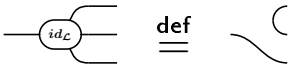
□

This alternative condition allows to show that translations form a category.

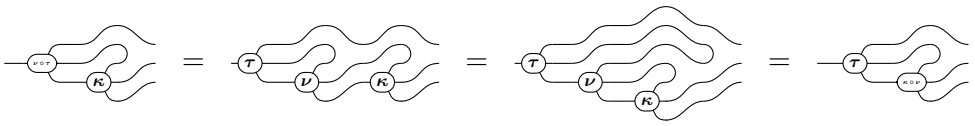
Proposition 2. *There is a category* ***GL*** *with objects the graphical languages and arrows the translations. The composition of two translations is defined as:*



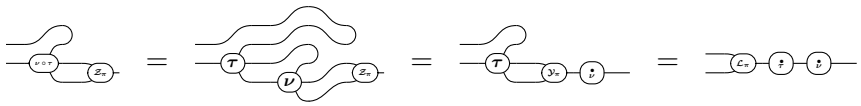
and the identities as:



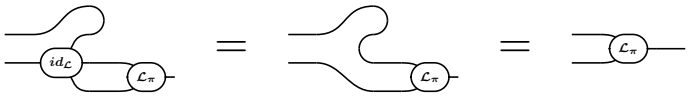
Proof. This composition is associative. Given three translations $\tau : \mathcal{L} \to \mathcal{Y}$, $\nu : \mathcal{Y} \to \mathcal{Z}$ and $\kappa : \mathcal{Y} \to \mathcal{Z}$:



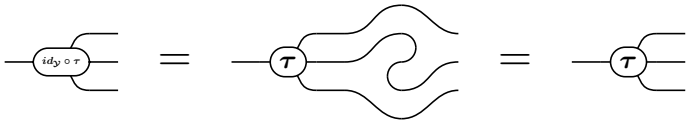
Here the key step relies on the associativity of the monad $U \circ F$. We check that $\nu \circ \tau$ satisfy the alternative soundness condition by setting: $(\nu \overset{\bullet}{\circ} \tau) \overset{\text{def}}{=} \overset{\bullet}{\nu} \circ \overset{\bullet}{\tau}$.



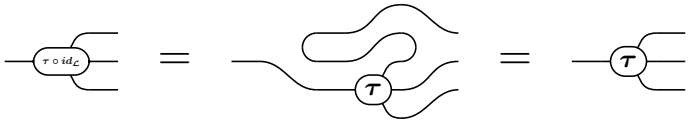
We check that $id_{\mathcal{L}}$ satisfy the alternative soundness condition by setting: $\overset{\bullet}{id}_{\mathcal{L}} \overset{\text{def}}{=} id_{\overset{\bullet}{\mathcal{L}}}$.



This construction acts as the identity for the composition of translations. Given a translation $\tau : \mathcal{L} \to \mathcal{Y}$ we have:



and



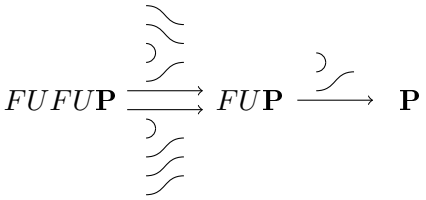
Here we see that the unit law of the monad $U \circ F$ is used. □

This last proof gives us even more: a functor $\bullet : \mathbf{GL} \to \mathbf{Prop}$.

Proposition 3. *There is a full and essentially surjective functor* $\bullet$: ***GL*** $\to$ ***Prop*** *defined by* $\mathcal{L} \mapsto \overset{\bullet}{\mathcal{L}}$ *and* $\tau \mapsto \overset{\bullet}{\tau}$.

Proof. The functoriality of $\bullet$ follows directly from the previous proof where it is shown that $\overset{\bullet}{id}_{\mathcal{L}} \overset{\text{def}}{=} id_{\overset{\bullet}{\mathcal{L}}}$ and $(\nu \overset{\bullet}{\circ} \tau) \overset{\text{def}}{=} \overset{\bullet}{\nu} \circ \overset{\bullet}{\tau}$.

To show that $\bullet$ is essentially surjective we use the fact that the adjunction $F \dashv U$ is monadic. So C-**Prop** is equivalent to the Eilenberg-Moore category of the monad $U \circ F$. Thus any prop $\mathbf{P}$ can be seen as the following co-equalizer:



A proof of this can be found in chapter 3 of [34]. From this we construct a graphical languages $\mathcal{P}$ defined by $\mathcal{P}_s \overset{\text{def}}{=} U\mathbf{P}$, $\mathcal{P}_e^n \overset{\text{def}}{=} UFU\mathbf{P}$ and:

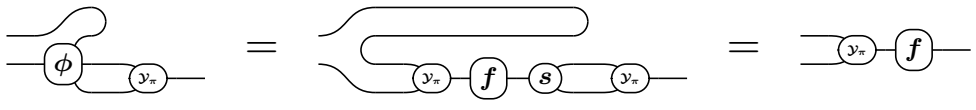
We have:

And then it follows that: $\overset{\bullet}{\mathcal{P}} \simeq \mathbf{P}$.

To show that $\bullet$ is full we consider a prop morphism $f : \overset{\bullet}{\mathcal{L}} \to \overset{\bullet}{\mathcal{Y}}$. We recall that C-**Sig** inherit from **Set** the external axiom of choice. The projection $\mathcal{Y}_\pi$ is an epimorphism by definition. Let $s : \overset{\bullet}{\mathcal{Y}} \to F\mathcal{Y}_s$ be a section of $\mathcal{Y}_\pi$.

We define the translation $\phi : \mathcal{L} \to \mathcal{Y}$ as

Checking the alternative soundness condition gives us:



So $\overset{\bullet}{\phi} = f$ and $\bullet$ is full.

□

This last functor is clearly not faithful. The same prop morphism can be described by different translations. We say that two graphical languages are **equivalent** and write $\mathcal{L} \sim \mathcal{Y}$ if $\overset{\bullet}{\mathcal{L}} = \overset{\bullet}{\mathcal{Y}}$.

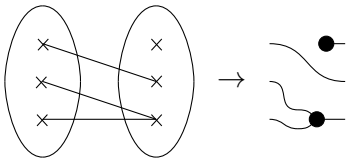
Definition 16 (Interpretation)**.** *An **interpretation** of a C-coloured graphical language $\mathcal{L}$ into a C-coloured prop $\mathbf{P}$ is a C-coloured prop morphism $[\![\_]\!] : \overset{\bullet}{\mathcal{L}} \to \mathbf{P}$. $\mathcal{L}$ is said **universal** for $\mathbf{P}$ if $[\![\_]\!]$ is full and **complete** for $\mathbf{P}$ if $[\![\_]\!]$ is faithful.*

Note that there is always a trivial universal and complete interpretation of $\mathcal{L}$ into $\overset{\bullet}{\mathcal{L}}$: the identity!

Example 5. *In practice the adjunction $F \dashv U$ allows us to describe an interpretation by matching each generator in the signature to an arrow in the model prop. There is an interpretation of $\mathbb{M}$ into* **Fun** *given by:*

$$[\![\bullet\!-]\!] \overset{\text{def}}{=} \emptyset \to \mathbf{1} \qquad\qquad [\![\supset\!\bullet\!-]\!] \overset{\text{def}}{=} \begin{cases} 1 \mapsto 1 \\ 2 \mapsto 1 \end{cases}$$

This interpretation is universal and complete, in fact the string diagram of $\mathbb{M}$ matches the potatoe diagrams sometimes used to write functions between finite sets:



1.3.3 Constructions

Now that graphical languages are well established we study the various ways to manipulate them.

Definition 17 (Free graphical language)**.** *Given a C-coloured signature Σ, the **free graphical language** Σ over the signature Σ is defined by $\Sigma_s \overset{\text{def}}{=} \Sigma$ and $\Sigma_e \overset{\text{def}}{=} \emptyset$.*

In practice denoting the free graphical language in the same way as the signature is never a problem, it will even be convenient later. There is an inclusion functor C-**Sig** $\to$ C-**GL** mapping a signature to the corresponding free graphical language. We have $\overset{\bullet}{\Sigma} = F\Sigma$ and $\Sigma_\pi = id_{F\Sigma}$.

Definition 18 (Sums)**.** *The sum of two graphical languages $\mathcal{L}$ and $\mathcal{Y}$ is defined by $(\mathcal{L}+\mathcal{Y})_s \overset{\text{def}}{=} \mathcal{L}_s + \mathcal{Y}_s$, $(\mathcal{L}+\mathcal{Y})_e \overset{\text{def}}{=} \mathcal{L}_e + \mathcal{Y}_e$, where the sum of two familly of equations is defined as $(\mathcal{L}+\mathcal{Y})_e^n \overset{\text{def}}{=} \mathcal{L}_e^n + \mathcal{Y}_e^n$ and:*

$$(\mathcal{L}+\mathcal{Y})_e^\ell \overset{\text{def}}{=} \left[\; \mathcal{L}_e^\ell \;\; \iota_1 \;\; , \;\; \mathcal{Y}_e^\ell \;\; \iota_2 \; \right]$$

$$(\mathcal{L}+\mathcal{Y})_e^r \overset{\text{def}}{=} \left[\; \mathcal{L}_e^r \;\; \iota_1 \;\; , \;\; \mathcal{Y}_e^r \;\; \iota_2 \; \right]$$

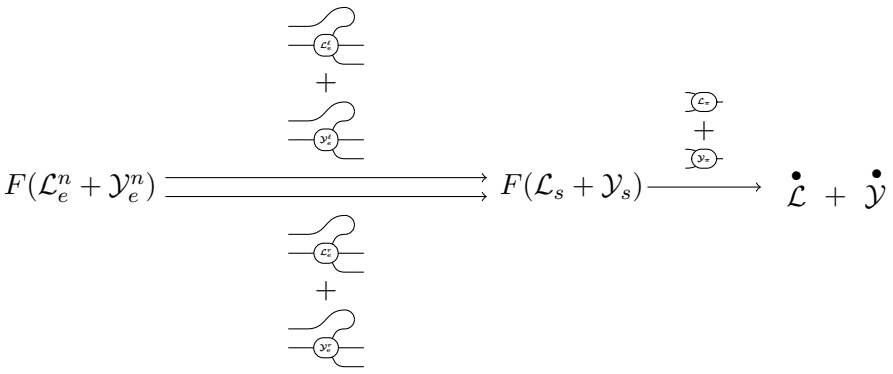
Where $\iota_1 : \mathcal{L}_s \to \mathcal{L}_s + \mathcal{Y}_s$ and $\iota_2 : \mathcal{Y}_s \to \mathcal{L}_s + \mathcal{Y}_s$ are the injections of the co-product

Here we need to be careful, the sum is not a co-product in the category of graphical languages however we do have a proper co-product in the category of props.

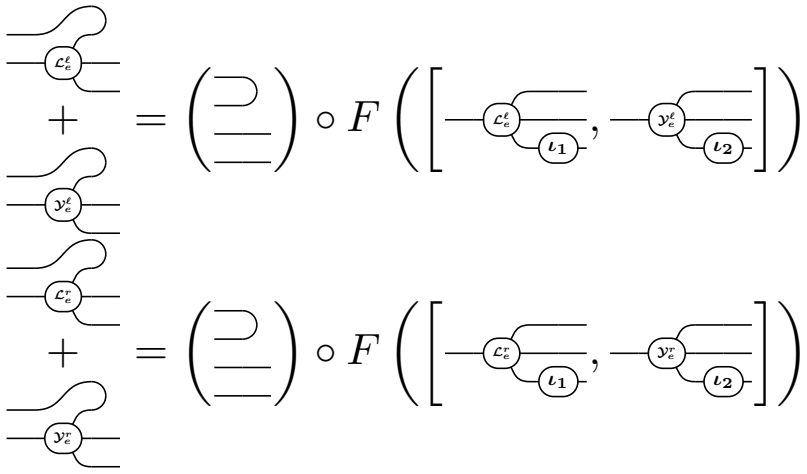
In practice, the resulting graphical language has for generator the disjoint union of both sets of generators and for equations the disjoint union of both sets of equations.

Proposition 4. $(\mathcal{L} \overset{\bullet}{+} \mathcal{Y}) \simeq \mathcal{L} \overset{\bullet}{+} \mathcal{Y}$

Proof. Since co-products commute with co-equalizers and F preserves co-products, we have the following co-equalizer:



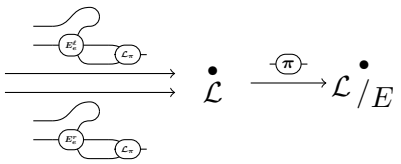
The injections of $F(\mathcal{L}_s + \mathcal{Y}_s)$ are $F\iota_1$ and $F\iota_2$ so:



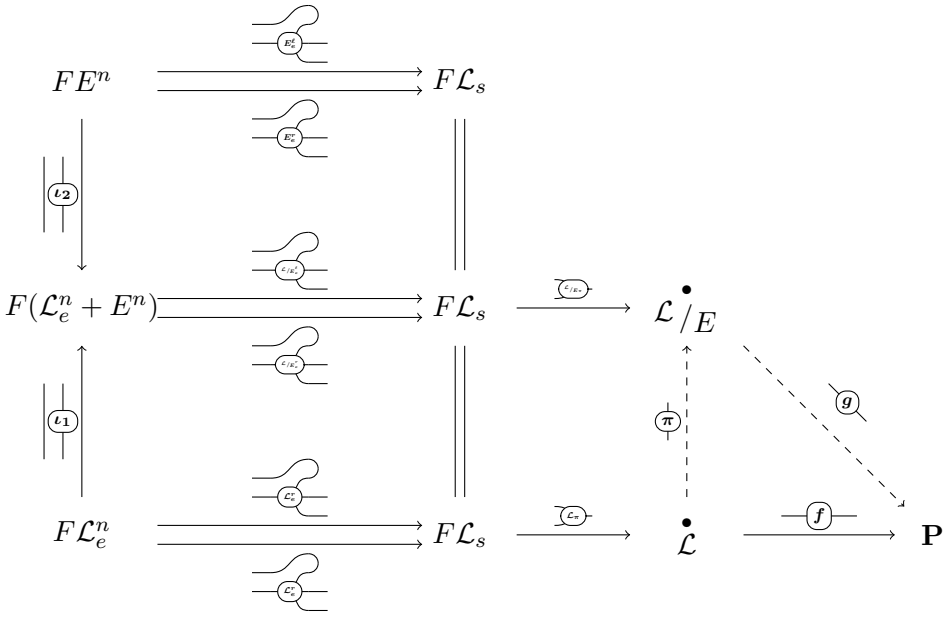
$(\mathcal{L} \overset{\bullet}{+} \mathcal{Y})$ and $\mathcal{L} \overset{\bullet}{+} \mathcal{Y}$ are then co-equalizers of the same maps and then: $(\mathcal{L} \overset{\bullet}{+} \mathcal{Y}) \simeq \mathcal{L} \overset{\bullet}{+} \mathcal{Y}$. □

Definition 19 (Quotients)**.** *The quotient of a graphical languages $\mathcal{L}$ by a family of equations E over $\mathcal{L}_s$ is defined as* $(\mathcal{L}/E)_s \overset{\text{def}}{=} \mathcal{L}_s$, $(\mathcal{L}/E)^n_e \overset{\text{def}}{=} \mathcal{L}^n_e + E^n$ *and:*

In practice, quotienting amounts to add new equations to a graphical language. This corresponds to a co-equalizer of props.

Proposition 5.  *is a co-equalizer in C-**Prop**.*

Proof. We use the following diagram:

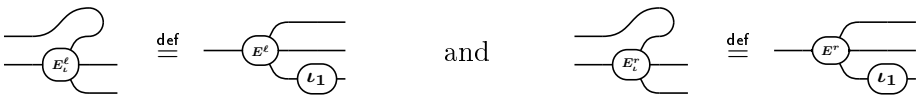


By definition of $(\mathcal{L}/E)^{\ell}_{e}$ and $(\mathcal{L}/E)^{r}_{e}$, the lower left rectangle commutes. Then the universal property of $\overset{\bullet}{\mathcal{L}}$ gives us a unique $\pi : \overset{\bullet}{\mathcal{L}} \to \mathcal{L}\overset{\bullet}{/}E$ making the right square commutes. Now the fact that the right and upper left rectangles commute ensures us that π behaves as expected.

It remains to show the universal property. Let $f : \overset{\bullet}{\mathcal{L}} \to \mathbf{P}$ be a prop morphism satisfying the co-equalizer property. Using the commutativity of the two left rectangles and the universal property of the co-product we see that f also satisfies the universal property of the co-equalizer $\mathcal{L}\overset{\bullet}{/}E$. So there is a unique prop morphism $g : \mathcal{L}\overset{\bullet}{/}E \to \mathbf{P}$ making the right triangle commute. □

With this notation it follows that for any graphical language $\mathcal{L}$ we have $\mathcal{L} = \mathcal{L}_s/\mathcal{L}_e$.

By definition, we have $\Sigma/(E+R) = (\Sigma/E)/R$. Furthermore, given a sum of signatures $\Sigma + \Gamma$, any family of equation over Σ can be canonically extended to a family of equation over the signature $\Sigma + \Gamma$ by setting:



We often simply denote E' as E when the signature is given by the context. Then we have $(\Sigma/E) + \Gamma = (\Sigma + \Gamma)/E$. This allows easy algebraic manipulation on graphical languages without the need to come back to categorical universal algebra. As an example, the axiomatisation of the sum of two graphical languages is:

$$\mathcal{L} + \mathcal{Y} = \mathcal{L}_s/\mathcal{L}_e + \mathcal{Y}_s/\mathcal{Y}_e = (\mathcal{L}_s + \mathcal{Y}_s/\mathcal{Y}_e)/\mathcal{L}_e = (\mathcal{L}_s + \mathcal{Y}_s)/\mathcal{Y}_e/\mathcal{L}_e = (\mathcal{L}_s + \mathcal{Y}_s)/(\mathcal{Y}_e + \mathcal{L}_e)$$

Example 6. *Coming back a last time to our running example of commutative monoids, we have:*

From now on I will mainly use those notations and come back to the categorical formalism only when it can't be avoided.

Chapter 2

Computer Scientist's Quantum Mechanics

Mit dem Geschirrwaschen ist es doch genau wie mit der Sprache [der Physik]: Wir haben schmutziges Spülwasser und schmutzige Küchentücher, und doch gelingt es, damit die Teller und Gläser schließlich sauberzumachen.

Niels Bohr, quoted by Werner Heisenberg [44].

Quantum mechanics has a weird status among physical theories. It has been tested many times with a remarkable amount of precision and it can't be questioned that it is an excellent description of reality. But at the same time its foundations are permanently discussed and more than a century after its introduction there is still no consensus on a preferred way to interpret the mathematical theory, or even if such an interpretation is necessary or not. I will not explicitly take any position here, even if Categorical Quantum Mechanics, the original research program to which this work is a small contribution, is clearly not neutral in such debates.

I will only introduce the part of quantum mechanics relevant to present the work in this thesis. Clearly, this chapter is more an exposition of the point of view and notations I will adopt than a real introduction to the subject. Everything will be finite-dimensional and dynamics will only be considered through discrete jumps. Our object of study is also completely abstract. I manipulate qubits without any consideration for their physical implementations. So there is no Hamiltonian, no Schroedinger equation, no spins, no particles nor fields in this chapter. Be either reassured or disappointed.

2.1 Basics

In this section, I will introduce the basic notions of quantum mechanics through the example of a computer. I will stay evasive on purpose on the exact kind of computational model I have in mind since it doesn't matter. The main point is to consider a physical system evolving in a discrete-time. I choose for pedagogical reasons to introduce the desired notions step by step through the deterministic, probabilistic, and finally quantum case. If this makes sense from the mathematical point of view, it is completely wrong when it comes to physics. Deterministic and

[1] The language of physics is like dish washing: we have dirty rinsing water and dirty kitchen towels, and yet we manage to clean the plates and glasses.

probabilistic computation are **not** special cases of the pure quantum mechanics presented here. This can only be said of mixed state quantum mechanics that will be introduced later in Chapter 7.

2.1.1 Deterministic computation

Let's consider a deterministic computer A. It has an initial state and when started it evolves through discrete steps. At a given tick t, the finite set of possible states of A is denoted X_t. This state can be completely described by a collection of bits. I denote $\mathbf{2}$ the two-element set $\{0,1\}$. A state of A at tick t can then be seen as an elements of $\mathbf{2}^{n_t}$ where $n_t \in \mathbb{N}$ is large enough such that A at tick t can be described by n_t bits, that is $\log_2(|X_t|) \leq n_t$.

Definition 20 (Deterministic state)**.** *A **deterministic state** of size n is a binary word denoted $|x\rangle$ with $x \in \mathbf{2}^n$.*

It is not possible to know the exact dynamic of A without additional information. However, in general the evolution of A between tick t and $t+1$ will correspond to a function $f : \mathbf{2}^n \to \mathbf{2}^m$ where n is large enough to describe A at tick t and $m \in \mathbb{N}$ is large enough to describe A at tick $t+1$. Moreover, given any function $f : \mathbf{2}^n \to \mathbf{2}^m$, nothing prevent us to build a machine that being in state $|x\rangle$ at tick t, is in state $|f(x)\rangle$ at tick $t+1$. Thus, all functions $f : \mathbf{2}^n \to \mathbf{2}^m$ are admissible dynamics.

Definition 21 (Deterministic computation)**.** *A **deterministic computation** is a function $f : \mathbf{2}^n \to \mathbf{2}^m$.*

It might seem odd to reduce everything to bits but doing this allows us to make everything as simple as possible and to consider a well-behaved prop. Furthermore, this is what is usually done in quantum computing. Such functions can be represented as matrices in $\mathcal{M}_{2^m \times 2^n}(\mathbf{2})$ which have exactly one 1 in each column matching each possible inputs to a unique output. We call such matrices deterministic. The composition of matrices then corresponds to the composition of functions. A state is then a column matrix with 2^n row with exactly one 1 indicating which of all possibilities the state corresponds to. In particular, we have:

$$|0\rangle \stackrel{\text{def}}{=} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } |1\rangle \stackrel{\text{def}}{=} \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Given two such computers A and B, they can be seen as one big computational system $A \times B$ by taking the Cartesian product of the states and the Cartesian product of the transition functions. Note that the number of bits necessary to describe the composed system is obtained by addition. In fact, $\mathbf{2}^n \times \mathbf{2}^m = \mathbf{2}^{n+m}$. An important point to state here is that to describe a compound system, it is enough to give the states of the components separately. This obvious fact is not true in the non-deterministic setting. Those constructions can be gathered in a prop.

Definition 22 (**Det**)**. Det** *is the monochromatic prop whose arrows $n \to m$ are the **deterministic** matrices in $\mathcal{M}_{2^m \times 2^n}(\mathbf{2})$, i.e., the matrices with exactly one 1 in each column. Composition is given by the matrix product and tensor product by the Kronecker product.*

The **Kronecker product** $M \otimes N \in \mathcal{M}_{2^{c+d} \times 2^{a+b}}(\mathbf{2})$ of two matrices $M \in \mathcal{M}_{2^c \times 2^a}(\mathbf{2})$ and $N \in \mathcal{M}_{2^d \times 2^b}(\mathbf{2})$, is defined by $(M \otimes N)_{i,k,j,\ell} = M_{i,j} N_{k,\ell}$. Deterministic states are arrows $0 \to n$ in **Det**.

2.1.2 Probabilistic computation

Now we allow probabilistic behaviours. For example, the computer can toss a coin and then store the result in the memory. Then the computer will be described by probabilistic bits which can be written as formal convex sums $p_0 |0\rangle + p_1 |1\rangle$, where p_0 and p_1 are respectively the probability to observe the deterministic state $|0\rangle$ and $|1\rangle$. By definition $p_0 + p_1 = 1$. The state of a probabilistic computer is then an element of the convex hull of $\mathbf{2}^n$ for a large enough integer n.

Definition 23 (Probabilistic state)**.** *A **probabilistic state** of size n is a convex sum $\sum_{x \in \mathbf{2}^n} p_x |x\rangle$ with $p_x \in [0,1]$ and $\sum_{x \in \mathbf{2}^n} p_x = 1$.*

Looking at the dynamic, this time, we cannot allow any functions between the convex hulls. We need to restrict to functions mapping convex sums to convex sums. Probabilistic states can be represented by column vectors whose coefficients are in $[0,1]$ and sums to 1. Probabilistic computation then corresponds to stochastic matrices.

Definition 24 (Probabilistic computation)**.** *A **Probabilistic computation** is a **stochastic matrix** in $\mathcal{M}_{2^m \times 2^n}([0,1])$, i.e., a matrix such that each column sums to 1.*

Each coefficient gives the conditional probability to obtain a deterministic state starting from another deterministic state. In the same idea, probabilistic states are column stochastic matrices describing a probability distribution over the deterministic states.

The case of composite probabilistic systems is more subtle than the deterministic one. In fact, the correct state space of $A \times B$ is not the Cartesian product of the state spaces of A and B, but the convex hull of this Cartesian product. This implies that some composite states contain correlations that cannot be expressed by a state of A and a state of B. If there are no correlations between two systems the composite dynamic is obtained by taking the Kronecker product of the corresponding stochastic matrices.

This define a prop **Sto** whose objects are integers and arrows $n \to m$ are stochastic matrices in $\mathcal{M}_{2^m \times 2^n}([0,1])$.

Definition 25 (**Sto**)**. Sto** *is the monochromatic prop whose arrows $n \to m$ are the stochastic matrices in $\mathcal{M}_{2^m \times 2^n}(\mathbf{2})$, i.e., the matrices whose column sum to 1. Composition is given by the matrix product and tensor product by the Kronecker product.*

Since deterministic matrices are stochastic, **Det** is a sub-prop of **Sto**. If the state of the computer is probabilistic, in real life, we only have access to one outcome of the distribution with a given probability. This is another clear difference with the deterministic case. We have to take into account a procedure to extract results, the information stored in the coefficients is not directly accessible.

2.1.3 Quantum computation

A first approach is to see quantum computers as more complicated probabilistic ones. A quantum state is still given by a formal sum of deterministic states, but this time, with complex coefficients. Furthermore, two quantum states are considered physically equivalent if there are equal up to multiplication by a complex number. Formally, a **qubit**, the quantum equivalent of a bit, has state space the complex projective space $\mathbb{CP}^1$, *i.e.*, the set of complex lines in the two-dimensional complex space. In practice, given such a complex line we choose a normed directing vector. Let $|0\rangle$ and $|1\rangle$ be a basis of $\mathbb{C}^2$ representing the deterministic states, the **computational basis**. A

qubit can be written $a\,|0\rangle + b\,|1\rangle$, where $a, b \in \mathbb{C}$ such that $|a|^2 + |b|^2 = 1$, the coefficient a and b are called **amplitudes**. We say that the qubit is a **superposition** of $|0\rangle$ and $|1\rangle$. We end up with vectors that look likes generalized probability distributions. As for a real line, the choice of a directing vector is not unique, it is up to a **phase**, *i.e.*, a modulus one complex number. We must always remember that a normed complex vector describes a quantum state only up to a **global phase**.

Definition 26 (Quantum state)**.** *A **Quantum state** of size n is a formal sum $\sum_{x \in \mathbf{2}^n} a_x\,|x\rangle$ with $a_x \in \mathbb{C}$ and $\sum_{x \in \mathbf{2}^n} |a_x|^2 = 1$. Furthermore two quantum states $|\phi\rangle$ and $|\psi\rangle$ are equivalent if there is $\omega \in \mathbb{C}$ such that $|\phi\rangle = \omega\,|\psi\rangle$.*

When it comes to composite systems, we are in a similar situation as for probabilistic states. Given two systems A and B described by vectors in $\mathbb{C}^{2^n}$ and $\mathbb{C}^{2^m}$, the composite system is described by vectors in the tensor product $\mathbb{C}^{2^n} \otimes \mathbb{C}^{2^m} = \mathbb{C}^{2^{n+m}}$. As in the probabilistic case, we have **separated** states, that can be written as the tensor product of smaller states, but also **entangled** states, which cannot be factorized. An example of entangled state is the **Bell** state $\frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle$.

Like in the probabilistic case we don't have access directly to a quantum state we need to **measure** it. When we measure a quantum state it collapses to the deterministic state $|x\rangle$ with probability $|a_x|$. This is called the **Born rule**. So the dynamic is given by norm preserving linear maps able to preserve those probabilities. $\mathbb{C}^{2^n}$ has a Hermitian scalar product defined in such a way that the $|x\rangle$ form an orthonormal basis. The scalar product is denoted $\langle\phi|\psi\rangle$. The **dagger** of a matrix M is defined as $M^\dagger \overset{\text{def}}{=} \overline{M}^t$, we take the complex conjugate of the coefficients of the transpose. The dagger of $|\phi\rangle$ is denoted $\langle\phi| \overset{\text{def}}{=} |\phi\rangle^\dagger$.

Definition 27 (Quantum computation)**.** *A **quantum computation** is an **isometry** in $\mathcal{M}_{2^m \times 2^n}(\mathbb{C})$, i.e., a matrix V satisfying $V^\dagger V = I$.*

This allows to define a prop of quantum computation.

Definition 28 (The prop **Qub**)**. Qub** *is the monochromatic prop whose arrows $n \to m$ are the isometries in $\mathcal{M}_{2^m \times 2^n}(\mathbb{C})$. Composition is given by the matrix product and tensor product by the Kronecker product.*

Det is not a sub-prop of **Qub** since some deterministic maps are not injective and hence not isometries. However, the reversible matrices in **Det** are in **Qub**. **Sto** is also not a sub-prop of **Qub** for similar reason. But this time even the reversible matrices in **Sto** are not in **Qub** since the square of the coefficient in the columns of a stochastic matrix usually sums to strictly less than 1. One could think about taking the square roots of the coefficients in **Sto** to match the Born rule, but this is not functorial. Definitely, stochastic maps are very different from quantum ones.

Now that we have introduced the mathematics of basic quantum mechanics it's time to say how seeing quantum systems as generalized probabilistic ones is misleading. In the probabilistic case, the difference between probabilistic and deterministic states is clear. Probabilistic states represent uncertainty but there are no doubts that only the deterministic states correspond to physical reality. This is not the case in quantum mechanics, being a superposition only makes sense with respect to an arbitrarily chosen basis. For example, some polarizations of a photon are superpositions of others. However, we cannot say that some polarizations are more fundamental or real than others. In our case, the illusion of similarity comes from the fixed

computational basis, but another choice of basis would have described the same process. This choice of basis only matters when it comes to measurements. Indeed, we presented the Born rule in the computational basis. In general, given an observable state $|\phi\rangle$, the probability to observe the quantum state $|\psi\rangle$ in state $|\phi\rangle$ is $|\langle\psi|\phi\rangle|$.

We see here more clearly why numerous phenomena, like entanglement or non-cloning, considered as counterintuitive in the quantum case are accepted by everyone in the probabilistic case. Convex sums are almost universally considered as non-physical while the ontology of quantum states is still a highly discussed topic.

2.2 The Bloch sphere

In this section, we will study more closely the mathematical structure of qubits. Qubits admit a nice visualisation: the **Bloch sphere**.

2.2.1 From unitaries to rotations

Quantum computations are modeled by unitaries. Studying one qubit computation amounts to study the group $U(2)$ of 2×2 complex matrices satisfying $U^\dagger U = UU^\dagger = I$. Note that $U(1)$ is the group of modulus one complex numbers. There is a close link between $U(2)$ and the group $SO(3)$ of rotations in $\mathbb{R}^3$. The determinant of a unitary is always of modulus 1 and any unitary can be written:

$$U = e^{\frac{i\alpha}{2}} S \text{ with } \det(U) = e^{i\alpha} \text{ and } S \in SU(2), \text{ the subgroup of } U(2) \text{ with } S = 1.$$

This decomposition is not unique, there are exactly two possibilities: $U = e^{\frac{i\alpha}{2}} S$ and $U = -e^{\frac{i\alpha}{2}}(-S)$. Any matrix $S \in SU(2)$ is of the form:

$$S = \begin{pmatrix} z & -\overline{w} \\ w & \overline{z} \end{pmatrix} \text{ with } z, w \in \mathbb{C} \text{ such that } |z|^2 + |w|^2 = 1.$$

From this first form can be deduced another one by introducing the **Pauli matrices**:

$$I \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad X \stackrel{\text{def}}{=} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y \stackrel{\text{def}}{=} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Setting $z = a + id$ and $w = b - ic$ we get:

$$S = aI + biX + ciY + diZ \text{ with } a, b, c, d \in \mathbb{R} \text{ such that } a^2 + b^2 + c^2 + d^2 = 1.$$

There is a unique $\theta \in [0, 2\pi[$ such that $a = \cos(\frac{\theta}{2})$ and $\sin(\frac{\theta}{2})^2 = b^2 + c^2 + d^2$. So any unitary $U \in U(2)$ can be written:

$$U = e^{i\alpha}\left(\cos(\tfrac{\theta}{2})I + |\sin(\tfrac{\theta}{2})|\,(n_x X + n_y Y + n_z Z)\right) \text{ with } |n_x|^2 + |n_y|^2 + |n_z|^2 = 1.$$

Indeed, If $\sin(\frac{\theta}{2}) \neq 0$ then we take $n_x = \frac{b}{|\sin(\frac{\theta}{2})|}$, $n_y = \frac{c}{|\sin(\theta)|}$ and $n_z = \frac{d}{|\sin(\theta)|}$. Else if $\sin(\frac{\theta}{2}) = 0$ any unit vector $\overrightarrow{n} = (n_x, n_y, n_z)$ works.

This decomposition allows to map any unitary to a couple $(\theta, \overrightarrow{n})$ where $\theta \in [0, 2\pi[$ and $\overrightarrow{n} \in \mathbb{R}^3$ with $||\overrightarrow{n}|| = 1$. This correspond to a rotation of angle θ around the axis directed by $\overrightarrow{n}$. In fact this map is a group morphism $R : U \mapsto R_U$. This mapping is not one to one, if we

look at the antecedent of the identity rotation we find exactly the matrices $e^{i\alpha}I$. So this map provides a group isomorphism ${}^{U(2)}\big/_{U(1)} \simeq SO(3)$.

The decomposition into Pauli matrices gives us a direct way to see unitaries as rotations. All Pauli matrices correspond to rotations of angle $\theta = \pi$ around the corresponding axis. Also, all diagonal unitaries are of the form $aI + biZ$ and then, correspond to rotations around the Z axis.

2.2.2 From quantum states to the sphere

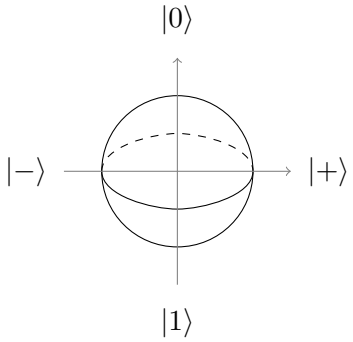
We can now describe the Bloch sphere. We represent each unit vector of $\mathbb{C}^2$ by a point on the unit sphere of $\mathbb{R}^3$. By convention, we set $|0\rangle$ to be represented by the north pole of the sphere with coordinates $(0,0,1)$. For any state $|x\rangle$ if there is a unitary U such that $|x\rangle = U|0\rangle$, we want to represent $|x\rangle$ by the point $R_U(0,0,1)$ on the sphere. We will need the following fact:

$$R_U(0,0,1) = R_{U'}(0,0,1) \qquad \Leftrightarrow \qquad U^\dagger U' \text{ is diagonal.}$$

Indeed since R is a group morphism $R_{U^\dagger U'}(0,0,1) = (0,0,1)$. So $R_{U^\dagger U'}$ is a rotation around the Z axis and hence $U^\dagger U'$ is diagonal.

Now let's show that our mapping on unit vector is well defined. Given two unitaries U and U' such that $|x\rangle = U|0\rangle$ and $|x\rangle = U'|0\rangle$ we get $U^\dagger U'|0\rangle = |0\rangle$ so $U^\dagger U'$ is diagonal and then $R_U(0,0,1) = R_{U'}(0,0,1)$. If two unit vector $|x\rangle$ and $|y\rangle$ have the same image on the sphere then there are two unitaries such that $|x\rangle = U|0\rangle$ and $|y\rangle = U'|0\rangle$. Furthermore $U^\dagger U'$ is diagonal hence $U'U^\dagger$ is diagonal. But $|y\rangle = U'U^\dagger|x\rangle$ so there is $\alpha \in \mathbb{R}$ such that $|y\rangle = e^{i\alpha}|x\rangle$. This gives us a one-to-one mapping $\mathbb{CP}^1 \to S^2$. Each quantum state corresponds to a unique point on the sphere.

A direct formula can be given. Given a unit vector $|x\rangle = z|0\rangle + w|1\rangle$, up to a global phase we can assume that $z \in \mathbb{R}$ and write $|x\rangle = \cos(\frac{\theta}{2})|0\rangle + \sin(\frac{\theta}{2})e^{i\alpha}|1\rangle$. Setting $U_x \stackrel{\text{def}}{=} \begin{pmatrix} z & -\overline{w} \\ w & \overline{z} \end{pmatrix}$ we have $|x\rangle = U_x|0\rangle$ and $U_x = \cos(\frac{\theta}{2})I + |\sin(\frac{\theta}{2})|\,(\cos(\phi)X + \sin(\phi)Y)$. The corresponding rotation R_{U_x} is of angle θ around an axis in the plan XY directed by the angle ϕ. The corresponding point has then coordinates $(\cos(\theta), \sin(\theta)\cos(\phi), \sin(\theta)\sin(\phi))$. Any point of the sphere can be reached from $|0\rangle$ by one and only one such rotation giving us a perfect correspondence between $\mathbb{CP}^1$ and S^2.

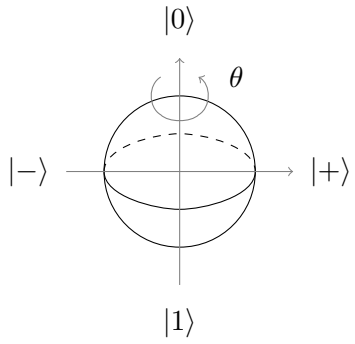


Bloch sphere

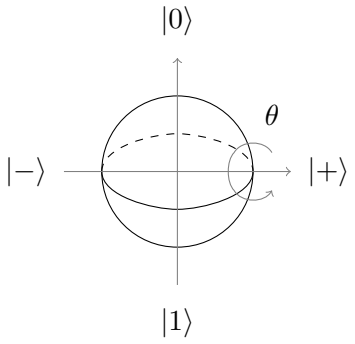
The poles are reached by $|0\rangle$ and $|1\rangle$. The intersections of the sphere with the X axis are reached by the vectors of the **diagonal basis**: $|+\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|-\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$.

2.2.3 Noticeable unitaries and the corresponding rotations

We now introduce the most common unitaries. We already noted that Pauli matrices are unitaries. X,Y and Z correspond respectively to rotations of angle π around the X, Y and Z axis. In general we define the **phase-shift** gates: $Z(\theta) \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$. They correspond to rotations of angle θ around the Z axis.



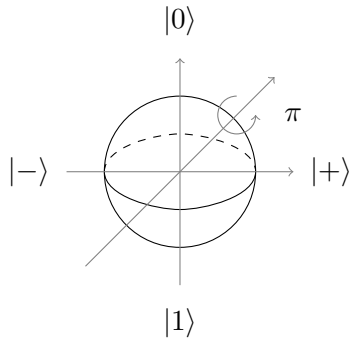
Rotation around the Z axis



Rotation around the X axis

Rotations around the X axis are reached by the gates $X(\theta) \stackrel{\text{def}}{=} \frac{1}{2}\begin{pmatrix} 1+e^{i\theta} & 1-e^{i\theta} \\ 1-e^{i\theta} & 1+e^{i\theta} \end{pmatrix}$.

Another common unitary is the Hadamard gate $H \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. The Hadamard gate is of the form $H = -i\left(\frac{1}{\sqrt{2}}iX + \frac{1}{\sqrt{2}}iZ\right)$. It then corresponds to a rotation of angle π around the diagonal axis in the XZ plane. This rotation maps the Z axis to the X axis and vice versa. This is coherent with the fact that $H|0\rangle = |+\rangle$ and $H|1\rangle = |-\rangle$. The π angle is compatible with the fact that $H^2 = I$.



Rotation corresponding to H

The Hadamard gates exchanges the Z and X axis. In fact we have $HZ(\theta)H = X(\theta)$. In practice I will not use the rotations around the Y axis, but those can be recovered defining $Y(\theta) = X(\frac{\pi}{2})Z(\theta)X(\frac{-\pi}{2})$.

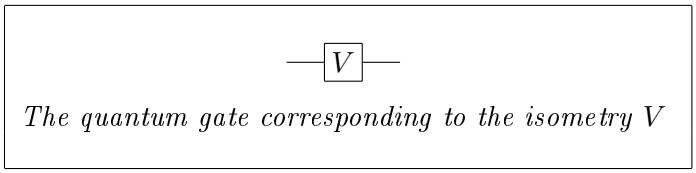
From now on we will simplify the model and do not discuss anymore the subtlety of projective spaces. I will call qubit a unit vector in $\mathbb{C}^2$ and consider the Bloch sphere as a useful way to gain intuition about $U(2)$. Of course, we will keep in mind that when it comes to physics, the global phases don't matter.

2.3 Quantum circuits

We've seen that for one qubit quantum computation, the Bloch sphere allows us to intuitively grasps what is the effect of a computation. However, when considering more than one qubits no such visualizations are available. At least, not in dimensions normal human's brains are used to. To understand bigger unitaries we decompose them into elementary ones and display this decomposition graphically in the form of quantum circuits.

2.3.1 Definition

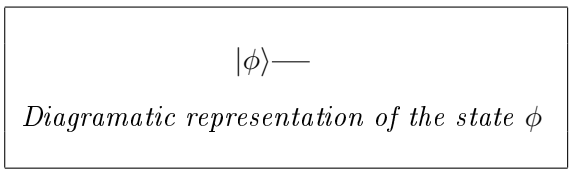
It is time to introduce our first quantum graphical language. To be honest, more than a graphical language, quantum circuits are just string diagrams representing **Qub**. Thus the graphical language $Circ$ has for generators all isometries and for equations every equation that holds in **Qub**. We have: $\overset{\bullet}{Circ} \simeq \mathbf{Qub}$. The corresponding string diagrams are called **quantum circuits** and the boxes **quantum gates**.



The quantum gate corresponding to the isometry V

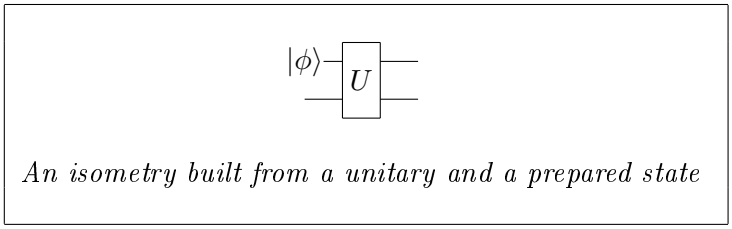
We denote $[\![\_]\!] : \overset{\bullet}{Circ} \to \mathbf{Qub}$ the prop morphism that associates to each quantum gate $n \to m$ the corresponding unitary matrix.

Since unit vectors are isometries $0 \to 1$ any qubit can also be represented in $Circ$:



Diagramatic representation of the state ϕ

The arrows $n \to m$ in **Qub** always satisfy $n \leq m$. Arrows $0 \to 0$ corresponds to phases. Usually, quantum circuits are used to represent unitaries. In fact, any isometry can be seen as a unitary in which states have been plugged in some inputs. So we can see $Circ$ as the unitary quantum circuits together with preparation of qubits states. In fact, allowing only preparations of $|0\rangle$ is enough.



An isometry built from a unitary and a prepared state

The question of the possibility of designing a compact graphical language axiomatizing **Qub**, with few generators and equations, remains open as I write this thesis. This question is important. In fact, one big advantage of the circuit representation against the matrices is the size. Describing unitaries requires an exponential number of coefficients, 2^{2n} if it acts on n-qubits. However, in practice, we use very specific kinds of unitaries, and then one can hope that not all this complexity

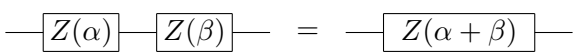
is required to describe those computations. That is, we can sometimes use polynomial circuits to represent exponentially huge matrices. The problem is then to rewrite circuits efficiently. The application is mainly the simulation of unitary evolutions and the compilation of quantum programming languages. Those considerations have led numerous people to look for an interesting set of basic gates. The choice of a set of basic gates depends on different parameters. Clearly, we want them to be expressive enough, we say that a set of gates is **universal** if any unitary can be built from the basic set of gates. Universality is, of course, a very strong property, in practice we can allow the weaker **approximate universality**, meaning that we can always build, from the basic gates, unitaries that are arbitrary close, for the usual distance, to any target unitary. I will provide examples later.

Finally, a last factor is the physical realizability of such gates. This is extremely dependant on the exact physical hardware we aim for and is not really the subject of this thesis. Thus the choice of a particular set of gates is here mostly motivated by mathematical reasons. For example, the swap gate will play a prominent role here because it is fundamental in the definition of props. However, in practice, the swap gate is difficult to implement on most physical hardware and is usually obtained by combining other gates with more obvious physical implementation.

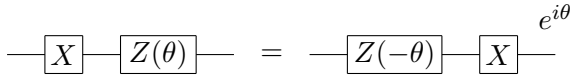
2.3.2 One qubit gates

A lot has already been said in the previous section about one-qubit gates and their relations to rotations of the Bloch sphere. Here I will focus on rewriting rules between them. Most of those rules follow directly from the Bloch sphere point of view. In circuits the phase-shift gates are denoted:

First, the phase shift gates around the same axis combine by adding their angles as expected from rotations.

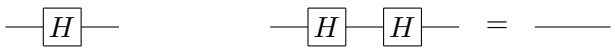


If $\theta \equiv \pi[2\pi]$ we omit the angle and just write X and Z which are exactly the Pauli gates. The X gate is also called the Not gate since $X|x\rangle = |\wedge x\rangle$. The Not gate has interesting interactions with Z phase gates given by the π commutation rule:

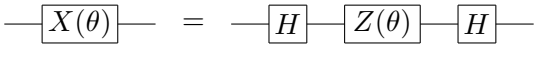


Here we used a floating scalar $e^{i\theta}$ to represent $0 \to 0$ unitaries. We see that Z does not treat $|0\rangle$ and $|1\rangle$ symmetrically. $|0\rangle$ is strictly preserved while $|1\rangle$ is only preserved up to a -1 phase that must be compensated in the commutation rule.

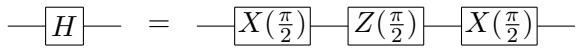
The Hadamard gate, H, is denoted:



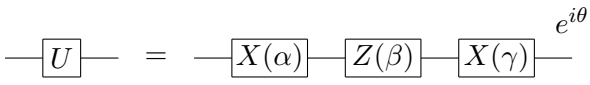
As we have seen, it as for effect to exchange the Z and X axis so it naturally translates Z phase gates into X phase gates:



The H gate can be expressed from phase shifts:

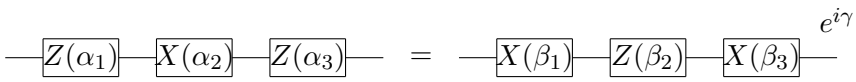


More generally, any one-qubit unitary can be obtained from phase shifts gates. This corresponds to the decomposition of rotations into Euler angles:



From what we said above it follows that $\{H, Z(\theta), e^{i\theta}I\}$, where θ can take all values in $[0, 2\pi[$, is a universal set of gates for one qubit unitaries. It is only approximately universal if we only allow θ to be of the form $\frac{k\pi}{4}$. The gate $Z(\frac{\pi}{4})$ is called the T gate. The set obtained from this set of gates is called the **Clifford+T** fragment. The not universal nor approximately universal **Clifford** fragment corresponds to values of θ of the form $\frac{k\pi}{2}$.

We see also that we can obtain a complete graphical language for one-qubit unitaries if we manage to explain how decomposition into Euler angles can be composed. This is done *via* the **Euler rule**:

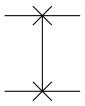


Where: $x^+ \stackrel{\text{def}}{=} \frac{\alpha_1+\alpha_3}{2}$, $x^- \stackrel{\text{def}}{=} x^+ - \alpha_3$, $z \stackrel{\text{def}}{=} \cos(\frac{\alpha_2}{2})\cos(x^+) + i\sin(\frac{\alpha_2}{2})\cos(x^-)$, $z' \stackrel{\text{def}}{=} \cos(\frac{\alpha_2}{2})\sin(x^+) - i\sin(\frac{\alpha_2}{2})\sin(x^-)$, $\beta_1 \stackrel{\text{def}}{=} \arg(z) + \arg(z')$, $\beta_2 \stackrel{\text{def}}{=} 2\arg(i + |\frac{z}{z'}|)$, $\beta_3 \stackrel{\text{def}}{=} \arg(z) - \arg(z')$ and $\gamma = x^+ - \arg(z) + \frac{\pi-\beta_2}{2}$.

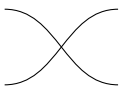
This provides a complete and universal graphical language for the one-qubit fragment of **Qub**. Obtaining such a set of rules for multi-qubit gates is more difficult.

2.3.3 Multi qubit gates

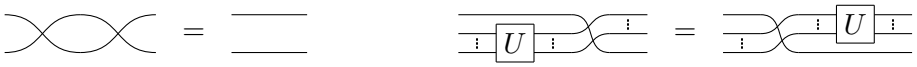
Now we move to gates acting on more than one qubits. The first obvious example is the swap gates defined as $SWAP\,|x\rangle\,|y\rangle = |y\rangle\,|x\rangle$ and usually depicted:



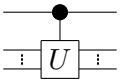
However in direct connection with the prop formalism, since the SWAP gate makes **Qub** a prop, I will prefer the crossing notation that we saw before.



It is also common in the circuit community anyway. This gate satisfies what is expected of a symmetry map:

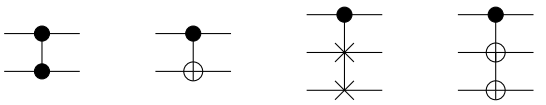


A very common family of multi qubits gates are the **controlled gates**. Given a unitary U the controlled gate CU is defined by $CU\,|0x\rangle \stackrel{\text{def}}{=} |0\rangle\,|x\rangle$ and $CU\,|1x\rangle \stackrel{\text{def}}{=} |0\rangle\,U\,|x\rangle$. They are depicted as:



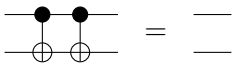
And we have $CU = \begin{pmatrix} I & \\ & U \end{pmatrix}$.

Famous gates like the CNot, CZ, Friedkin, or Toffoli gates are in this family and correspond respectively to controlling the X, Z, swap, and CNot gates.



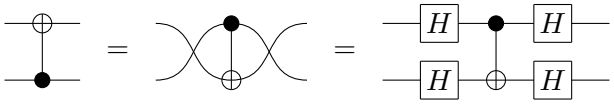
CZ CNot Fredkin Toffoloi

The most common of them remains the CNot gate acting as $CNot\,|x\rangle\,|y\rangle = |x\rangle\,|x \oplus y\rangle$. It is an involution:

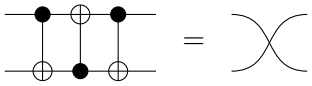


The CNot has interesting interactions with phase shifts:

Using swaps we can also define the upside-down CNot which can also be obtained using Hadamard gates:



The following identity is very important, it that will appear numerous times in this thesis in various forms:



A good surprise is that adding the CNot gate to any universal set of gates for one qubit unitaries is enough to scale universality to multi-qubit gates. However, finding a nice set of equations to obtain a complete graphical language for **Lin** is a very difficult question that is still open. Quantum circuits remain the most common way to represent quantum computation. They can be found in any introduction to quantum computing like [45] and [46]. This is also the language used in most modern quantum computing papers. However, we will see in the next chapter that there is now more competition in the world of quantum graphical languages.

Chapter 3

ZX-calculus

> "What's that? You want to know if Anansi looked like a spider? Sure he did, except when he looked like a man. No, he never changed his shape. It's just a matter of how you tell the story. That's all."
>
> Neil Gaiman, Anansi Boys

The ZX-calculus can be summarized very quickly (and of course a little bit erroneously) as an extension of quantum circuits to linear but possibly non-unitary maps. More precisely, quantum circuits are interpreted as isometry and ZX diagrams as linear maps. Thus any quantum circuit corresponds to a ZX diagram but the converse does not hold. At first, such extension appears to be a mathematical curiosity that drives us away from physical reality. I must admit this is indeed partially true, even if seeing diagrams as computation in an un-normalized post-selected model of quantum mechanics can give us back some physical intuition on this matter.

However, this extension to linear maps has also several advantages. The most obvious one follows for the recent achievement of completeness results for the ZX calculus, see [47] for a more detailed history. If we do not know any complete set of rules for circuits with respect to unitaries we do for ZX calculus with respect to linear maps. It has since then become a fashionable sport to take a quantum circuit, translate it into ZX-calculus, use the flexibility of the ZX rules to rewrite it and then, apply a lot of clever tricks to find our way back to quantum circuits. Such methods have led to very competitive inputs in the race for quantum circuit simplification [48]. Another advantage is the nice topological behaviour of ZX diagrams, they can be represented by graphs and then be manipulated by graphical proof assistants [49]. Finally being more flexible than circuits allows numerous applications to post-selected processes [50],[51].

In this chapter, I will present various aspects of the calculus and will only formally define the exact graphical language at the end.

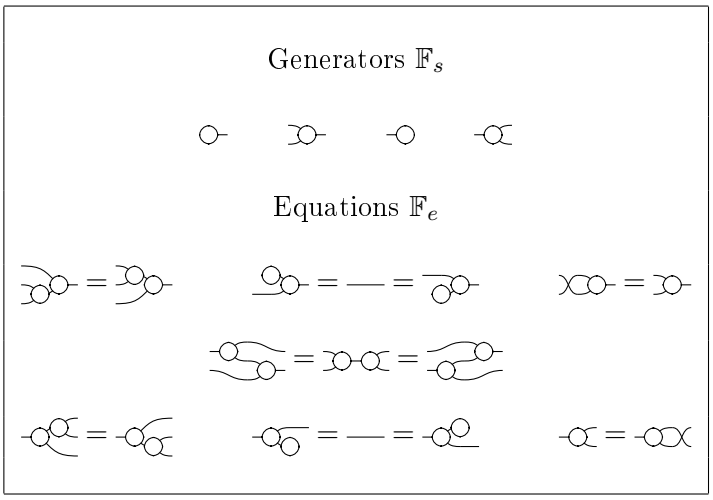
3.1 Spiders

The main building blocks of ZX calculus are the green and red spiders.

3.1.1 Frobenius algebra

What I call Frobenius algebra in this thesis are in fact commutative Frobenius algebras. There exist also non-commutative Frobenius algebras but they will not appear in this thesis. We start by defining the following monochromatic graphical language.

The language $\mathbb{F}$ of Frobenius algebras.



Note that this language can also be defined from the language of monoids introduced in Chapter 1 as:

$$\mathbb{F} = \{\mathbb{M} + \mathbb{M}^{op}\} \Big/ \left\{ \quad = \quad = \quad \right\}$$

Where $\mathbb{M}^{op}$ is the the same as $\mathbb{M}$ where all generators and equation are mirrored. We also define **special** Frobenius algebras:

$$\mathbb{SF} \overset{\text{def}}{=} \mathbb{F} \Big/ \{ \quad = \quad \}$$

Definition 29 (Frobenius algebra)**.** *We say that a monochromatic prop* $\mathbf{P}$ *has a* ***Frobenius algebra*** *when there is a monochromatic prop morphism* $\overset{\bullet}{\mathbb{F}} \to \mathbf{P}$*.*

In ZX calculus we will use two Frobenius algebras, denoted by the colours green and red. Like all generators of ZX calculus, those Frobenius algebras have interpretation in the prop of linear maps.

Definition 30 (**Lin**)**.** *The monochromatic prop* **Lin** *has for maps* $n \to m$ *the matrices in* $\mathcal{M}_{2^m \times 2^n}(\mathbb{C})$*. Composition is the matrix product and the tensor is the Kronecker product.*

There is no canonical choice for the interpretation of ZX, but different possibilities that are usually similar up to normalizing scalars. In this thesis I will use the **well tempered** normalization from [52].

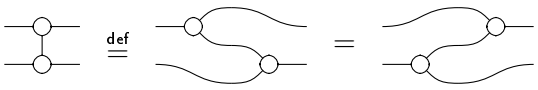
$$[\![\quad]\!] \overset{\text{def}}{=} \sqrt[4]{2}\begin{pmatrix}1\\0\end{pmatrix} \qquad [\![\quad]\!] \overset{\text{def}}{=} \frac{1}{\sqrt[4]{2}}\begin{pmatrix}1&0&0&1\\0&1&1&0\end{pmatrix} \qquad [\![\quad]\!] \overset{\text{def}}{=} \sqrt[4]{2}\begin{pmatrix}1&0\end{pmatrix} \qquad [\![\quad]\!] \overset{\text{def}}{=} \frac{1}{\sqrt[4]{2}}\begin{pmatrix}1&0\\0&1\\0&1\\1&0\end{pmatrix}$$

$$[\![\quad]\!] \overset{\text{def}}{=} \frac{1}{\sqrt[4]{2}}\begin{pmatrix}1\\1\end{pmatrix} \qquad [\![\quad]\!] \overset{\text{def}}{=} \sqrt[4]{2}\begin{pmatrix}1&0&0&0\\0&0&0&1\end{pmatrix} \qquad [\![\quad]\!] \overset{\text{def}}{=} \frac{1}{\sqrt[4]{2}}\begin{pmatrix}1&1\end{pmatrix} \qquad [\![\quad]\!] \overset{\text{def}}{=} \sqrt[4]{2}\begin{pmatrix}1&0\\0&0\\0&0\\0&1\end{pmatrix}$$

With this interpretation we have:

Those Frobenius algebras are not special. Each time we want to remove a loop we need a scalar.

With Frobenius algebras, we can use vertical wires that should not exist in string diagrams. However here the rules of ZX calculus allows to write such wires without ambiguity, in fact:

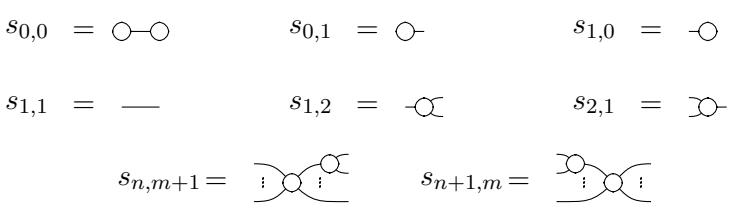


Frobenius algebras enjoy a very nice graphical calculus.

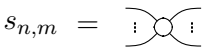
3.1.2 Spider theorem

Given a Frobenius algebra, we will define a family of maps called spiders.

Definition 31 (Spiders). *The **spiders** $s_{n,m} : n \to m$, with $n, m \in \mathbb{N}$, are define inductively as:*



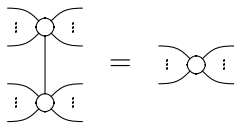
In general we write:



There are in fact numerous ways to inductively define $s_{n,m}$, however, the equations defining Frobenius algebras ensure that all define the same spider. With this definition:

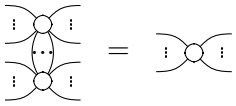
The main interest of spiders is that they admit a very nice composition rule called **spider fusion** this is given by the spider theorem.

Theorem 2 (Spider theorem). *Given a Frobenius algebra, the corresponding spiders composes as follows:*

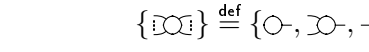


Proof. See [53] for a distributive law approach and [17] for a graphical proof. □

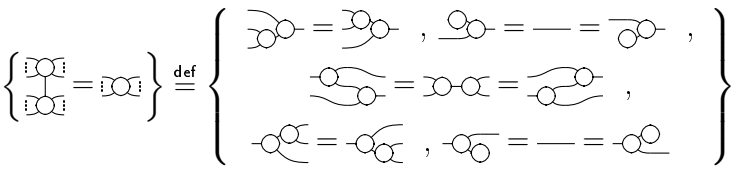
In the case of special Frobenius algebras the spider theorem can be extended:



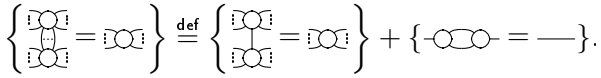
When it is required that there is at least one wire linking the two spiders. From now on, since they will be ubiquitous in this thesis, I will use shortcut notations for Frobenius algebras:



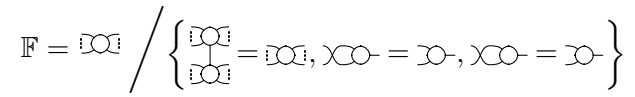
and



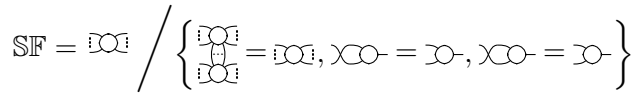
In the case of special Frobenius algebra I will use:



So the corresponding graphical languages will be written:



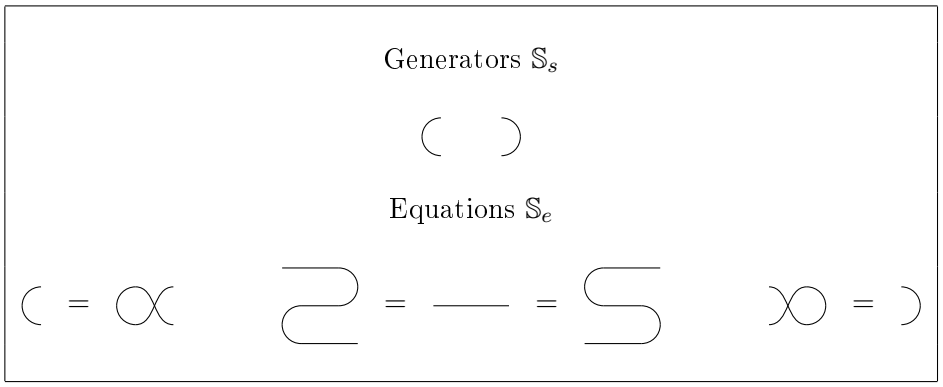
and



3.1.3 Cups and caps

We now introduce symmetric compact structures.

The language $\mathbb{S}$ of symmetric compact structures.

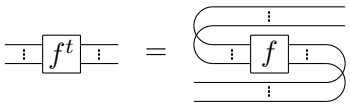


The notion of symmetric compact structure allows to define a corresponding notion of props.

Definition 32 (symmetric compact closed props)**.** *A prop* $\mathbf{P}$ *is symmetric compact closed if for each colour* c *there is a prop morphism* $F_c : \overset{\bullet}{\mathbb{S}} \to \mathbf{P}$ *such that* $F_c(1) = c$*.*

The name compact closed comes from other categorical points of view on this definition.
Symmetric compact structures allows to define arrows $0 \to n+n$ and $n+n \to 0$ by:

The **transpose** of an arrow is define as:



We have $(f^t)^t = f$. This gives a monochromatic prop morphism $\_^t : \mathbf{P}^{op} \to \mathbf{P}$. In ZX calculus there is a compact structure with interpretation in **Lin**:

$$\left[\!\left[\cup\right]\!\right] \overset{\text{def}}{=} \sum_{x\in\mathbf{2}} |xx\rangle = \begin{pmatrix}1\\0\\0\\1\end{pmatrix} \qquad \left[\!\left[\cap\right]\!\right] \overset{\text{def}}{=} \sum_{x\in\mathbf{2}} \langle xx| = \begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix}$$

With this interpretation the transpose of diagrams corresponds to the transpose of matrices in **Lin**, we have $\left[\!\left[f^t\right]\!\right] = \left[\!\left[f\right]\!\right]^t$.
Any Frobenius algebra directly provides a cup and a cap defined by:

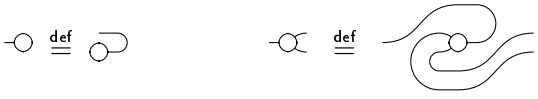
They always satisfy the equations defining compact structures.

There is an alternative definition of Frobenius algebras using compact structures that will be used in Chapter 5.

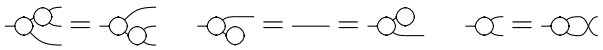
Lemma 1 (alternative definition of $\mathbb{F}$)**.** *We have:*

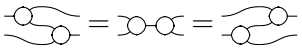
$$\mathbb{F} \simeq \{\mathbb{M} + \mathbb{S}\} \Big/ \left\{ \text{  } \right\}$$

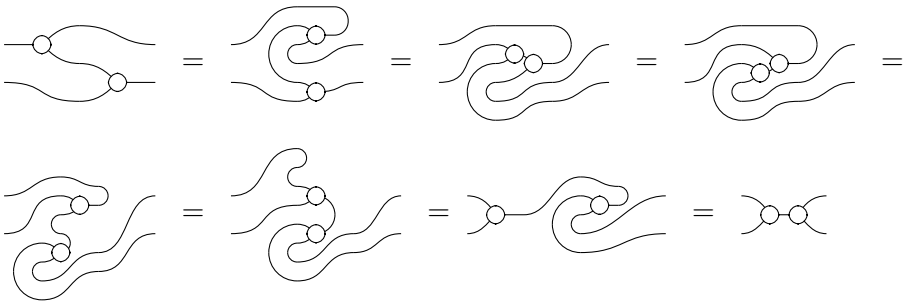
Proof. We start with the alternative definition and will show how to recover the original one. We define:



By transposing everything we directly obtain the equations:

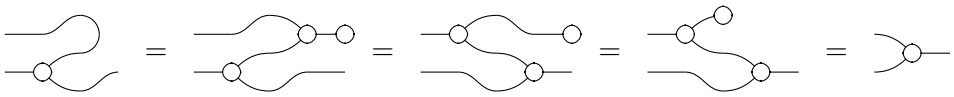


It remains to show  . We only show one side, the other can be shown symmetrically.



Conversely setting:

we have:



□

Note that in ZX calculus both Frobenius algebras induce the same compact structures:

In particular, the canonical compact structure being equal to the compact structure induced by the green Frobenius algebra we can bend the legs of green spiders. The same holds for red spiders.

Similar properties often referred to as the **Only Topology Matters** paradigm, will be addressed in depth in Chapter 5.

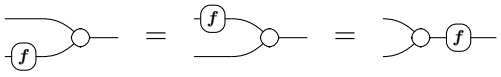
3.2 Phases

Starting from a Frobenius algebra in a monochromatic prop we can define a family of arrows that interact nicely with spiders.

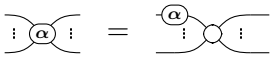
3.2.1 Definition

Intuitively phases are invertible $1 \to 1$ arrows that can go through spiders:

Definition 33. *Phases A* ***phase*** *of a Frobenius algebra is an invertible arrow* $: 1 \to 1$ *such that:*



This definition allows to introduce without ambiguity generalized spiders indexed by phases:

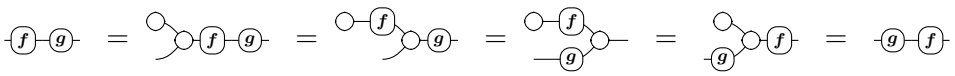


Those are the spiders that will occur in ZX calculus.

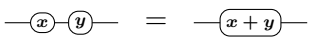
3.2.2 Phase groups

Phases always form a group. In fact, the composite of two phases is a phase, the identity $id_1 : 1 \to 1$ is a phase and all phases are invertible and those inverse are also phases. Thus, in any prop, when we have a Frobenius algebra we have an associated phase group G.

So we represent phases as spiders $1 \to 1$ indexed by elements of the phases group. The phase group is Abelian.

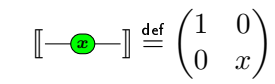
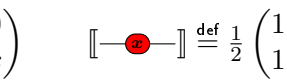


So we will use an additive notation for the phase group. This gives:

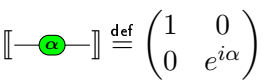
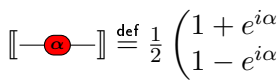


Note that given an invertible scalar, that is an invertible arrow $s : 0 \to 0$ then $s \otimes id_1$ is a phase. So the group of invertible scalars is always a subgroup of the phase group.

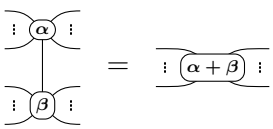
In ZX calculus up to invertible scalars the phase groups of the green and red Frobenius algebras are both isomorphic to $\mathbb{C}^*$. Phases have the following form:

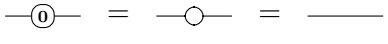
 $\overset{\text{def}}{=} \begin{pmatrix} 1 & 0 \\ 0 & x \end{pmatrix}$  $\overset{\text{def}}{=} \frac{1}{2}\begin{pmatrix} 1+x & 1-x \\ 1-x & 1+x \end{pmatrix}$

However the ZX calculus does not use all those phases but only a subgroup isomorphic to $\mathbb{U}$, the group of modulus one complex numbers. The phases that will be used are of the form:

 $\overset{\text{def}}{=} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}$  $\overset{\text{def}}{=} \frac{1}{2}\begin{pmatrix} 1+e^{i\alpha} & 1-e^{i\alpha} \\ 1-e^{i\alpha} & 1+e^{i\alpha} \end{pmatrix}$

Then we can compose them by taking addition modulo 2π. And even extend this to spider fusion:

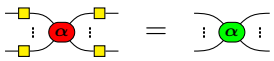


Note that:  .

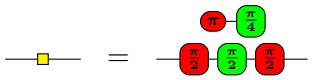
3.2.3 Euler rule and Hadamard

ZX calculus contains two different Frobenius algebras and then two families of phases. We can go from red to green and back using the Hadamard gate which is denoted in ZX:

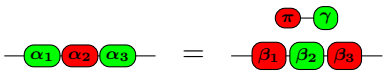
With $[\![$$]\!] \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$. We have:



This provides a symmetry between red and green called **colour swap**, ensuring that given any equation, we can obtain a second one by colour swapping, that is, exchanging red and green in the diagram. The Hadamard gate is here a generator but we have already seen in Chapter 2 that it can in fact be obtained from the red and green phases.

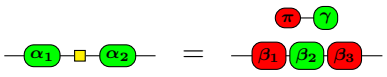


Following the same road as in Chapter 2 this leads us to consider the **Euler** rule for ZX calculus.



Where: $x^+ \stackrel{\text{def}}{=} \frac{\alpha_1+\alpha_3}{2}$, $x^- \stackrel{\text{def}}{=} x^+ - \alpha_3$, $z \stackrel{\text{def}}{=} \cos(\frac{\alpha_2}{2})\cos(x^+) + i\sin(\frac{\alpha_2}{2})\cos(x^-)$, $z' \stackrel{\text{def}}{=} \cos(\frac{\alpha_2}{2})\sin(x^+) - i\sin(\frac{\alpha_2}{2})\sin(x^-)$, $\beta_1 \stackrel{\text{def}}{=} \arg(z) + \arg(z')$, $\beta_2 \stackrel{\text{def}}{=} 2\arg(i + |\frac{z}{z'}|)$, $\beta_3 \stackrel{\text{def}}{=} \arg(z) - \arg(z')$ and $\gamma = x^+ - \arg(z) + \frac{\pi-\beta_2}{2}$.

In fact, it has been shown in [54] that a more general rule encompassing the decomposition of the Hadamard gate can be given, this is the one we will keep from now on:



Where: $x^+ \stackrel{\text{def}}{=} \frac{\alpha_1+\alpha_2}{2}$, $x^- \stackrel{\text{def}}{=} x^+ - \alpha_2$, $z \stackrel{\text{def}}{=} -\sin(x^+) + i\cos(x^-)$, $z' \stackrel{\text{def}}{=} \cos(x^+) - i\sin(x^-)$, $\beta_1 \stackrel{\text{def}}{=} \arg(z) + \arg(z')$, $\beta_2 \stackrel{\text{def}}{=} 2\arg(i + |\frac{z}{z'}|)$, $\beta_3 \stackrel{\text{def}}{=} \arg(z) - \arg(z')$ and $\gamma = x^+ - \arg(z) + \frac{\pi-\beta_2}{2}$.

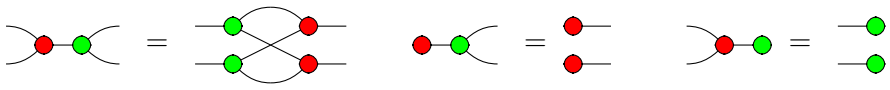
Note what we have seen so far is already enough to provide a ZX calculus that is universal and complete for one qubit unitaries. We will now enrich the calculus with multi-qubit gate consideration that will need to the completeness result for **Lin**.

3.3 The calculus

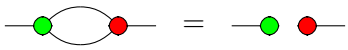
Now that we understand the building blocks of ZX we make them interact.

3.3.1 Interactions

The two Frobenius algebra of ZX have a very special relation called **strong complementarity** and corresponding to the following equations:

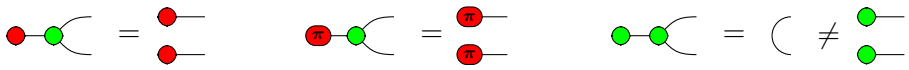


From those equations we can deduce the following useful fact called the **Hopf rule**:

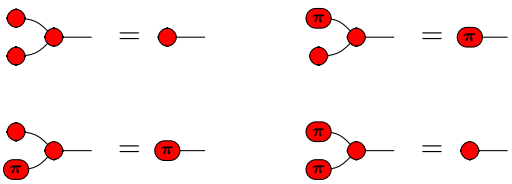


In ZX calculus the states of the computational basis are represented by red spiders up to a normalization scalar.

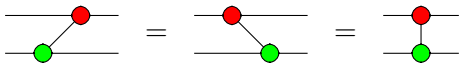
Then the copy rule allows to see the green spider as a copying mechanism. This is not contradictory with the no-cloning theorem, this copy can only duplicate the computational basis, not arbitrary states. For example, it does not duplicate $|+\rangle$ that corresponds up to a scalar to a green one-legged spider.



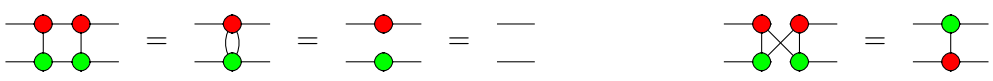
The red spider is closely linked to the Xor gate, indeed:



From this, it follows that the CNot admits a very nice representation in ZX calculus.



We again used vertical wires since here there is no ambiguity. This allows to give interpretations of the bi-algebra and Hopf rules. If we rotate the two rules see that the Hopf rule ensures that the CNot is an involution and the bi-algebra rule corresponds to the identity relating swaps and three CNots.

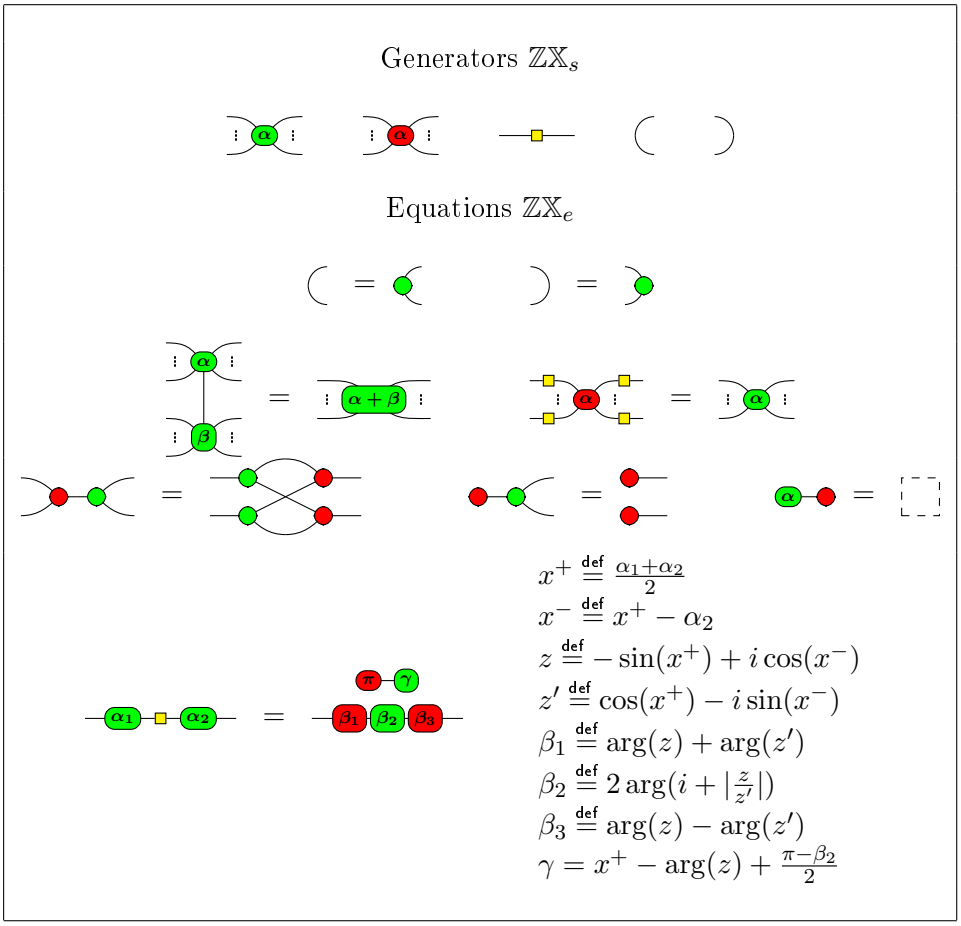


Hopf rule *Bi-algebra rule*

3.3.2 Completeness

It is now time to formally give the generators and equations of ZX calculus.

The graphical language $\mathbb{ZX}$.



All the rule that we have presented so far follow from these. We now sum up the interpretations we have given so far we have:

$$\left[\!\left[\right]\!\right] \stackrel{\text{def}}{=} 2^{\frac{n+m-2}{4}} \left(|0\rangle^n \langle 0|^m + e^{i\alpha} |1\rangle^n \langle 1|^m\right)$$

$$\left[\!\left[\right]\!\right] \stackrel{\text{def}}{=} 2^{\frac{2-n-m}{4}} \sum_{x \in \mathbf{2}^{n+m}} \frac{1+e^{i\alpha+i\left(\bigoplus_{i=0}^{n+m} x_i\right)}}{2} |x_1, ..., x_n\rangle \langle x_{n+1}, ..., x_{n+m}|$$

$$\left[\!\left[\right]\!\right] \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \left[\!\left[\right]\!\right] \stackrel{\text{def}}{=} \sum_{x,y \in \mathbf{2}} |xy\rangle \langle yx|$$

$$\left[\!\left[\right]\!\right] \stackrel{\text{def}}{=} |00\rangle + |11\rangle \qquad \left[\!\left[\right]\!\right] \stackrel{\text{def}}{=} \langle 00| + \langle 11|$$

With this interpretation, the set of rules is complete for **Lin**.

Theorem 3 (Completeness)**.** *Given two diagram D and D' over the signature $\mathbb{ZX}_s$:*

$$[\![D]\!] = [\![D']\!] \qquad \Leftrightarrow \qquad \mathbb{ZX}_\pi(D) = \mathbb{ZX}_\pi(D')$$

Proof. See [54] □

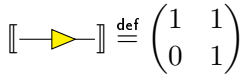
So in theory we could completely replace matrices with diagrams. Of course, there is no magic here. If we look to normal forms for ZX calculus like the one of [55], we end up seeing the matrix appearing in the diagram. However, in some specific situations, ZX calculus has a clear advantage over linear algebra.

3.3.3 Variations

The ZX-calculus we presented here is only one among many variations.

First, one can change the normalisation, the scalars in the interpretation of generators. This makes scalars appearing and disappearing in the rules. The well-tempered normalisation of [52] I choose here has the advantage that the yellow box is really the Hadamard gate and does the colour swap in a clean way. Furthermore, the CNot as a scalar free representation and copies, bi-algebra, and Hopf rules do not add any scalars. However the two main drawbacks are that we can't obtain the computational basis states without floating scalars and the Frobenius algebras are not special, we need to add a scalar to remove loops. I personally prefer this normalization since it provides a direct connection with circuits and simplifies the computation since copies and bi-algebras are ubiquitous while loops are in my experience rare.

There are also alternative sets of rules that are complete. Most of them exist for historical reasons since the power of the Euler rule was not discovered directly. See [47] for an history of the completeness results. There are also variations of ZX adding new generators to simplify the rules. This is the case of the ΔZX of [56] which introduces the **triangle**:



Variations also exist while the whole phase group of the Frobenius algebra is used and not only modulus-one complex numbers [57]. Even more exotic variations allow generalized phases indexed by quaternions [58].

Finally, numerous calculi have been developed for fragments, that is calculi which are complete and universal for subcategories of **Lin** like [59] or [12]. The set of rules can be simpler since they were for most of them intermediary steps in the long road toward the completeness for **Lin**.

Part II

Only Topology Matters

Chapter 4

Paradigms in graphical language design

You should not have a favourite weapon. To become over-familiar with one weapon is as much a fault as not knowing it sufficiently well. You should not copy others, but use weapons which you can handle properly.

Miyamoto Musashi in [60]

In this thesis, we choose as a definition of graphical language a prop presented by generators and equations. But this definition is too general for some purposes. Graphical proof assistants like Quantomatic [49] and PyZX [61] allow to manipulate graphical languages and can be used to check or even automatically find graphical proofs. When the ΔZX-calculus of [56] was implemented in Quantomatic, the triangle generator, that we introduce in Chapter 3, was an obstacle. In fact, both Quantomatic and PyZX consider the generators in string diagrams as vertices in a graph, imposing equations on the triangle which are not true in ΔZX-calculus. Here we see clearly that arbitrary props presented by generators and equations were clearly not the definition of graphical languages the developers of those softwares had in mind. In this chapter, we introduce a notion of paradigm which allows to consider restricted families of graphical languages without rebuilding from scratch the theory introduced in Chapter 1. A paradigm defines some canonical generators and equations that are assumed to be true in any paradigmatic graphical language. Those generators and equations are then considered as fundamental and will not appear explicitly in the definitions of graphical languages once it has been stated that we work in a given paradigm. To provide a concrete example, monoids are semi-groups that always come with an additional generator, the unit, and additional equations, the unit laws. We can see monoids as a paradigm over semigroups and then work in the framework of monoids where the unit is always implicitly there and satisfies the same implicit unit laws. This allows to simplify the presentation of props by hierarchizing the equations and emphasizing the ones that really matter. Paradigms also allow to organise in a unique framework different approaches to graphical languages. The paradigm corresponding to the graphical languages of Quantomatic and PyZX will be introduced and studied in detail in Chapter 5. I expect the connection between paradigms and graphical proof assistants softwares to be more general. In fact, a paradigm corresponds to a notion of free paradigmatic prop. Characterizing those free paradigmatic props allows to design dedicated data structures and algorithms for automatic rewriting.

4.1 Definition

The definition of paradigm relies heavily on the categorical framework of Chapter 1. The goal here is to define everything in such a way that the properties of props transfer nicely to paradigmatic props. For the rest of this Chapter, we fix a set of colours C.

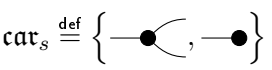
4.1.1 Paradigmatic generators

We want to promote some generators as more fundamental than the others.

Definition 34 (Paradigmatic generators)**.** *We define a signature $\mathfrak{p}_s$ gathering **Paradigmatic generators**.*

Since we want paradigmatic generators to be present in any paradigmatic graphical language we will require that the signature of the paradigmatic graphical languages is of the form: $\Sigma + \mathfrak{p}_s$ where Σ can be any signature. In practice, we will present paradigmatic generators as we present any signature. Like in Chapter 1, we will use a running example, the **Cartesian paradigm** denoted $\mathfrak{car}$.

Example 7. *The paradigmatic generators of $\mathfrak{car}$ are gathered in the signature:*

$\mathfrak{car}_s \stackrel{\text{def}}{=}$ 

4.1.2 Paradigmatic equations

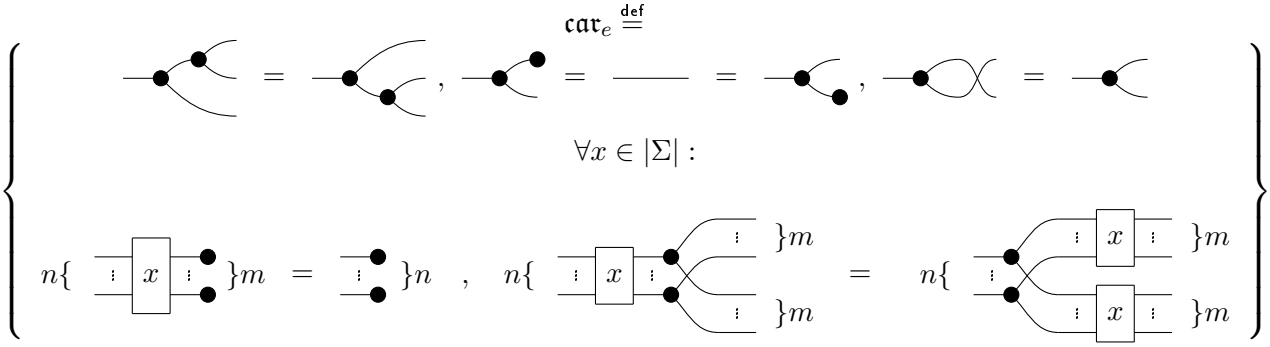
The definition of paradigmatic equations is more subtle since they are allowed to depend on some generators.

Definition 35. *A **family of paradigmatic equations** is a tuple $\mathfrak{p}_e \stackrel{\text{def}}{=} \{\mathfrak{p}^n, \mathfrak{p}^\ell, \mathfrak{p}^r\}$ where $\mathfrak{p}^n : C\text{-}\mathbf{Sig} \to C\text{-}\mathbf{Sig}$ is a functor and $\mathfrak{p}^\ell, \mathfrak{p}^r : \mathfrak{p}^n \Rightarrow UF(\_ + \mathfrak{p}_s)$ are two natural transformations defining for each signature Σ a family of equation $\mathfrak{p}(\Sigma)_e$ over the signature $\Sigma + \mathfrak{p}_s$ defined as: $\mathfrak{p}(\Sigma)^n_e \stackrel{\text{def}}{=} \mathfrak{p}^n\Sigma$, $\mathfrak{p}(\Sigma)^\ell_e \stackrel{\text{def}}{=} \mathfrak{p}^\ell{}_\Sigma$ and $\mathfrak{p}(\Sigma)^r_e \stackrel{\text{def}}{=} \mathfrak{p}^r_\Sigma$.*

Lets look more closely at this definition. The signature Σ contains the non-paradigmatic generators. The paradigmatic equations involve at the same time paradigmatic and non-paradigmatic generators so they must be over the signature $\Sigma + \mathfrak{p}_s$. The functor $\mathfrak{p}^n : C\text{-}\mathbf{Sig} \to C\text{-}\mathbf{Sig}$ associate to each signature Σ a set of names $\mathfrak{p}\Sigma^n_e = \mathfrak{p}_e(\Sigma)$ that will define the paradigmatic equations. It remains to define the left and right hand side signature maps. This is done by the natural transformations $\mathfrak{p}^\ell$ and $\mathfrak{p}^r$ whose components in Σ are of type $\mathfrak{p}(\Sigma)^n_e \to UF(\Sigma + \mathfrak{p}_s)$.

Informally, we require the naturality of $\mathfrak{p}^\ell$ and $\mathfrak{p}^r$ to ensure that the paradigmatic equations are defined uniformly from the signatures. Thus, two signatures with similar generators give similar equations. We will see formally how this is working later when we will tackle the categorical framework. In practice, we present a family of paradigmatic equations in the same way we present normal families of equations. The only difference is that explicit dependence in the generators in Σ will appear.

Example 8. *The paradigmatic family of equation $\mathfrak{car}_e$ is defined as:*



4.1.3 Paradigms

We can now define properly what a paradigm is.

Definition 36 (Paradigm). *A* ***paradigm*** $\mathfrak{p}$ *is a tuple* $\mathfrak{p} \stackrel{\text{def}}{=} (\mathfrak{p}_s, \mathfrak{p}_e)$ *where* $\mathfrak{p}_s$ *is a signature gathering paradigmatic generators and* $\mathfrak{p}_e$ *is a paradigmatic family of equations over the signature* $\mathfrak{p}_s$.

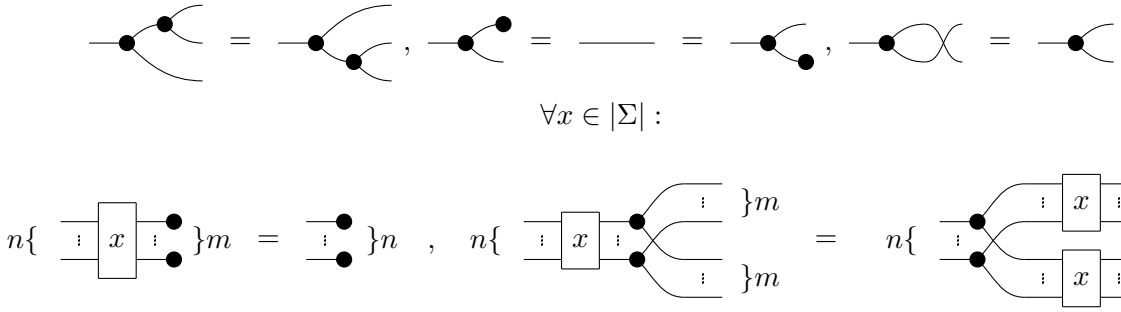
As for graphical language, in practice, we will not use the full categorical formalism but informally present the paradigm.

Example 9 (Cartesian paradigm). *The* ***Cartesian paradigm*** $\mathfrak{car}$ *over monochromatic props is presented as:*

The Cartesian paradigm $\mathfrak{car}$

Paradigmatic generators $\mathfrak{car}_s$

Paradigmatic equations $\mathfrak{car}_e$



We will come back to this paradigm and discuss it more in-depth later in this chapter.

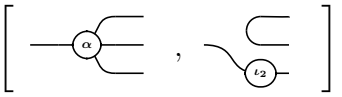
4.2 Paradigmatic graphical languages

We now focus on how paradigms allow us to define paradigmatic graphical languages. We fix a paradigm $\mathfrak{p}$.

4.2.1 Definition

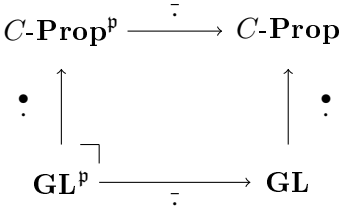
Definition 37 (Paradigmatic graphical language)**.** *A **Paradigmatic graphical language** is a graphical language $\mathcal{L}$ of the form $\mathcal{L}_s = \Sigma + \mathfrak{p}_s$ and $\mathcal{L}_e = E + \mathfrak{p}(\Sigma)_e$ where Σ is a signature and E is a family of equations over the signature $\Sigma + \mathfrak{p}_s$.*

The category $\mathbf{GL}^{\mathfrak{p}}$ of **paradigmatic graphical languages** is a subcategory of **GL**. The objects are the paradigmatic graphical languages. The arrows are the translations of the form:



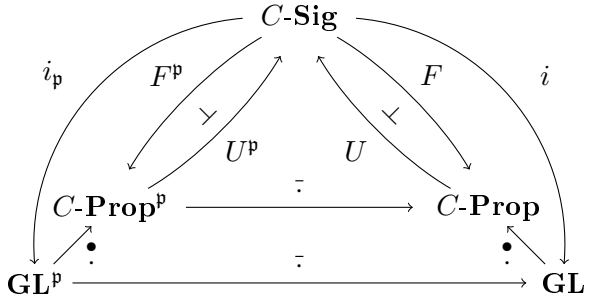
$: \Sigma + \mathfrak{p}_s \to UF(\Sigma' + \mathfrak{p}_s)$

They map paradigmatic generators to paradigmatic generators. We have a faithful inclusion functor $\bar{\cdot} : \mathbf{GL}^{\mathfrak{p}} \to \mathbf{GL}$. The category of **paradigmatic props** C-$\mathbf{Prop}^{\mathfrak{p}}$ is the image of $\mathbf{GL}^{\mathfrak{p}}$ by the functor $\overset{\bullet}{\cdot} \circ \bar{\cdot}$. It is a subcategory of C-**Prop** and the faithful inclusion functor is also denoted $\bar{\cdot} : C\text{-}\mathbf{Prop}^{\mathfrak{p}} \to C\text{-}\mathbf{Prop}$. We have a pull-back square ensuring the equation $\overset{\bullet}{\overline{\mathcal{L}}} = \overline{\overset{\bullet}{\mathcal{L}}}$:



4.2.2 The new F and U

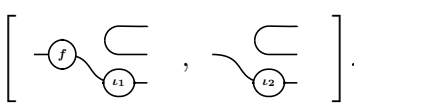
We construct the following diagram that we will explain in details:



Given a signature Σ we can construct the **free paradigmatic graphical language** over it $i_{\mathfrak{p}}(\Sigma)$.

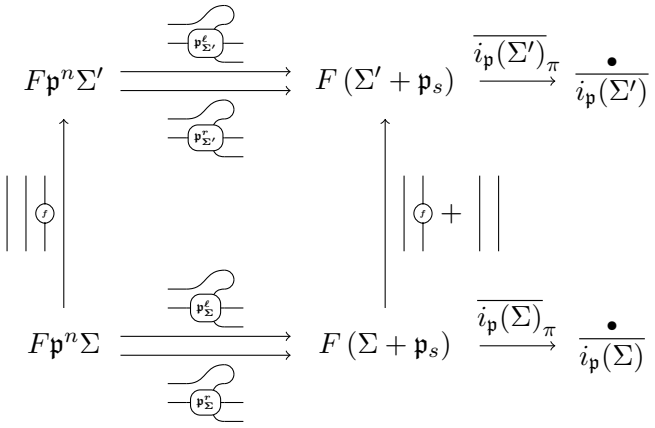
Lemma 2. *There is a functor $i_{\mathfrak{p}} : C\text{-}\boldsymbol{Sig} \to \boldsymbol{GL}^{\mathfrak{p}}$ defined by $i_{\mathfrak{p}}(\Sigma) \overset{\text{def}}{=} (\Sigma + \mathfrak{p}_s) \Big/ \mathfrak{p}(\Sigma)_e$ and:*

$$f \mapsto \left[\quad , \quad \right].$$



Furthermore: $\epsilon_{\Sigma' + \Sigma_{\mathfrak{p}}} \circ i_{\mathfrak{p}}(\alpha) = F(f + id_{\mathfrak{p}_s}) = F(f) + id_{F\mathfrak{p}_s}$.

Proof. The functoriality of $i_{\mathfrak{p}}$ follows from the definition of paradigmatic equations by natural transformations. The naturality enforces the soundness condition in the following diagram:



□

We already noticed in Chapter 1 that $\overset{\bullet}{\cdot} \circ i = F$. So we define $F^{\mathfrak{p}} \overset{\text{def}}{=} \overset{\bullet}{\cdot} \circ i_{\mathfrak{p}}$ and $U^{\mathfrak{p}} \overset{\text{def}}{=} U \circ \bar{\cdot}$. $F^{\mathfrak{p}}$ sends a signature to the free paradigmatic prop. $U^{\mathfrak{p}}$ is a forgetful functor from paradigmatic props to signatures.

4.2.3 The paradigmatic monadic adjunction

Our goal is now to show that $F^{\mathfrak{p}}$ and $U^{\mathfrak{p}}$ can be given the same status as F and U. That is, that we have a monadic adjunction $F^{\mathfrak{p}} \dashv U^{\mathfrak{p}}$. This will show that we can manipulate paradigmatic graphical languages in the same way we manipulate graphical languages.

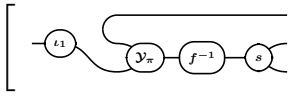
Theorem 4. *$F^{\mathfrak{p}} \dashv U^{\mathfrak{p}}$ is a monadic adjunction.*

Proof. We will use Beck's monadicity theorem to show that $U^{\mathfrak{p}}$ is monadic. See [32] for more details. We need three conditions:

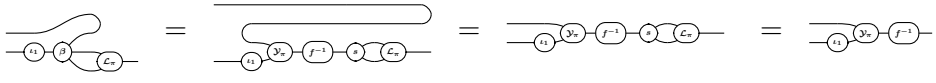
- $U^{\mathfrak{p}}$ is conservative. That is if $U^{\mathfrak{p}} f$ is an isomorphism then f is an isomorphism.

We have $U^{\mathfrak{p}} = U \circ \bar{\cdot}$, U is monadic and then conservative so we just have to show that $\bar{\cdot}$ is conservative. Let $f : P \to Q$ be an arrow in $\mathbf{Prop}^{\mathfrak{p}}$ such that $\overline{f}$ is an isomorphism. So there is an arrow f^{-1} in $\mathbf{Prop}$ such that $\overline{f} \circ f^{-1} = id_{\overline{Q}}$ and $f^{-1} \circ \overline{f} = id_{\overline{P}}$. We need to show that f^{-1} is in $\mathbf{Prop}^{\mathfrak{p}}$.

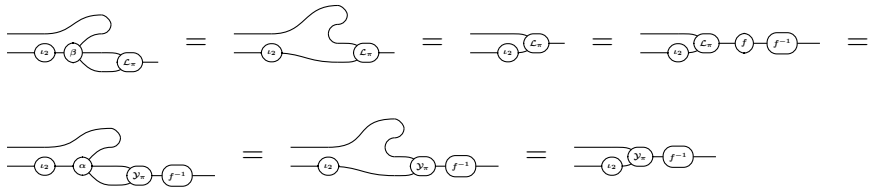
We will come back to graphical languages and translations. By definition, there are two paradigmatic graphical languages $\mathcal{L}, \mathcal{Y} : \mathbf{GL}^{\mathfrak{p}}$ such that $\overset{\bullet}{\mathcal{L}} = P$ and $\overset{\bullet}{\mathcal{Y}} = Q$, and a translation $\alpha : \mathcal{L} \to \mathcal{Y}$ such that $\overset{\bullet}{\alpha} = f$. We construct $\beta : \mathcal{Y} \to \mathcal{L}$ in $\mathbf{GL}^{\mathfrak{p}}$ such that $\overset{\bullet}{\beta} = f^{-1}$. Taking a section s of $\pi_{\mathcal{L}}$. We define:

$\beta \overset{\text{def}}{=}$ 

We now check the soundness condition. This will at the same time proves that β is a well defined translation and that $\overset{\bullet}{\beta} = f^{-1}$. We have:



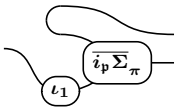
and



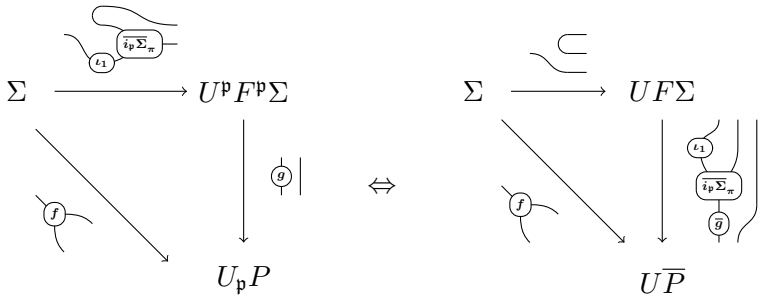
So the soundness condition holds from the universal property of $F(\mathcal{Y}_s) + F(\mathfrak{p}_s)$.

• $U^{\mathfrak{p}}$ has a left adjoint. We show that $F^{\mathfrak{p}}$ is this left adjoint.

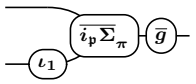
We check the universal property. Given a signature Σ. Using $U^{\mathfrak{p}}F^{\mathfrak{p}}\Sigma = U\,\overline{i_{\mathfrak{p}}\Sigma}^{\bullet} = U\,\overline{i_{\mathfrak{p}}\Sigma}^{\bullet}$, we define a morphism $\eta^{\mathfrak{p}}_{\Sigma} : \Sigma \to U^{\mathfrak{p}}F^{\mathfrak{p}}\Sigma$:



Given a paradigmatic prop P and a map $f : \Sigma \to U^{\mathfrak{p}}P$ we need to show there is a unique map satisfying $U^{\mathfrak{p}}F^{\mathfrak{p}}(f) \circ \eta^{\mathfrak{p}}_{\Sigma} = f$. We start by uniqueness. Let $g : F^{\mathfrak{p}}\Sigma \to P$ be a map such that $U^{\mathfrak{p}}g \circ \eta^{\mathfrak{p}}_{\Sigma} = f$.

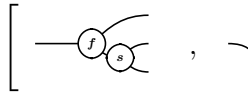


Since U has a left adjoint F, the universal property gives:

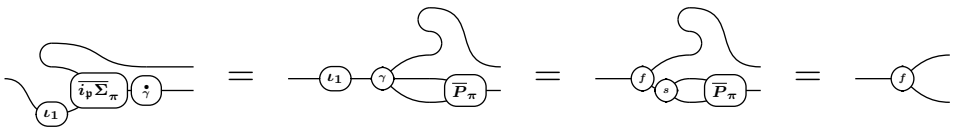
$Ff =$  .

Since $\overline{i_{\mathfrak{p}}\Sigma}_{\pi}$ is an epimorphism then $\overline{g} \circ \overline{i_{\mathfrak{p}}\Sigma}_{\pi}$ uniquely characterizes $\overline{g}$. Furthermore by definition of paradigmatic translations we know that $\overline{g} \circ \overline{i_{\mathfrak{p}}\Sigma}_{\pi} \circ F\iota_2$ doesn't depends on g. So by the universal property of the co-product $F(\Sigma)+F(\mathfrak{p}_s)$ a g satisfying this property is unique.

Now for existence, we take a section $s : \overline{P} \to F\overline{P}_s$ of $\overline{P}_{\pi}$ and define:

$\gamma \overset{\text{def}}{=}$  and $g \overset{\text{def}}{=} \overset{\bullet}{\gamma}$

It satisfies:



hence $F^{\mathfrak{p}} \dashv U^{\mathfrak{p}}$.

• $\mathbf{Prop}^{\mathfrak{p}}$ has and $U^{\mathfrak{p}}$ preserves co-equalizers of $U^{\mathfrak{p}}$-split pair.

We consider a $U^{\mathfrak{p}}$-split pair $f, g : P \to Q$. This means we have a split co-equalizer:

$$U^{\mathfrak{p}}P \underset{U^{\mathfrak{p}}g}{\overset{U^{\mathfrak{p}}f}{\rightrightarrows}} U^{\mathfrak{p}}Q \xrightarrow{h} \Delta \quad \Leftrightarrow \quad U\overline{P} \underset{U\overline{g}}{\overset{U\overline{f}}{\rightrightarrows}} U\overline{Q} \xrightarrow{h} \Delta$$

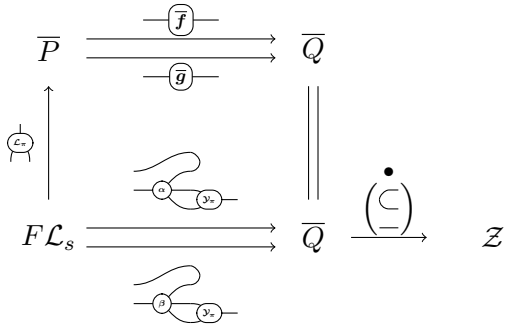
$\overline{f}$ and $\overline{g}$ form a U-split pair and then, since U is monadic we know that they admit a co-equalizer preserved by U:

$$\overline{P} \underset{\overline{g}}{\overset{\overline{f}}{\rightrightarrows}} \overline{Q}$$

We will construct a paradigmatic graphical language $\mathcal{Z}$ such that $\overset{\bullet}{\mathcal{Z}}$ is a co-equalizer of $\overline{f}$ and $\overline{g}$ in C-**Prop** and of f and g in C-$\mathbf{Prop}^{\mathfrak{p}}$.

Let $\mathcal{L}$ and $\mathcal{Y}$ be paradigmatic graphical languages such that $\overset{\bullet}{\mathcal{L}} = P$ and $\overset{\bullet}{\mathcal{Y}} = Q$. Since $\overset{\bullet}{\cdot}$ is full we can take two paradigmatic translations $\alpha, \beta : \mathcal{L} \to \mathcal{Y}$ such that $\overset{\bullet}{\alpha} = f$ and $\overset{\bullet}{\beta} = g$.

We define a family of equation $\mathcal{E} \overset{\text{def}}{=} (\mathcal{L}_s, \alpha, \beta)$ over the signature $\mathcal{Y}_s$. We define $\mathcal{Z} \overset{\text{def}}{=} \overline{\mathcal{Y}}/_{\mathcal{E}}$. We know that $\overset{\bullet}{\overline{\mathcal{Z}}}$ is a co-equalizer in C-**Prop**.



The soundness condition makes the diagram commute, and $\mathcal{L}_\pi$ being an epimorphism, it follows that $\overset{\bullet}{\overline{\mathcal{Z}}}$ is a co-equalizer of $\overline{f}$ and $\overline{g}$. Here, by construction, $\mathcal{Z}$ is a paradigmatic graphical language, and $\overline{\subset}$ is a paradigmatic translation.

It remains to show that we have a proper co-equalizer in C-$\mathbf{Prop}^{\mathfrak{p}}$. Given a paradigmatic prop R and a paradigmatic prop morphism t satisfying the property we have a unique prop morphism h such that $h \circ \overset{\bullet}{\left(\overline{\subset}\right)} = \bar{t}$. Let τ be a paradigmatic translation such that $\overset{\bullet}{\tau} = t$. We have:

$$\overset{\bullet}{\tau} \circ \overset{\bullet}{\left(\overline{\subset}\right)} = \tau \circ \overset{\bullet}{\left(\overline{\subset}\right)} = \overset{\bullet}{\tau} = t$$

so $\overset{\bullet}{\tau} = h$ and h is a paradigmatic prop morphism. One can be surprised that τ is a valid antecedent for h and t at the same time, in fact, the difference here is in the typing of the translation, the similarity follows from the fact that the codomain of the translations are different

graphical languages but with the same signature. So we see that $\mathbf{Prop}^{\mathfrak{p}}$ has co-equalizers and $\bar{\cdot}$ preserves them.

Finally the Beck monadicity theorem gives us a monadic adjunction $F^{\mathfrak{p}} \dashv U^{\mathfrak{p}}$. □

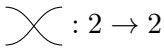
This adjunction provides a monad over C-**Sig**. We happen to be in the same situation as before with $F \dashv U$. Defining graphical languages in the same way we did in Chapter 1 with respect to this new monad gives us exactly $\mathbf{GL}^{\mathfrak{p}}$. We can now work with **paradigmatic graphical languages** where the paradigmatic generators and equations are implicitly embedded in the language. If needed, we can still use the functor $\bar{\cdot}$ to work in props and then show that we can go back in the paradigm.

4.3 Examples of paradigms

We end this chapter by giving some examples of paradigms.

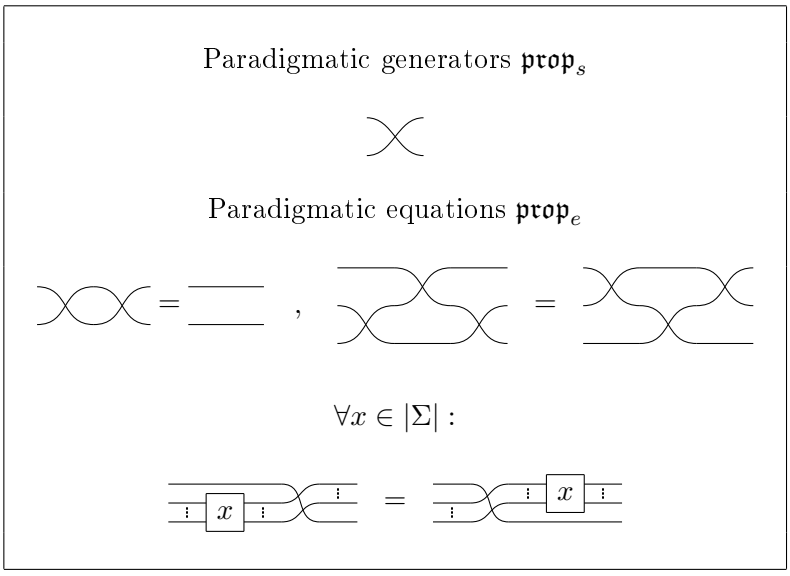
4.3.1 Props as a paradigm over pros

Our first example is a bit forced since we don't consider paradigm over props but pros. A **pro** is defined in exactly the same way as a prop but without any requirement on the presence of a symmetric structure. Presentations of pros can be defined exactly in the same way as presentation of props as it was done in Chapter 1. Thus we have a notion of graphical language without swaps. From here we can recover props as a paradigm. There is one paradigmatic generator, the swap:



and the paradigmatic equations assert the expected properties of a prop. note that if some paradigmatic equations only require paradigmatic generators, the paradigmatic equations ensuring naturality require all non-paradigmatic generators.

The Prop paradigm $\mathfrak{prop}$



Here asserting that all generators go through the swap is enough to recover the naturality of the swap by compositionality. However, we have to assert explicitly the Yang-Baxter equation, which does not directly follow since the swap is a paradigmatic generator.

Considering props as a paradigm over pros leads to the question of iterating paradigms. If in theory, the categorical machinery might work, in practice it could be tricky. For example, the paradigmatic generators of the first paradigm would not contribute to the paradigmatic equations of the second, such thing would have to be taken into account. To avoid such technicalities and to keep with what has been explicitly presented in the thesis we will always define clearly our paradigms in one step starting from props, even if some more involved presentation might be possible.

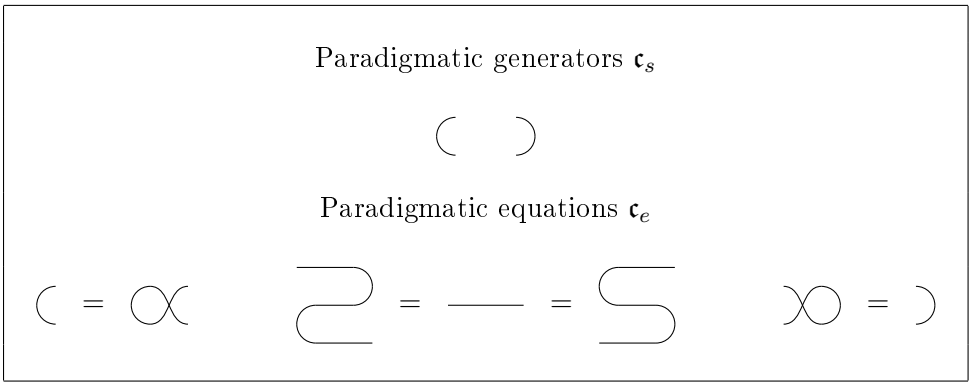
4.3.2 No paradigmatic equations

Given a graphical language $\mathcal{L}$ we obtain a paradigm $\mathfrak{l}$ by setting $\mathfrak{l}_s \overset{\text{def}}{=} \mathcal{L}_s$ and $\mathfrak{l}(\Sigma)_e \overset{\text{def}}{=} \mathcal{L}_e$. Such paradigms are simple in the sense that the paradigmatic equations only depend on the paradigmatic generators and do not involve non-paradigmatic ones.

The compact closed paradigm

We define a paradigm asserting that a prop is equipped with a compact structure. We call it the **compact closed** paradigm and the paradigmatic props will be called **compact closed props**. As expected, the paradigmatic generators as just the cup and the cap and the paradigmatic equations ensure that those two form a compact structure.

The compact closed paradigm $\mathfrak{c}$



Working in this paradigm there are additional things we need to take into account compared to ordinary props. A translation will automatically send the canonical compact structure to the canonical compact structure. So we need to keep in mind that two props can be equivalent as props but different as compact closed props since they don't have the same canonical compact structure. In general, when we give an interpretation for a paradigmatic prop we always have to precise what are the canonical paradigmatic generator in the model.

This paradigm will be used extensively in Chapter 5.

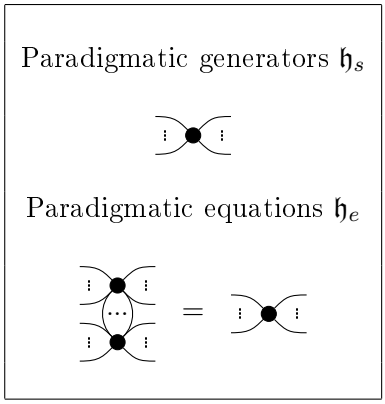
Hyper-graph categories

Hyper-graph categories were defined numerous times and given various names, see [62] for details. Restricting ourselves to props, the idea is that we are given a canonical special Frobenius algebra

that is informally considered as an extension of wire. Then more than two generators can be connected to the same wire, in the same way, that more than two vertices can be connected to the same hyper-edge in a hyper-graph hence the name. Such situations are common when copying and sharing data is an option. Another interesting example are the nodes of an electric circuits diagram which are required to satisfy Kirchhoff's law [36].

The **hyper graph paradigm** $\mathfrak{h}$ is defined as:

The hyper graph paradigm $\mathfrak{h}$

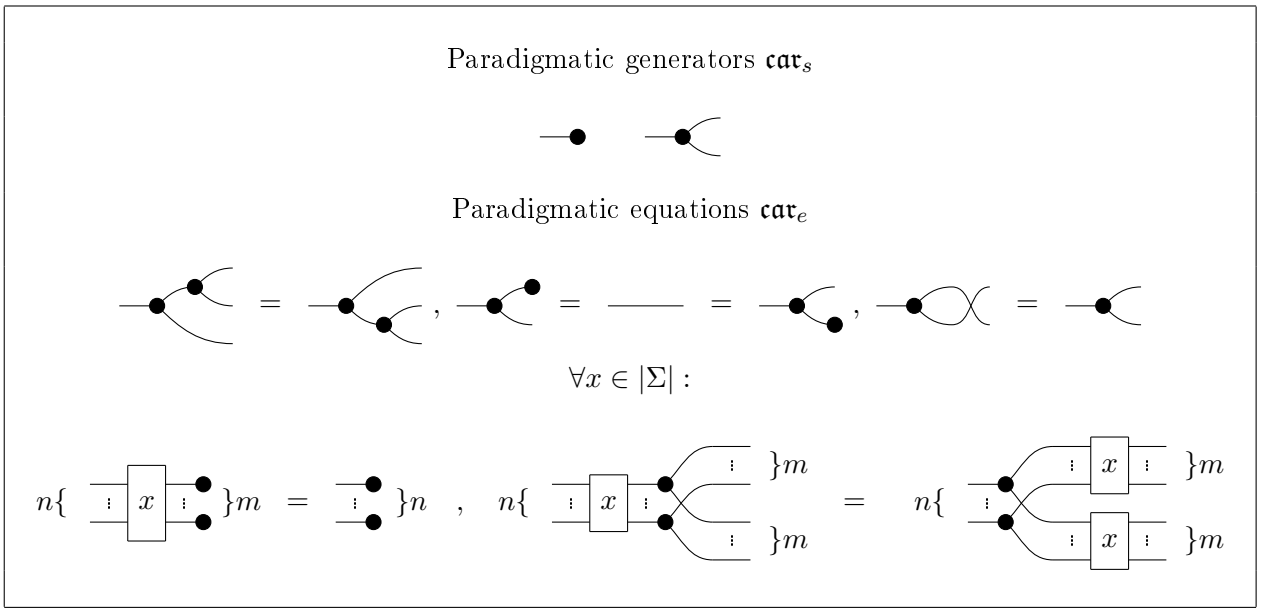


Here we used the spider notation for Frobenius algebras introduced in Chapter 3. As for the paradigm $\mathfrak{c}$, the paradigmatic equations do not depend on non-paradigmatic generators.

4.3.3 Cartesian paradigm

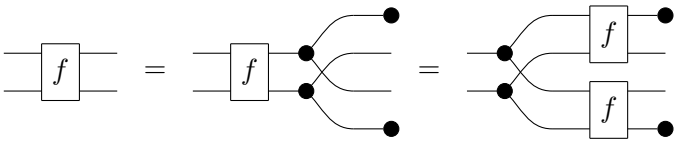
Another way to build paradigms is to state that all generators must be morphisms of a given structure. This is the case for the Cartesian paradigm.

The Cartesian paradigm $\mathfrak{car}$



Cartesian props are exactly the props in which the tensor product is a product and the tensor unit a terminal object, we call them. The projections can be defined as:

With this definition we have for all $f : n \to m$:



Which implies for all $f, g : n \to m$:

$$\pi_1 \circ f = \pi_1 \circ g \quad \wedge \quad \pi_2 \circ f = \pi_2 \circ g \quad \Rightarrow \quad f = g$$

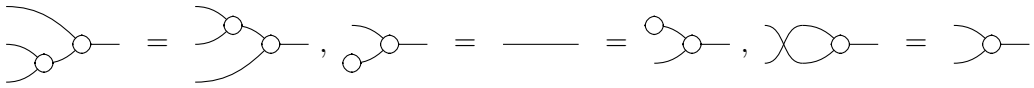
Conversely, the diagonal and terminal maps satisfy the equations, and then any prop with a product as tensor and a terminal object as tensor unit fits into the Cartesian paradigm.

The dual paradigm $\mathfrak{car}^{op}$ is also interesting.

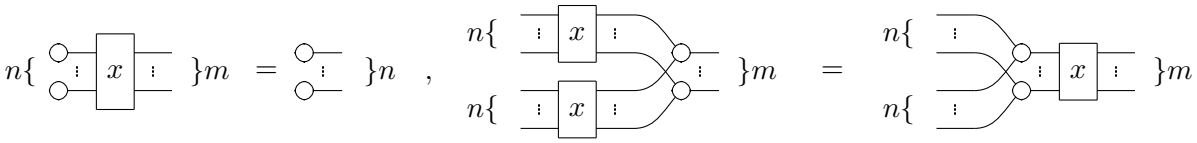
The co-Cartesian paradigm $\mathfrak{car}^{op}$

Paradigmatic generators $\mathfrak{car}_s^{op}$

Paradigmatic equations $\mathfrak{car}_e^{op}$



$\forall x \in |\Sigma| :$



A model of this paradigm in set is given by taking the addition on $\mathbb{R}$ as white monoid. Then the paradigm exactly ensures that all generators are linear maps. This duality between copy and addition will be developed further in Chapter 7.

Chapter 5

Flexsymmetry

Dans une thèse, il faut inventer un mot.[a]

[a] You have to coin at least one word in a PhD.
Emmanuel Jeandel [63]

I had numerous discussions about how to hierarchize the equations of ZX-calculus, arguing about which equation follows from the axioms of Frobenius algebra, which from the axioms of props, *etc...*, those shaped the presentation I gave in Chapter 3. A connected discussion was about the Only Topology Matter paradigm (here paradigm doesn't have the formal sense of Chapter 4) mentioned in [64] and later rebranded Only Connectivity Matter. This informal paradigm states that diagrams can be manipulated as graphs. Indeed, a great amount of the elegance of graphical languages like the ZX-[64], ZW-[65] and ZH-[66] comes from this fact. This allows intuitive graphical manipulations and then simpler implementation into graphical proof assistant softwares [49]. A formal definition of Only Topology Matter has been made in [67], but does not rely on equational theories. The main motivation to define flexsymmetry was to provide this equational characterization. This is in fact what led to rigorously define the paradigms of Chapter 4 in the first place.

Flexsymmetry has a strong link with Frobenius algebras. Frobenius algebras can interact with various compact structures being flexsymmetric with respect to some and not with respect to others. So some graphical calculi of interest, even if they are built from Frobenius algebras, fail to satisfy the only topology matter paradigm in a broad sense. Examples are the ΔZX-calculus of [56], the qutrit ZX-calculus of [68] or the graphical linear algebra of [20] which will be developed in more details in Chapter 7. Investigating those led to equations very similar to the ZW-calculus of [65] and suggested a way to modify their axiomatisation to recover flexsymmetry.

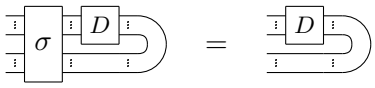
5.1 Introducing flexsymmetry

Our goal is to define a paradigm such that paradigmatic diagrams behave essentially as graphs. From now on we work in the compact closed paradigm $\mathfrak{c}$ defined in Chapter 4. So all graphical languages come with a canonical symmetric compact structure.

5.1.1 Flexsymmetric generators

Given a graphical language $\mathcal{L}$ we define what it means to be able to exchange the inputs and outputs of a diagram.

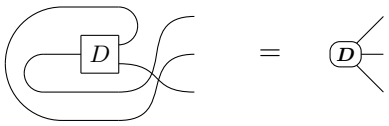
Definition 38 (Flexsymmetry). *Let $D : n \to m$ be a diagram in a compact closed graphical language $\mathcal{L}$. D is said **flexsymmetric** if the equation:*



holds in $\mathcal{L}$ for all permutations of $n + m$ wires σ.

In this definition, the permutation box σ represents a combination of swaps that implement the corresponding permutation between the wires. Such a diagram is unique up to prop axioms. It follows directly from this definition that the cup the cap and the identity are flexsymmetric. However, the swap isn't. Note that this property is absolutely not compositional, we can perfectly obtain non flexsymmetric diagrams from flexsymmetric ones and conversely.

When a diagram is flexsymmetric we can think of it as one big graph vertex. In practice, this corresponds to a simplified representation for a flexsymmetric diagram D:



Note that here we just choose a particular permutation of the wire but they are all equivalent by flexsymmetry.

5.1.2 Flexsymmetric paradigm

The flexsymmetric paradigm then ensures that all generators of a language satisfy the flexsymmetry equation.

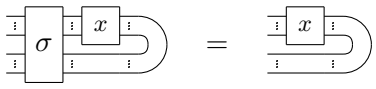
Definition 39 (Flexsymmetric paradigm). *The **flexsymmetric paradigm** $\mathfrak{f}$ is defined by:*

The flexsymmetric paradigm $\mathfrak{f}$

Paradigmatic generators $\mathfrak{f}_s$

Paradigmatic equations $\mathfrak{f}_e$

$\forall x \in |\Sigma|$:



for all σ, permutation of $n+m$ elements.

Notice that only the generators are required to be flexsymmetric not all diagrams nor the swap.

Following Chapter 4, this allows to define flexsymmetric graphical languages and flexsymmetric props. By definition, a flexsymmetric prop is a prop that can be axiomatized by a flexsymmetric graphical language. This is clearly not the case with all props. An easy example is given by taking any prop and freely adding a $1 \to 1$ arrow which, by definition, will cause any attempt to enforce the flexsymmetry equation to fail.

5.1.3 Flexsymmetry and Frobenius algebras

The flexsymmetry paradigm provides a new point of view on Frobenius algebras. First, the language $\mathbb{F}$ of Frobenius algebras defined in Chapter 3 fits in the paradigm. In fact, given a Frobenius algebra, we directly have a symmetric compact structure and all generators are flexsymmetric with respect to this compact structure. The converse question is: how simple can be the axiomatization of $\mathbb{F}$ if we remove the redundant equations that follow from flexsymmetry? The answer is surprisingly nice.

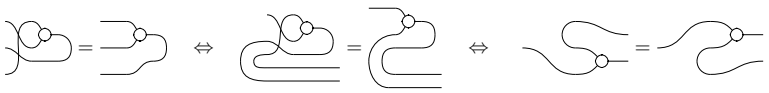
Lemma 3. *Let $\mathbb{M}^{\mathfrak{f}}$ be the graphical language of monoids $\mathbb{M}$ seen in the flexsymmetric paradigm $\mathfrak{f}$. We have:*

$$\mathbb{F} \simeq \overline{\mathbb{M}^{\mathfrak{f}}}$$

Proof. We just have to use the alternative definition of Frobenius algebras given in Chapter 3.

First, ○— is directly flexsymmetric, since there are no non-trivial permutations on one wire.

So we just show that the flexsymmetric equation is equivalent to the one from the alternative definition. One can generate the group of permutations on three elements using one transposition and one 3-cycle. The monoid being commutative, this directly gives us the flexsymmetric equation for one transposition, the one exchanging the two inputs of the multiplication. It only remains to show it for a 3-cycle:



□

So in the flexsymmetric paradigm if we need spiders, instead of specifying the entire Frobenius algebra, we only have to provide half of it, that is, a monoid. Thus, in the flexsymmetric paradigm, the shortcuts for spider and spider fusion will design the signature and equations of monoids and not Frobenius algebras.

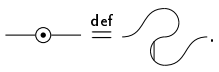
5.2 Flexsymmetrisation

In the introduction of this chapter, I mentioned various languages that fail to fit in the flexsymmetric paradigm. In fact, under some conditions, it is possible to fix this by slightly modifying the graphical language.

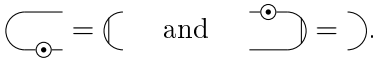
5.2.1 Flexsymmetry up to dualizers

The definition of flexsymmetry implies the choice of a canonical symmetric compact structure. However, distinct symmetric compact structures may be available in the same prop. We will study a particular but very common case when the notions of flexsymmetry with respect to two different compact structures are linked. We first look at how we can switch between two compact structures by slightly modifying a graphical language.

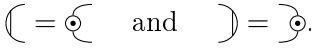
Given a prop with two symmetric compact structures $\left\{ \cup , \cap \right\}$ and $\left\{ \cup , \cap \right\}$, the **dualizer** $d : 1 \to 1$ is defined as:



It is an invertible map with inverse  . It is self transpose for both compact structure. Furthermore we have:



We say the the two structures are **compatible** if the dualizer is an involution. In this case we can use the notations:



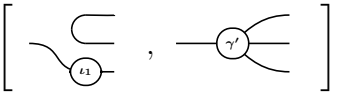
Notice that, conversely, given a self transpose involution for a given compact structure, we can, in the same way, construct a new compatible one whose dualizer will be the considered involution.

Our goal is to replace the canonical compact structure in a compact closed graphical language with another compatible one without changing the prop. We then replace all occurrences of cups and caps in the equations with the modified version composed or pre-composed with the dualizer d. This is simple but making it formal requires some care, especially if the diagram representing the dualizer itself involves cups and caps. To avoid such circles we first add freely a generator. We then implement the substitution on all equations but with the new generator instead of the

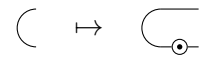
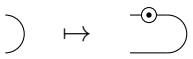
dualizer. Finally, we will quotient by an additional equation that identifies the dualizer and the new generator.

Now we take the time to formalise this strategy. Let $\mathcal{L}$ be a compact closed graphical language. We have $\overline{\mathcal{L}} = (\mathcal{L}_s + \mathfrak{c}_s) \Big/ (\mathcal{L}_e + \mathfrak{c}(\Sigma)_e)$. We define a free graphical language $\mathbb{Y} \stackrel{\text{def}}{=} \{Y : 1 \to 1\}$ with only one $1 \to 1$ generator and no equations. The language $\overline{\mathcal{L}} + \mathbb{Y}$ is then exactly the language $\mathcal{L}$ to which is added freely a generator $1 \to 1$.

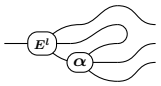
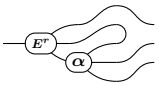
We define a signature map $\mathcal{L}_s + \mathfrak{c}_s \to UF\,(\mathcal{L}_s + \mathfrak{c}_s + \mathbb{Y})$ which modifies the compact structure:

$\gamma \stackrel{\text{def}}{=}$ 

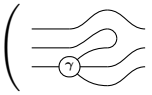
Where:

$\gamma' :$  $\qquad \gamma' :$ 

With $Y : 1 \to 1$ pictured as dotted node. We use this signature map to turn the family $\mathcal{L}_e$ of equations over $\mathcal{L}_s + \mathfrak{c}_s$ into a family $s_Z\mathcal{L}_e$ of equations over the signature $\mathcal{L}_s + \mathbb{Y} + \mathfrak{c}_s$ that will add the dualizer to the compact structure. Our goal in doing this is that since the dualizer is involutive, replacing the compact structure in $s_Z\mathcal{L}_e$ gives back the equations of $\mathcal{L}_e$. Given a family of equations E over a signature Σ and a signature map $\alpha : \Sigma \to UF\Sigma'$ we defined the family of equations $\alpha \circ E$ over the signature Σ' as:

$(\alpha \circ E)^n \stackrel{\text{def}}{=} E^n \qquad (\alpha \circ E)^l \stackrel{\text{def}}{=}$  $\qquad (\alpha \circ E)^r \stackrel{\text{def}}{=}$ 

The idea is that $\alpha \circ E$ is the same as E but where the left and right hand-side diagrams have been replaced by their translations through α.

For convenience, I will write γZ instead of  (Z) for the diagram Z where the signature map γ has been applied to all generators.

Definition 40 (cup-cap switch)**.** *Let $\mathcal{L}$ be a graphical language under the paradigm $\mathfrak{c}$. Let $Z \in UF(\mathcal{L}_s + \mathfrak{c}_s)$ be a diagram such that $\mathcal{L}_\pi Z$ is a self-transposed involution in $\overset{\bullet}{\mathcal{L}}$. The **cup-cap switch** of the graphical language $\mathcal{L}$ by the dualizer Z is defined as:*

$$s_Z\mathcal{L} \stackrel{\text{def}}{=} (\mathcal{L}_s + \mathfrak{c}_s + \mathbb{Y}) \Big/ (\gamma \circ \mathcal{L}_e + \mathfrak{c}_e + \gamma \circ \mathfrak{c}_e + \{Y = \gamma Z\})$$

Note that since $\mathfrak{c}(\mathcal{L}_s)_e = \mathfrak{c}(\mathcal{L}_s + \mathbb{Y})_e$ then we just write $\mathfrak{c}_e$ and $s_Z\mathcal{L}$ is a paradigmatic graphical language for $\mathfrak{c}$.

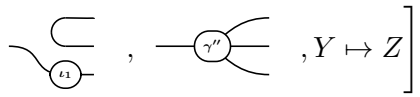
In practice the dualizer will often be a generator of $\mathcal{L}$ and then the process really just amounts to replace the cups and caps. The cup-cap switch leads to an equivalent language.

Lemma 4. *$\mathcal{L} \simeq s_Z\mathcal{L}$.*

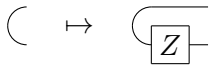
Proof. We show that γ defines a translation $\mathcal{L} \to s_Z\mathcal{L}$ providing an isomorphism of props $\overset{\bullet}{\gamma}$.

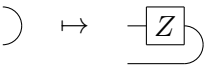
First we need to check the soundness condition, in other word that the equations of $\mathcal{L}$ transported by γ still hold in $s_Z\mathcal{L}$. This holds directly since $\mathcal{L}_e$ and $\mathfrak{c}_e$ transformed into $\gamma \circ \mathcal{L}_e$ and $\gamma \circ \mathfrak{c}_e$ which are among the equations of $s_Z\mathcal{L}$. So γ can be seen as a translation $\mathcal{L} \to s_Z\mathcal{L}$.

Now we construct an inverse defined as:

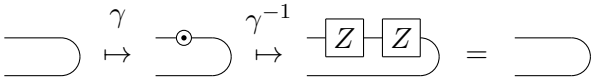
$\gamma^{-1} \stackrel{\text{def}}{=}$ 

Where:

$\gamma'':$ 

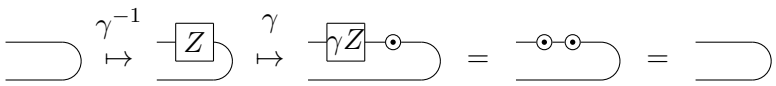
$\gamma':$ 

We have $\gamma^{-1} \circ \gamma = id$ as translation:

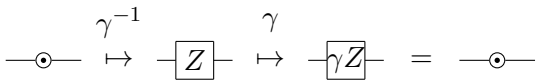


But we still have to check soundness. First $\gamma \circ \mathcal{L}_e$ and $\gamma \circ \mathfrak{c}_e$ transformed into $\gamma^{-1} \circ \gamma \circ \mathcal{L}_e$ and $\gamma^{-1} \circ \gamma \circ \mathfrak{c}_e$ which are equivalent to $\mathcal{L}_e$ and $\mathfrak{c}_e$ since $Z \circ Z = id$ in $\mathcal{L}$. $\gamma^{-1}\mathfrak{c}_e$ holds in $\mathcal{L}$, it fact those equations state that Z is a self-transposed involution. Finally the equation $Y = \gamma Z$ transforms into $Y = \gamma^{-1}\gamma Z$ which is equivalent to $Z = Z$, and this obviously holds. So γ^{-1} is a valid translation.

We already know that $\overset{\bullet}{\gamma^{-1}} \circ \overset{\bullet}{\gamma} = id$, it remains to look at $\overset{\bullet}{\gamma} \circ \overset{\bullet}{\gamma^{-1}}$. This signature map leaves $\mathcal{L}_s$ invariant. For the compact structure we have:



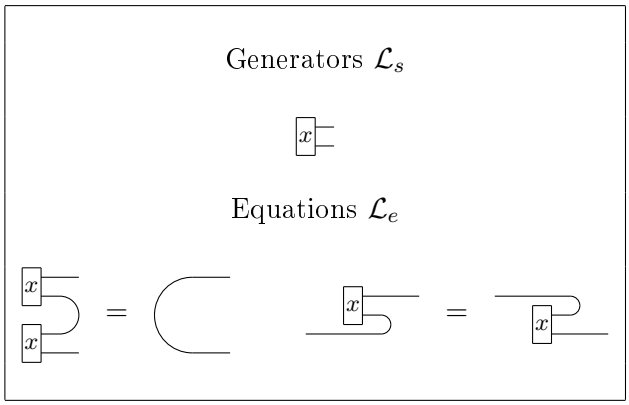
And finally for Y:



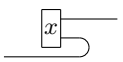
So $\overset{\bullet}{\gamma}$ is an isomorphism of props and $\mathcal{L} \simeq s_Z\mathcal{L}$. □

Here is a concrete example of cup-cap switch.

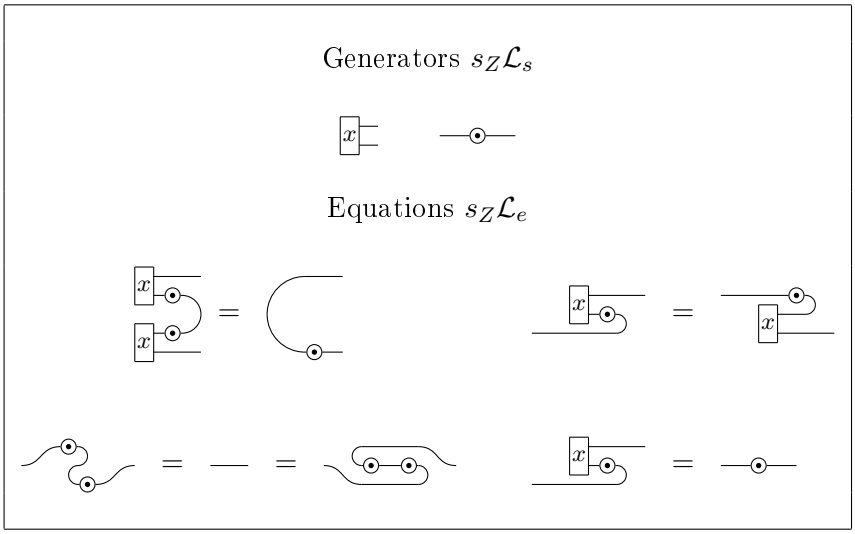
Example 10. *We consider the following compact closed graphical language $\mathcal{L}$:*



An admissible self transposed and involutive dualizer is:

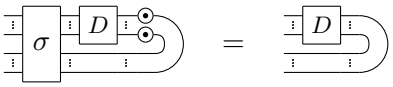
$Z \stackrel{\text{def}}{=}$ 

Then the graphical language $s_Z\mathcal{L}$ is:



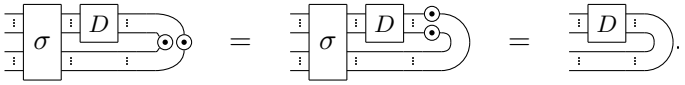
We now come back to flexsymmetry. In general, a generator that is flexsymmetric for a given symmetric compact structure will not be flexsymmetric for another. But in the case where the two compact structures are compatible we can introduce a weaker version of flexsymmetry.

Definition 41 (Flexsymmetry up to dualizer)**.** *Let $D : n \to m$ be a diagrams in a compact closed graphical language $\mathcal{L}$ and —⊙— $: 1 \to 1$ be a self transposed involution. D is said **flexsymmetric up to dualizer** if the equation:*



holds in $\mathcal{L}$ for all permutations of $n + m$ wires σ.

Notice that if the dualizer is the identity we recover the normal notion of flexsymmetry. We see that depending on which compact structure we see as canonical we can exchange between flexsymmetry and flexsymmetry up to dualizer by applying a cup-cap switch.



We see that another way to recover flexsymmetry from flexsymmetry up to dualizer would be to define a transformation that hides the necessary dualizers inside the diagrams. This is exactly the point of subdivision.

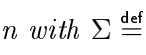
5.2.2 Subdivision

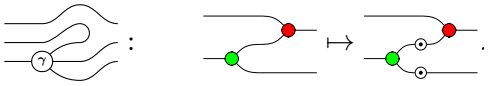
In practice, subdivision is very similar to the cup-cap switch. We replace each occurrence of some generators in the equations with the same generators composed with dualizers. The generators we want to subdivide are gathered in a signature Δ which is a sub-signature of $\mathcal{L}_s$.

Definition 42 (Subdivision of diagrams)**.** *Provided a signature Σ, a sub signature $\Delta \subseteq \Sigma$ and a diagram $Z \in F(\Sigma)[1,1]$. The subdivision is defined by the signature map:*

$$\S_{Z|\Delta} : \Sigma \to UF(\Sigma)$$
$$x \in \Sigma(n,m) \mapsto \begin{cases} Z^{\otimes m} \circ x \text{ if } x \in \Delta(n,m) \\ x \text{ if } x \notin \Delta(n,m) \end{cases}$$

The name subdivision comes from the fact that this operation corresponds to subdividing some edges of the underlying graph.

Example 11. *A Subdivision with* $\Sigma \overset{\text{def}}{=}$  $Z \overset{\text{def}}{=}$ *and* $\Delta \overset{\text{def}}{=}$ *gives:*



We now extend this to the whole graphical language. We will always subdivide by an involution.

Definition 43 (Subdivision of graphical languages)**.** *Given a graphical language $\mathcal{L}$, a sub-signature $\Delta \subset \mathcal{L}_s$, and a diagram $Z \in UF(\mathcal{L}_s)[1,1]$. The* ***subdivision*** *of $\mathcal{L}$ by Z is defined by:*

$$\S_{Z|\Delta}\mathcal{L} \overset{\text{def}}{=} (\mathcal{L}_s + \mathbb{Y}) \Big/ \left(\S_{Z|\Delta} \circ \mathcal{L}_e + \left\{Y = \S_{Z|\Delta} Z\right\} + \left\{Y^2 = id\right\}\right).$$

As cup-cap switch, subdividing a graphical language does not change its corresponding prop.

Lemma 5. *Given a graphical language $\mathcal{L}$, a subset $\Delta \subset \mathcal{L}_s$, and a diagram $Z \in F(\mathcal{L}_s)[1,1]$ such that $\mathcal{L}_\pi(Z)$ is an involution in $\overset{\bullet}{\mathcal{L}}$ we have $\mathcal{L} \simeq \S_{Z|\Delta}\mathcal{L}$.*

Proof. The proof is extremely similar to what we have done with cup-cap switch. Let $\S_{Z|\Delta}$ be defined as:

$$\S_{Z|\Delta} : \mathcal{L}_s \to UF(\mathcal{L}_s + \mathbb{Y})$$
$$x \in \mathcal{L}_s(n,m) \mapsto \begin{cases} Y^{\otimes m} \circ x \text{ if } x \in \Delta(n,m) \\ x \text{ if } x \notin \Delta(n,m) \end{cases}$$

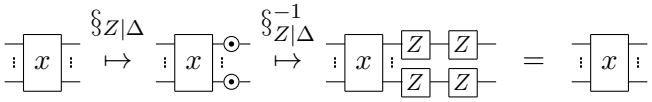
We show that $\S_{Z|\Delta}$ defines a translation $\mathcal{L} \to \S_{Z|\Delta}\mathcal{L}$ providing an isomorphism of prop $\overset{\bullet}{\S}_{Z|\Delta}$.

First we need to check the soundness condition, in other word that the equations of $\mathcal{L}$ transported by $\S_{Z|\Delta}$ still hold in $\S_{Z|\Delta}\mathcal{L}$. This holds directly since $\mathcal{L}_e$ is transformed into $\S_{Z|\Delta} \circ \mathcal{L}_e$ which are among the equations of $\S_{Z|\Delta}\mathcal{L}$. So $\S_{Z|\Delta}$ can be seen as a translation $\mathcal{L} \to \S_{Z|\Delta}\mathcal{L}$.

Now we construct an inverse defined as:

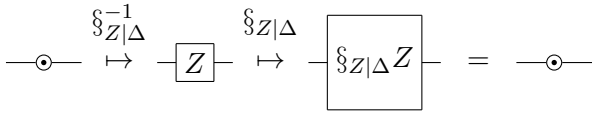
$$\S^{-1}_{Z|\Delta} : \mathcal{L}_s + \mathbb{Y} \to UF\mathcal{L}_s$$
$$x \in \mathcal{L}_s(n,m) \mapsto \begin{cases} Z^{\otimes m} \circ x \text{ if } x \in \Delta(n,m) \\ x \text{ if } x \notin \Delta(n,m) \end{cases}$$
$$Y \mapsto Z$$

We have $\S^{-1}_{Z|\Delta} \circ \S_{Z|\Delta} = id$ as translation:



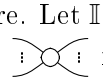
We still have to check soundness. First $\S_{Z|\Delta} \circ \mathcal{L}_e$ is transformed into $\S^{-1}_{Z|\Delta} \circ \S_{Z|\Delta} \circ \mathcal{L}_e$ which is equivalent to $\mathcal{L}_e$ since $Z \circ Z = id$ in $\mathcal{L}$. $Y \circ Y = id$ is transformed into $Z \circ Z = id$ which holds in $\mathcal{L}$. Finally the equation $Y = \S_{Z|\Delta} Z$ is transformed into $Y = \S^{-1}_{Z|\Delta} \S_{Z|\Delta} Z$ which is equivalent to $Z = Z$, and this obviously holds. So $\S^{-1}_{Z|\Delta}$ is a valid translation.

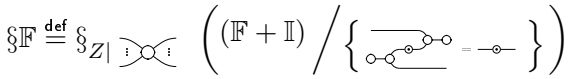
We already know that $\overset{\bullet}{\S}{}^{-1}_{Z|\Delta} \circ \overset{\bullet}{\S}_{Z|\Delta} = id$, it remains to look at $\overset{\bullet}{\S}_{Z|\Delta} \circ \overset{\bullet}{\S}{}^{-1}_{Z|\Delta}$. This signature map leaves $\mathcal{L}_s$ invariant. And for Y:



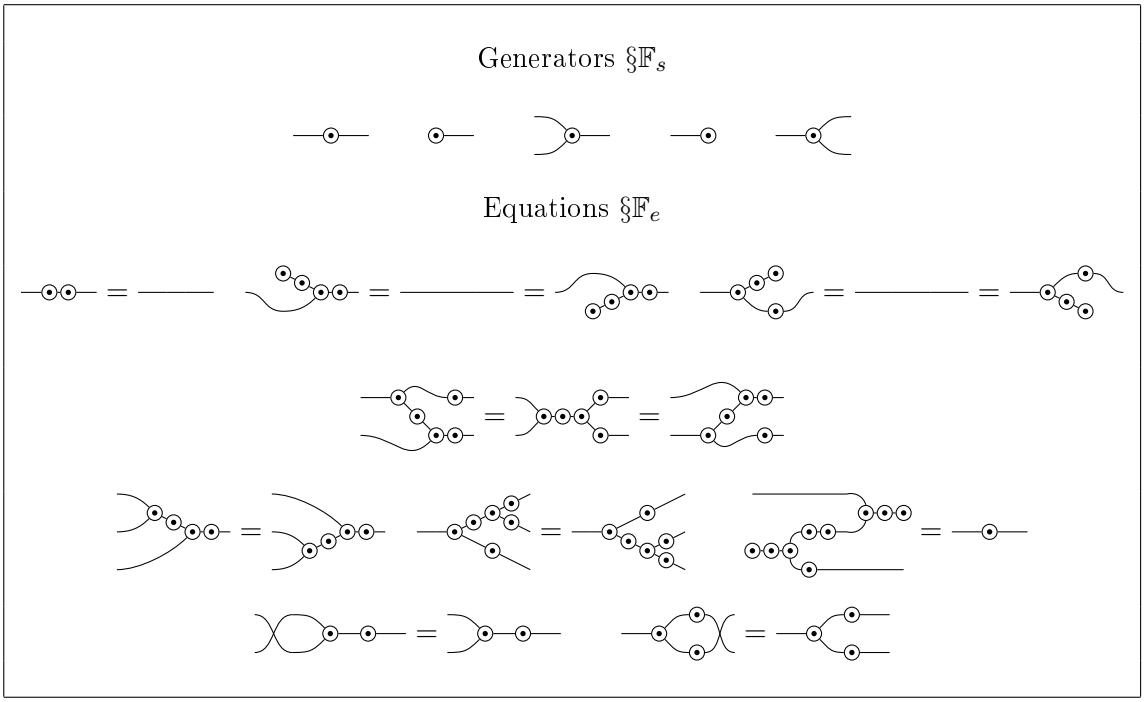
So $\overset{\bullet}{\S}_{Z|\Delta}$ is an isomorphism of props and $\mathcal{L} \simeq \S_{Z|\Delta}\mathcal{L}$. □

As promised subdivision can be used together with cup-cap switch to act on flexsymmetry in different ways, allowing to exchange flexsymmetry and flexsymmetry up to dualizers. Subdividing a generator flexsymmetric up to a dualizer by this dualizer gives a flexsymmetric generator. Subdividing a flexsymmetric generator by a dualizer gives a generator flexsymmetric up to this dualizer. A cup-cap switch with respect to a dualizer turns the flexsymmetric generator into generator flexsymmetric up to this dualizer. A generator that is flexsymmetric up to a dualizer will be flexsymmetric after a cup-cap switch with respect to this dualizer.

Now that cup-cap switch and subdivision are defined, we come back to Spiders. Subdividing a Frobenius algebra provides an interesting structure. Let $\mathbb{I} \overset{\text{def}}{=} \left(Y : 1 \to 1, \{Y^2 = id\}\right)$ and $\Delta \overset{\text{def}}{=} \mathbb{F}_s$, remember that we took the convention to write  instead of $\mathbb{F}_s$ in a graphical languages without any paradigm. A subdivided Frobenius algebra is defined by the graphical language:



The graphical language $\S\mathbb{F}$ of subdivided Frobenius algebras

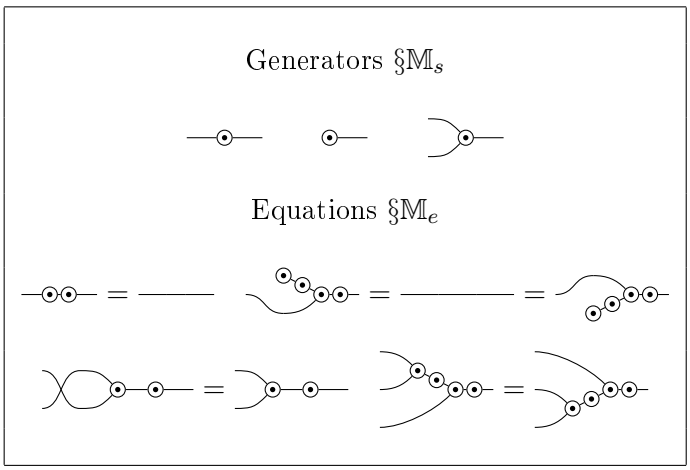


Subdivision provides a translation $\mathcal{L} \to \S_{Z|\Delta}\mathcal{L}$ that allows to translate directly some results to the subdivided version, just by subdividing everything. We call the subdivided version of the spider theorem the **harvestman theorem**.

Theorem 5 (harvestman theorem)**.** *Given a subdivided Frobenius algebra one can uniquely define the* ***harvestmen*** *with n inputs and m outputs as $h_{n,m} \stackrel{\text{def}}{=} \S_{Z|\Delta} s_{n,m}$ satisfying a fusion rule:*

Such structures already appeared in ZW-calculus [65] and ZH-calculus [66]. We also directly have a characterization in terms of flexsymmetric subdivided monoid §$\mathbb{M}$.

The graphical language §$\mathbb{M}$ of subdivided monoids



In the same way than spiders are flexsymmetric monoids, harvestmen are flexsymmetric subdivided monoids.

Lemma 6. $\overline{\S\mathbb{M}^{\mathfrak{f}}} \simeq \S\mathbb{F}$.

Proof. We have:

$$\S\mathbb{F} \simeq \S_{Z|}\ \left((\mathbb{F}+\mathbb{I})\Big/\left\{\ \right\}\right) \simeq (\mathbb{F}+\mathbb{I})\Big/\left\{\ \right\} \simeq \left(\overline{\mathbb{M}^{\mathfrak{f}}}+\mathbb{I}\right)\Big/\left\{\ \right\} \simeq \overline{(\mathbb{M}+\mathbb{I})^{\mathfrak{f}}}$$

By definition $\S\mathbb{M} \simeq \mathbb{M}+\mathbb{I}$, however this not directly tells us that $\S\mathbb{M}^{\mathfrak{f}} \simeq (\mathbb{M}+\mathbb{I})^{\mathfrak{f}}$. If we look at the flexsymmetric equations after the subdivision we see that we have flexsymmetry only up to the dualizer Z, to obtain proper flexsymmetry we then use a cup-cap switch this gives:

$$\overline{\S\mathbb{M}^{\mathfrak{f}}} \simeq s_Z\overline{\S\mathbb{M}^{\mathfrak{f}}} \simeq \overline{(\mathbb{M}+\mathbb{I})^{\mathfrak{f}}}.$$

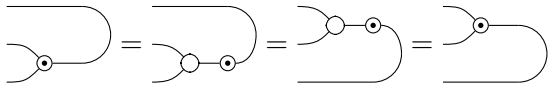
And finally $\overline{\S\mathbb{M}^{\mathfrak{f}}} \simeq \S\mathbb{F}$. □

In a flexsymmetric context, we will use the same shortcut with harvestmen as with spiders and use harvestmen and the harvestmen fusion rule to denote respectively the signature and the equations of $\S\mathbb{M}$.

The main application we will make of subdivision is the **softening** of spiders, which is defined in the following lemma.

Lemma 7 (Softening)**.** *Given a monoid flexsymmetric up to dualizer, subdividing it by this dualizer gives a flexsymmetric subdivided monoid.*

Proof. Subdivided a monoid gives a subdivided monoid by definition. We only have to show that we obtain flexsymmetry from flexsymmetry up tu dualizer. It follows from the following graphical manipulation:



□

Softening then turns spiders into harvestmen.

5.2.3 Applications

In this section, we investigate various concrete applications of cup-cap switch, subdivision, and softening.

Flexsymmetric ΔZX-calculus

The ΔZX-calculus was introduced in [56]. This language was difficult to implement in the graphical proof assistant Quantomatic [49] because of the triangle generator:

$$[\![\text{—▶—}]\!] = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

In fact Quantomatic is designed to see diagrams as graphs and then can only deal with flexsymmetric generators. With respect to the canonical compact structure of ΔZX, being flexsymmetric is equivalent to being self-transposed. This is not the case of the triangle. However it is flexsymmetric up to the NOT dualizer with interpretation: $[\![\text{—}\pi\text{—}]\!] \overset{\text{def}}{=} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Subdividing the triangle by this NOT dualizer gives a flexsymmetric generator with interpretation:

$$[\![\text{—}\pi\text{—▶—}]\!] \overset{\text{def}}{=} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Flexsymmetric graphical Abelian group algebra

It has been pointed many times that ZX-calculus is a particular case of a graphical calculus for group algebras [69, 70, 71]. Here we restrict to the group algebra over $\mathbb{C}$ for a finite Abelian. group G.

Definition 44 (Group algebra)**.** *Given a finite Abelian group $(G, \star, e)$ the group algebra $\mathbb{C}[G]$ is the $\mathbb{C}$-algebra spanned by the $|g\rangle$ for $g \in G$, in other words any element is of the form: $\sum_{g\in G} a_g |g\rangle$ with $a_g \in \mathbb{C}$. The convolution product is defined as:*

$$\left(\sum_{g\in G} a_g |g\rangle\right) * \left(\sum_{h\in G} b_h |h\rangle\right) = \sum_{h,g\in G} a_g b_h |g \star h\rangle.$$

The language is generated by two spiders. The interpretation is given into the monochromatic prop with arrows $n \to m$ the matrices in $\mathcal{M}_{|G|^m \times |G|^n}(\mathbb{C})$.

$$[\![\text{green spider } n \to m]\!] \overset{\text{def}}{=} |\overrightarrow{x}\rangle \mapsto |G|^{\frac{m+n-2}{4}} \sum_{y_j\in G} \delta_{x_1=\cdots=x_n=y_1=\cdots=y_m} |\overrightarrow{y}\rangle$$

$$[\![\text{red spider } n \to m]\!] \overset{\text{def}}{=} |\overrightarrow{x}\rangle \mapsto |G|^{\frac{2-m-n}{4}} \sum_{y_j\in G} \delta_{\bigstar_i x_i = \bigstar_j y_j} |\overrightarrow{y}\rangle$$

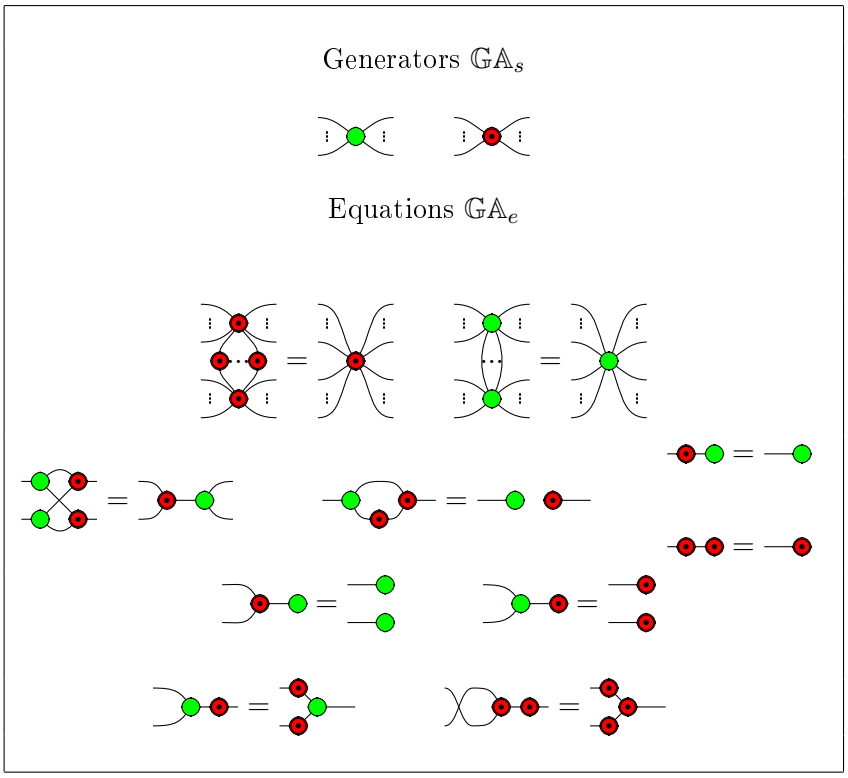
$$[\![\text{swap}]\!] = |xy\rangle \mapsto |yx\rangle \qquad [\![\text{cup}]\!] = \sum_{x\in G} |xx\rangle \qquad [\![\text{cap}]\!] = \sum_{y\in G} \langle yy|$$

The green spider is flexsymmetric with respect to the compact structure. The red spider is flexsymmetric up to the dualizer $\text{—}\bullet\text{—} : |g\rangle \mapsto |g^{-1}\rangle$. Softening the red spiders gives flexsymmetric harvestmen:

$$[\![\text{harvestmen } n \to m]\!] \overset{\text{def}}{=} |\overrightarrow{x}\rangle \mapsto |G|^{\frac{2-m-n}{4}} \sum_{y_j\in G} \delta_{\left(\bigstar_i x_i\right)\star\left(\bigstar_j y_j\right)=e} |\overrightarrow{y}\rangle$$

leading to a flexsymmetric language. Here, we do not claim to be complete.

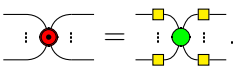
The flexsymmetric graphical language $\mathbb{GA}$ of Abelian group algebras



If $G = \mathbb{Z}/d\mathbb{Z}$ we can add an additional generator $1 \to 1$ with interpretation the Fourier transform:

$$[\![\text{—□—}]\!] \stackrel{\text{def}}{=} |x\rangle \mapsto \frac{1}{\sqrt{d}} \sum_{j=0}^{d-1} e^{\frac{2ij\pi}{d}} |j\rangle$$

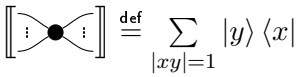
and satisfying:



The qudit ZX-calculus [70] is a special case of the group algebra construction with $G = \frac{\mathbb{Z}}{d\mathbb{Z}}$. In the case of qubits, where $d = 2$, the dualizer is the identity and then we recover the qubit ZX-calculus which is directly flexsymmetric. However, in the case $d \geq 3$, harvestmen are necessary. This gives an equivalent flexsymmetric presentation of the qutrit ZX-calculus [72]. In [72] some topological lemmas are shown which would directly follow from flexsymmetry. Furthermore, this presentation allows us to avoid the inverse Fourier transform as an explicit generator.

The black cap ZW-calculus (black fragment)

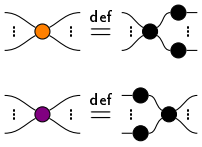
One of the starting points of this work was the odd look of equations of ZW-calculus. They are subdivided versions of more familiar equations. To see this we will only consider the black harvestman fragment of the ZW-calculus. The black harvestman has for interpretation:



$$\stackrel{\text{def}}{=} \sum_{|xy|=1} |y\rangle \langle x|$$

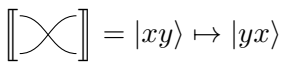
Where $|xy|$ is the Hamming weight (the number of 1) in the binary word xy, the concatenation of the binary words x and y.

We can see this harvestman as arising from the subdivision of two spiders, the indigo and the orange, which are the transpose of each other:

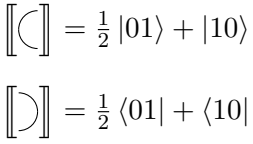


$$= \sum_{|xy|=1} |y\rangle \langle \overline{x}|$$

$$= \sum_{|xy|=1} |\overline{y}\rangle \langle x|$$



$$= |xy\rangle \mapsto |yx\rangle$$

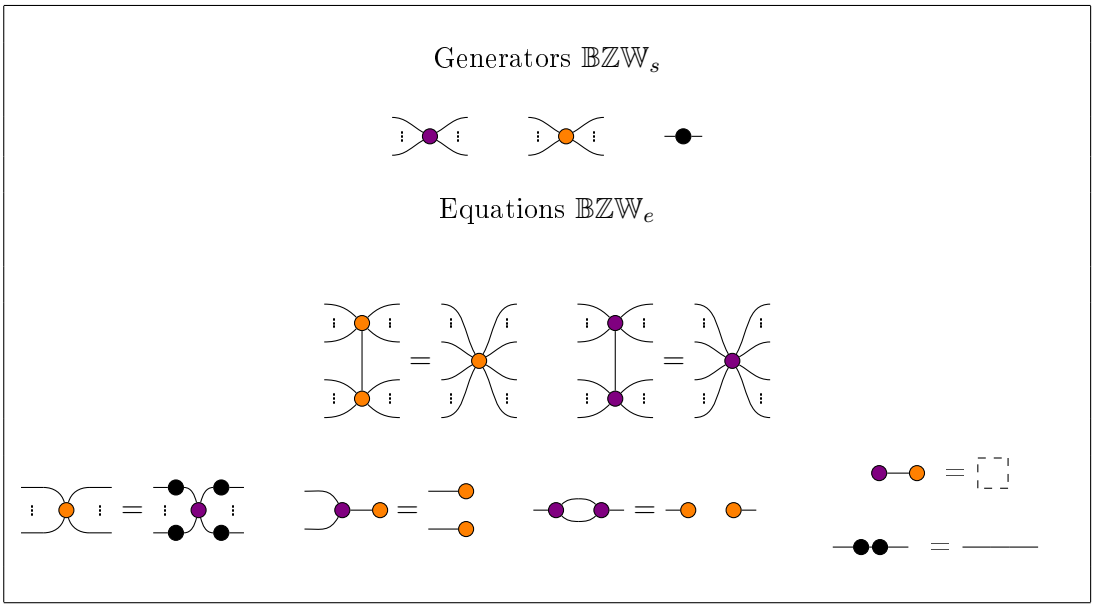


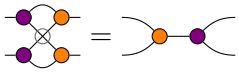
$$= \tfrac{1}{2} |01\rangle + |10\rangle$$

$$= \tfrac{1}{2} \langle 01| + \langle 10|$$

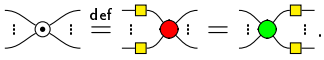
Those two spiders are both flexsymmetric up to the dualizer $= |x\rangle \mapsto |\overline{x}\rangle$ with respect to the canonical compact structure of the ZW-calculus. Thus, a cup-cap switch gives two flexible spiders satisfying equations close to those of the ZX-calculus given in Chapter 3.

The flexsymmetric graphical language $\mathbb{BZW}$ of the black fragment of the ZW-calculus.



Those spiders are not special but anti-special. More details about anti-special spiders can be found in [17]. The generators satisfy the equation  involving the fermionic swap with interpretation:  $\stackrel{\text{def}}{=} |xy\rangle \mapsto (-1)^{x \wedge y} |yx\rangle$. However the fermionic swap is not flexsymmetric with respect any of the two compact structures considered.

The monochromatic ZX-calculus

The ZX-calculus corresponds to $\mathbb{GA}$ with $G = \mathbb{Z}/2\mathbb{Z}$ and the Hadamard gate denoting the Fourier transform. As said before this language is already flexsymmetric, however using cup-cap switch and softening gives an interesting equivalent language. Applying a cup-cap switch with gives two monoids flexsymmetric up to . If we soften the green spider we obtain a flexsymmetric subdivided monoid that surprisingly can also be expressed in function of the red spider:  . We then obtain a new language with interpretation:

$$\left[\!\!\left[\right]\!\!\right] = 2^{\frac{n+m-2}{4}} |+\rangle^{\otimes m} \langle 0|^{\otimes n} + e^{i\alpha} |-\rangle^{\otimes m} \langle 1|^{\otimes n}$$

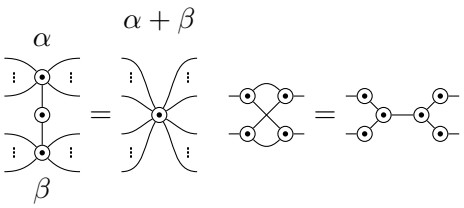
$$\left[\!\!\left[\right]\!\!\right] = |xy\rangle \mapsto |yx\rangle \qquad \left[\!\!\left[\right]\!\!\right] = |0+\rangle + |1-\rangle \qquad \left[\!\!\left[\right]\!\!\right] = \langle 0+| + \langle 1-|$$

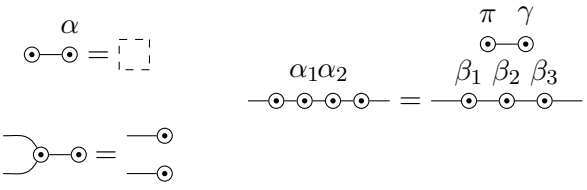
where $|+\rangle \overset{\text{def}}{=} \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle \overset{\text{def}}{=} \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Translating the axioms of [54] gives a complete calculus for qubits relying on only one harvestman.

The flexsymmetric graphical language $\mathbb{M}$-ZX of monochromatic ZX-calculus

Generators $\mathbb{M}$-ZX$_s$

Equations $\mathbb{M}$-ZX$_e$





$\beta_1 = \arg(z) + \arg(z')$
$\beta_2 = 2\arg(i + |\frac{z}{z'}|)$
$\beta_3 = \arg(z) - \arg(z')$
$\gamma = x^+ - \arg(z) + \frac{\pi-\beta_2}{2}$

$x^+ \overset{\text{def}}{=} \frac{\alpha_1+\alpha_2}{2}$
$x^- \overset{\text{def}}{=} x^+ - \alpha_2$
$z \overset{\text{def}}{=} -\sin(x^+) + i\cos(x^-)$
$z' \overset{\text{def}}{=} \cos(x^+) - i\sin(x^-)$
$z' = 0 \Rightarrow \beta_2 = 0$

5.3 Signature graphs

We now try to make precise the correspondence between flexsymmetric languages and graphs by giving an explicit construction of $F^{\dagger}\Sigma$. This provides a possible implementation of flexsymmetric graphical languages using what we call signature graphs.

5.3.1 Definition

Informally, given a monochromatic signature Σ, a **signature graph** over Σ is an open graph, with inputs and outputs, whose internal vertices correspond to generators in $|\Sigma|$. Furthermore, those internal vertices must have a degree equal to the total arity, number of inputs plus number of outputs of the corresponding generator.

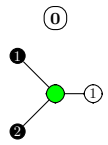
Definition 45 (Signature graphs)**.** *Given a signature Σ, a **signature graph** is a tuple (G, i, o, l). G is a (vertex) coloured multi graph with set of colours $|\Sigma| \uplus \{I, O\}$. We call **input vertices**, respectively **output vertices**, and denote $In(G)$, respectively $Out(G)$, the set of vertices coloured with I, respectively O. The other vertices are called **generator vertices**. G must satisfy:*

- *A vertex in $In(G) \bigcup Out(G)$ has degree one.*
- *A generator vertex, coloured in $x \in \Sigma(n, m)$, has degree $n + m$.*

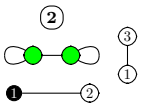
*$i : [\![1, |In(G)|]\!] \to In(G)$ and $o : [\![1, |Out(G)|]\!] \to Out(G)$ are bijective labelling functions. $l \in \mathbb{N}$ is a natural number called the **number of loops**.*

Each generator vertex corresponds to a generator in the signature. The inputs and outputs vertices correspond respectively to the inputs and outputs of diagrams. For a signature graph G the notation $G : n \to m$ means that $n = |In(G)|$ and $m = |Out(G)|$. A signature graph is represented graphically as G together with an additional node indicating its number of loops l.

Example 12. *As an example, we represent the signature graph corresponding to a generator $2 \to 1$ as:*



A more complicated example of a signature graph based on the same generator:



Definition 46 (Signature graph isomorphism)**.** *Two signature graphs are said **isomorphic** if there is a graph isomorphism preserving the colours and the labeling of inputs and outputs between them, and they have the same loop count.*

In practice, we will always consider two isomorphic signature graphs to be the same. Our definition of signature graph match closely the definition of labeled graphs in [67]. But here we will extend this definition to form a prop.

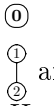
5.3.2 The category of signature graphs

We define the composition of two signature graphs intuitively by plugging outputs into inputs.

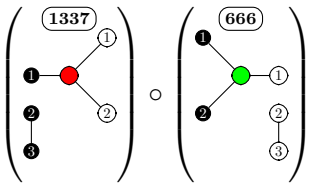
Definition 47 (Composition of signature graphs)**.** *The composition $H \circ G$ of a signature graph $G : a \to b$ with a signature graph $H : b \to c$ is constructed as follow:*

- *Take the disjoint union of the two graphs $G + H$, we call **interface** vertices the vertices in $In(H) \bigcup Out(G)$.*
- *For all $1 \leq j \leq b$ add an edge $(i_H(j), o_G(j))$. Now each interface vertex has degree 2.*
- *Remove all cycles of interface vertices, let k be the number of cycles removed.*
- *Replace all chain of interface vertices between non-interface vertices x and y by an edge (x, y).*

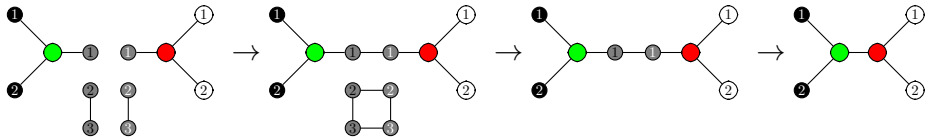
Then $In(H \circ G) \stackrel{\text{def}}{=} In(G)$, $Out(H \circ G) \stackrel{\text{def}}{=} Out(H)$, $i_{H \circ G} \stackrel{\text{def}}{=} i_G$, $o_{H \circ G} \stackrel{\text{def}}{=} o_H$ and $l_{H \circ G} = l_G + l_H + k$.

In the construction, the only cycles that can occur are those arising from the composition of  and  . This creates a loop that is taken into account by incrementing the number of loops of $H \circ G$.

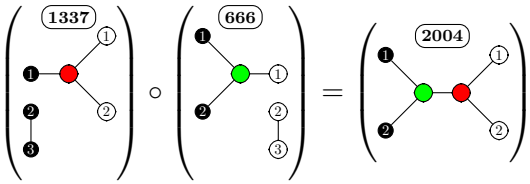
Example 13. *We compute:*



We start by the union graph, colouring the interface vertices in grey:



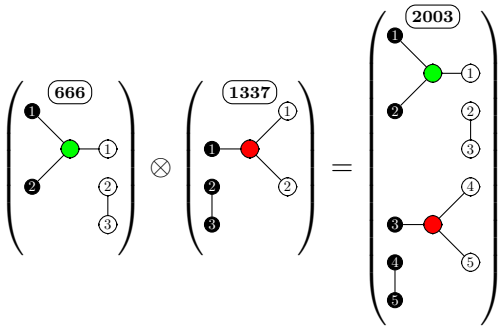
We had to remove one cycle so here $k = 1$. Finally:

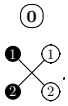


We define a tensor of signature graphs by taking the union graph and relabelling the inputs and outputs.

Definition 48 (Tensor of signature graphs)**.** *The tensor $H \otimes G$ of a signature graph $G : a \to b$ with a signature graph $H : c \to d$ is the disjoint union of the two graphs $G + H$. We set $In(G \otimes H) \stackrel{\text{def}}{=} In(G) \bigcup In(H)$, $Out(G \otimes H) \stackrel{\text{def}}{=} Out(G) \bigcup Out(H)$, $l_{G \otimes H} = l_G + l_H$, $i_{G \otimes H}(j) \stackrel{\text{def}}{=} i_G(j)$ if $j \leq a$ and else $i_H(j - a)$, $o_{G \otimes H}(j) \stackrel{\text{def}}{=} o_G(j)$ if $j \leq b$ and else $o_H(j - b)$.*

Example 14. *The only subtlety here is the relabelling:*

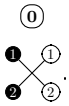


Proposition 6. *The category **Sig-gr**$_\Sigma$ with objects the natural numbers and morphisms the signature graphs over Σ up to signature graph isomorphism is a monochromatic compact closed prop. The identities are given by the signature graphs:*  *. The cup and cap are respectively*  *and*  *. The swap is*  *.*

Proof. When composing three signature graphs the two interfaces have no vertex in common this ensures the associativity of composition. The identities are given by the signature graphs of the form:  . So **Sig-gr**$_\Sigma$ is a category.

The tensor is associative and the tensor unit identity is the empty signature graphs: ⓪.

The functoriality follows from the fact that when taking the tensor product of two compositions, we can take the disjoint union of the interfaces and see it as the interface for the composition of the two tensors.

The symmetries are generated by the involutive swap signature graphs  . The naturality follows from signature graph isomorphisms.

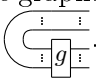
So **Sig-gr**$_\Sigma$ is a prop.

Finally,  and  form a compact structure. □

5.3.3 Free flexsymmetric props

The link between signature graphs and the flexsymmetry paradigm is given by the following theorem:

Theorem 6. ***Sig-gr***$_\Sigma \simeq F^\dagger\Sigma$

Proof. Given a morphism $f : n \to m$ in $F^{\dagger}\Sigma$ we will show how to interpret it as a signature graph. Using the compact structure each generator $g : a \to b$ corresponds to a state $\overline{g} : 0 \to a+b$: 

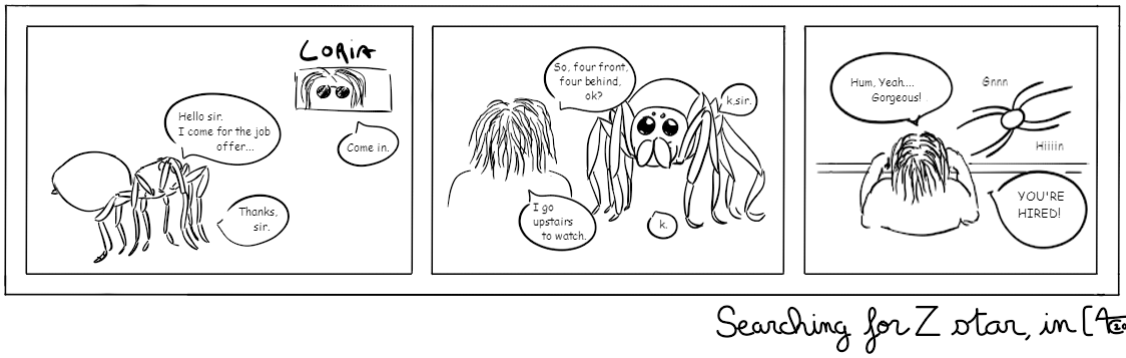
. The flexsymmetry equations exactly mean that those states are symmetric (we can permute the outputs) for all $g \in \Sigma$. So in a diagram representing f, we can see the generators as big vertices. Thus f is completely characterized by giving a list of vertex generators and how there are linked with each other and with the n inputs and m outputs. In other words, f is uniquely defined by a Σ-graph. Conversely, all interpretations of a signature graph as a string diagram are equivalent modulo the flexsymmetric equations.

This provides a bijective mapping between $F^{\dagger}\Sigma[n, m]$ and $\mathbf{Sig\text{-}gr}_{\Sigma}[n, m]$. This mapping corresponds to a full and faithful prop morphism $F^{\dagger}\Sigma \to \mathbf{Sig\text{-}gr}_{\Sigma}$ hence an isomorphism of prop. So $F^{\dagger}\Sigma \simeq \mathbf{Sig\text{-}gr}_{\Sigma}$. □

A flexsymmetric graphical language can be soundly represented by a signature graph quotiented by equations. The flexsymmetric paradigm then corresponds to only topology matter in the following sense: diagrams are basically graphs. Seeing string diagrams as formal combinatorial objects is not a new idea, they were first described topologically in [42]. An approach more similar to mine are the string graphs of [73] which can be seen as more complex signature graphs that can characterize $F\Sigma$, the free prop over Σ. Any graphical language can be represented as string graphs quotiented by equations. The signature graphs are simpler than string graphs, the key here is the symmetric compact structure and the flexsymmetry equations that allow to quotient and drop many subtleties in the description of string diagrams.

Chapter 6

Interacting monoids



Numerous graphical calculi with interpretations in **Lin** have been defined in the last decades. The ZX-calculus [64] relies on the interaction of the two mutually unbiased bases Z and X. Then the ZW-calculus [65] was built on two tripartite entanglement classes, the GHZ, and W-states. Finally, the ZH-calculus [66] has been introduced to represent easily hyper-graph states. At some point, it became unsure if the future of categorical quantum mechanics will see the multiplication of languages or if one would become hegemonic. In the latter case, the duel was mostly between ZX and ZH. In fact, if ZW-calculus was instrumental in the achievements of completeness, it never really spread out of a small community despite some applications to fermionic quantum computing [74]. Concerning the multiplication of languages we discovered that in a sense, everything interesting was already on the table. This has been done in [5] and this chapter is built on this work. ZX, ZW, and ZH are very similar and share a common core structure. The study of harvestmen and flexsymmetry presented in Chapter 5 gives insights on this common structure and allows us to find all the possible graphical languages on qubits sharing this structure. In fact, the space of qubits is a very small space, and then not too many structures are available. The only non-trivial candidates are essentially ZX, ZW, and ZH.

There exist some other formalisms trying to unify graphical languages, in particular in the context of interacting Frobenius algebras [21] or Hopf-Frobenius algebras [75]. However, these formalisms usually require too much structure and fail to capture all three examples simultaneously. Typically they do not capture the ZW-calculus.

6.1 Definition

We start by defining the common structure shared by the quantum graphical languages.

6.1.1 Monoids

The graphical language $\mathbb{M}$ of monoids has been introduced in Chapter 1. A model of this language in **Lin** corresponds exactly to a two-dimensional complex unital commutative algebra. An easy way to build some example are monoids algebras.

Example 15 (monoid algebra [76])**.** *Given a monoid $M = (X, *, e)$ with d elements, we can define a d-dimensional unital algebra $\mathbb{C}[M]$ by indexing each element of a basis by the elements of M.*

$$[\![\bigcirc\!\!-]\!] = |e\rangle \qquad \left[\!\!\left[\right.\right. \!\!\bigcirc\!\!- \!\!\left.\right]\!\!\right] = |i\rangle\,|j\rangle \mapsto |i * j\rangle$$

If M is a group, we will speak of a group algebra*.*

Starting from a monoid M of cardinality $d+1$ that contains a zero element (that we note $\perp$), we can build a contracted algebra $\mathbb{K}M$ *by essentially the same construction, but identifying the element $\perp$ with the matrix 0. An example of this construction is the* ***co-copy****. Considering the monoid defined by $i * j = i$ if $i = j$ and $\perp$ otherwise. We obtain*

$$[\![\bigcirc\!\!-]\!] = \sum_i |i\rangle \qquad \left[\!\!\left[\right.\right. \!\!\bigcirc\!\!- \!\!\left.\right]\!\!\right] = |i\rangle\,|j\rangle \mapsto \delta_{i,j}\,|i\rangle$$

We will be mainly interested in four examples that we call Z, X, H and W. Working in the basis $(|0\rangle, |1\rangle)$. They correspond to contracted algebras $\mathbb{C}M$.

Z	$\|0\rangle$	$\|1\rangle$
$\|0\rangle$	$\|0\rangle$	0
$\|1\rangle$	0	$\|1\rangle$

X	$\|0\rangle$	$\|1\rangle$
$\|0\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

H	$\|0\rangle$	$\|1\rangle$
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$

W	$\|0\rangle$	$\|1\rangle$
$\|0\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	0

Those multiplication tables describe the behaviour of the algebras on $|0\rangle$ and $|1\rangle$. We see that Z behaves like a Kronecker delta ensuring equality, X is the XOR gate, H the AND gate and W is the effect algebra on two elements.

The corresponding matrix representations in the computational basis are:

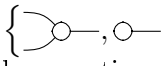
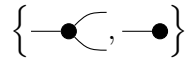
Z	X	H	W
$\begin{pmatrix}1&0&0&0\\0&0&0&1\end{pmatrix}$ $\begin{pmatrix}1\\1\end{pmatrix}$	$\begin{pmatrix}1&0&0&1\\0&1&1&0\end{pmatrix}$ $\begin{pmatrix}1\\0\end{pmatrix}$	$\begin{pmatrix}1&1&1&0\\0&0&0&1\end{pmatrix}$ $\begin{pmatrix}0\\1\end{pmatrix}$	$\begin{pmatrix}1&0&0&0\\0&1&1&0\end{pmatrix}$ $\begin{pmatrix}1\\0\end{pmatrix}$

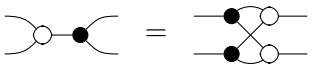
A co-monoid is a model of the graphical language $\mathbb{M}^{op}$ of co-monoids mentioned in Chapter 3. In **Lin** all co-monoids are obtained by transposing a monoid. We defined four examples called Z, X, H and W:

Z	X	H	W
$\begin{pmatrix}1&0\\0&0\\0&0\\0&1\end{pmatrix}$ $(1\ \ 1)$	$\frac{1}{2}\begin{pmatrix}1&0\\0&1\\0&1\\1&0\end{pmatrix}$ $(2\ \ 0)$	$\begin{pmatrix}1&2\\0&-1\\0&-1\\0&1\end{pmatrix}$ $(1\ \ 2)$	$\begin{pmatrix}0&0\\1&0\\1&0\\0&1\end{pmatrix}$ $(0\ \ 1)$

Those co-monoids respectively form Frobenius algebras with the monoid denoted by the same letter. Note that only Z is the transposed of Z.

6.1.2 Bi-algebra rule

Given a monoid  and a co-monoid  we say that they form a **bi-algebra pair** if the equation:

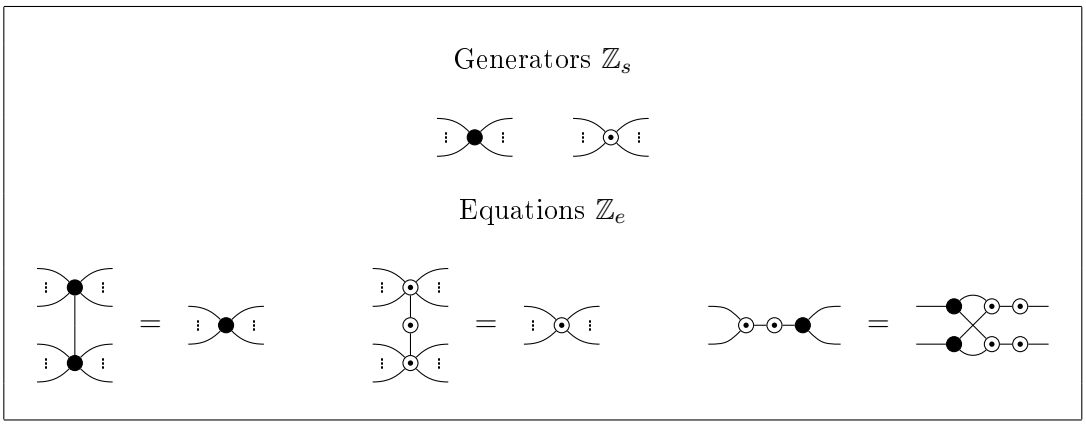


holds in **Lin**. A large family of examples is given by the interaction of the copy co-monoid with any monoid algebra. We denote bi-algebra pairs AB where A is a co-monoid and B is a monoid. Thus, examples of bi-algebra pairs are ZZ, ZX, ZW, and ZH.

6.1.3 Z^*-algebra

We now focus on bi-algebra pairs in a flexsymmetric context. Our goal is to obtain a flexsymmetric graphical language. Given a bi-algebra pair, we can find a compact structure making the monoid flexsymmetric and another one making the co-monoid flexsymmetric. However, nothing tells us that those compact structures are the same. But if they are compatible we can still use softening of Chapter 5 to obtain a flexsymmetric graphical language. This is the definition of a **Z^*-algebra**: a bi-algebra pair together with two compatible compact structures, one making the monoid flexsymmetric and the other making the co-monoid flexsymmetric. By convention, we will choose as canonical compact structure the one corresponding to the co-monoid. This allows a shorter definition:

The flexsymmetric graphical language $\mathbb{Z}$



A Z^*-algebra is then a model of $\mathcal{Z}$ in **Lin**. We can find some constructions related to Z^*-algebras in [77] and [78].

6.2 Classifications

We now look for all possible monoids and bi-algebra pairs in **Lin**.

6.2.1 Monoids

The classification of all unital commutative algebras of dimension two has been known for a long time [79]: there are only two algebras up to isomorphism.

Theorem 7 ([79]). *In* **Lin**, *any monoid is isomorphic either to Z or to W.*

Proof. We are looking for all unital algebras up to isomorphism in $\mathbb{C}^2$.

Given an algebra with unit, we choose a basis $(|0\rangle, |1\rangle)$ where $|0\rangle$ is the unit. Then the matrix representation of the monoid is $\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 1 & y \end{pmatrix}$. The change of basis $\begin{pmatrix} 1 & \frac{y}{2} \\ 0 & 1 \end{pmatrix}$ gives $\begin{pmatrix} 1 & 0 & 0 & \lambda \\ 0 & 1 & 1 & 0 \end{pmatrix}$ with $\lambda \coloneqq x + \frac{y^2}{2}$. Let $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ be an invertible matrix. Its determinant is $\Delta \coloneqq ad - bc \neq 0$. We want:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & \lambda \\ 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}^{\otimes 2} = \Delta^2 \begin{pmatrix} 1 & 0 & 0 & \mu \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

this gives the following system:

$$\begin{cases} \Delta = ad - bc \neq 0 \\ \Delta^2 = ad^2 - 2bcd + \lambda ac^2 \\ 0 = c\left(b^2 - \lambda a^2\right) \\ \Delta^2 \mu = \lambda a^3 - ab^2 \\ 0 = c\left(\lambda c^2 - d^2\right) \\ \Delta^2 = ad^2 - \lambda ac^2 \\ 0 = b^2 c - 2abd + \lambda a^2 c \end{cases}$$

If $c \neq 0$ then we have $d^2 = \lambda c^2$ and then $\Delta^2 = 0$, a contradiction. Setting $c = 0$ the system reduces to:

$$\begin{cases} \Delta = ad \neq 0 \\ \Delta^2 = ad^2 \\ \Delta^2 \mu = \lambda a^3 - ab^2 \\ 0 = -2abd \\ c = 0 \end{cases} \Rightarrow \begin{cases} \Delta = ad \neq 0 \\ \Delta^2 = ad^2 \\ \Delta^2 \mu = \lambda a^3 \\ b = 0 \\ c = 0 \end{cases} \Rightarrow \begin{cases} d \neq 0 \\ \mu = \frac{\lambda}{d^2} \\ a = 1 \\ b = 0 \\ c = 0 \end{cases}$$

Finally we have $a = 1$, $b = 0$, $c = 0$ and $d \neq 0$. The equivalence classes correspond to the elements of $\mathbb{C}$ up to multiplication by non-zero squares. In $\mathbb{C}$ there are only two $\lambda = 0$ and $\lambda \neq 0$. The case $\lambda \neq 0$ admit a very simple representative: the change of basis $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$ gives $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$. So we obtain two representatives Z and W.

□

Note that this also provides a classification of all co-monoids up to isomorphisms. The monoids Z, X, and H are isomorphic and by duality, the co-monoids Z, X, and H are also isomorphic.

6.2.2 Bi-algebra pairs

The classification of bi-algebra pair is also already known, we find a classification of low dimension bi-algebras in [80]. For the sake of completeness, we will also prove it here in our notations.

To simplify our classification up to isomorphism, we start by identifying all the algebra automorphisms in **Lin**.

Lemma 8. *The unique non-trivial automorphisms of Z and W are respectively* $\begin{pmatrix}0&1\\1&0\end{pmatrix}$ *and the matrices of the form* $\begin{pmatrix}1&0\\0&a\end{pmatrix}$ *with* $a \in \mathbb{C}^*$.

Proof. We start with Z:

$$\begin{pmatrix}a&b\\c&d\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}d&-b\\-c&a\end{pmatrix}^{\otimes 2} = \Delta^2\begin{pmatrix}1&0&0&0\\0&0&0&1\end{pmatrix}$$

this gives the following system:

$$\begin{cases}\Delta = ad - bc \neq 0\\ \Delta^2 = ad^2 + bc^2\\ 0 = -ab\,(c+d)\\ 0 = ab\,(a+b)\\ 0 = cd\,(c+d)\\ 0 = -cbd - dac\\ \Delta^2 = cb^2 + da^2\end{cases}$$

If $a = 0$ then:

$$\begin{cases}\Delta = bc \neq 0\\ \Delta^2 = bc^2\\ 0 = cd\,(c+d)\\ 0 = -cbd\\ \Delta^2 = cb^2\end{cases} \Rightarrow \begin{cases}\Delta = bc \neq 0\\ \Delta^2 = bc^2\\ \Delta^2 = cb^2\\ d = 0\end{cases} \Rightarrow \begin{cases}d = 0\\ b = 1\\ c = 1\end{cases}$$

the solution is $\begin{pmatrix}0&1\\1&0\end{pmatrix}$. If $a \neq 0$ and $b \neq 0$ we then have $\Delta = 0$, a contradiction. If $a \neq 0$ and $b = 0$:

$$\begin{cases}\Delta = ad \neq 0\\ \Delta^2 = ad^2\\ 0 = cd\,(c+d)\\ 0 = -dac\\ \Delta^2 = da^2\\ b = 0\end{cases} \Rightarrow \begin{cases}\Delta = ad \neq 0\\ \Delta^2 = ad^2\\ \Delta^2 = da^2\\ c = 0\\ b = 0\end{cases} \Rightarrow \begin{cases}a = d = 1\\ c = 0\\ b = 0\end{cases}$$

the solution is $\begin{pmatrix}1&0\\0&1\end{pmatrix}$. Now for W:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}^{\otimes 2} = \Delta^2 \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

this gives the system:

$$\begin{cases} \Delta = ad - bc \neq 0 \\ \Delta^2 = ad^2 - 2bcd \\ 0 = b^2 c \\ 0 = ab^2 \\ 0 = cd^2 \\ \Delta^2 = ad^2 \\ 0 = cb^2 - abd \end{cases} \Rightarrow \begin{cases} \Delta = ad \neq 0 \\ \Delta^2 = ad^2 \\ b = c = 0 \end{cases} \Rightarrow \begin{cases} d \neq 0 \\ a = 1 \\ b = c = 0 \end{cases}$$

the solutions are the matrices $\begin{pmatrix} 1 & 0 \\ 0 & d \end{pmatrix}$ with $d \neq 0$.

□

This result allows us to find all the bi-algebra pairs up to isomorphisms.

Lemma 9. *In **Lin**, up to isomorphism, the only bi-algebra pairs are ZZ, ZX, ZW, ZH, and the transpose of ZW.*

Proof. There are only two co-monoids up to isomorphism, the transpose of Z and W. Note that $Z^t = Z$.

Any monoid is of the form: $\begin{pmatrix} a & b & b & c \\ d & e & e & f \end{pmatrix}$.

We start by finding all the monoids satisfying the bi-algebra rule with W^t.

We want:

$$W^t \circ \begin{pmatrix} a & b & b & c \\ d & e & e & f \end{pmatrix} = \begin{pmatrix} a & b & b & c \\ d & e & e & f \end{pmatrix}^{\otimes 2} \left[I_2 \otimes \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \otimes I_2 \right] W^{\otimes 2}$$

This gives the following system:

$$\begin{cases} 0 = a\,(a-1) \\ 0 = d\,(a-1) \\ 0 = d^2 \\ 0 = b\,(2a-1) \\ 0 = e\,(a-1) + dc \\ 0 = de \\ 0 = c\,(2a-1) + 2b^2 \\ 0 = f\,(a-1) + 2be + cd \\ 0 = 2df + 2e^2 \end{cases} \Rightarrow \begin{cases} 0 = a\,(a-1) \\ d = 0 \\ 0 = b\,(2a-1) \\ 0 = c\,(2a-1) + 2b^2 \\ 0 = f\,(a-1) + 2bc \\ e = 0 \end{cases} \Rightarrow \begin{cases} \text{if } a = 0\text{:} \begin{cases} a = 0 & b = 0 \\ c = 0 & d = 0 \\ e = 0 & f = 0 \end{cases} \\ \text{if } a \neq 0\text{:} \begin{cases} a = 1 & b = 0 \\ c = 0 & d = 0 \\ e = 0 & f \in \mathbb{C} \end{cases} \end{cases}$$

The only rank 2 solution are the $\begin{pmatrix}1&0&0&0\\0&0&0&f\end{pmatrix}$ with $f \in \mathbb{C}^*$. They are monoids with units $\begin{pmatrix}1\\\frac{1}{f}\end{pmatrix}$. Since $\begin{pmatrix}1&0\\0&f\end{pmatrix}$ is an automorphism of W^t this gives a unique pair up to isomorphism: W^tZ.

Now with Z, we want:

$$Z \circ \begin{pmatrix}a&b&b&c\\d&e&e&f\end{pmatrix} = \begin{pmatrix}a&b&b&c\\d&e&e&f\end{pmatrix}^{\otimes 2} \left[I_2 \otimes \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{pmatrix} \otimes I_2 \right] {\Delta_Z}^{\otimes 2}$$

This gives the following system:

$$\begin{cases} a = a^2 & 0 = ad & d = d^2 \\ b = b^2 & 0 = be & e = e^2 \\ c = c^2 & 0 = cf & f = f^2 \end{cases} \quad \Leftrightarrow \quad \begin{cases} a,b,c,d,e,f \in \{0,1\} \\ (a \neq 1) \vee (d \neq 1) \\ (b \neq 1) \vee (e \neq 1) \\ (c \neq 1) \vee (f \neq 1) \end{cases}$$

The rank 2 solutions are:

$$\begin{pmatrix}1&0&0&0\\0&0&0&1\end{pmatrix} \begin{pmatrix}1&0&0&1\\0&1&1&0\end{pmatrix} \begin{pmatrix}0&1&1&0\\1&0&0&1\end{pmatrix} \begin{pmatrix}1&1&1&0\\0&0&0&1\end{pmatrix}$$
$$\begin{pmatrix}1&0&0&0\\0&1&1&1\end{pmatrix} \begin{pmatrix}1&0&0&0\\0&1&1&0\end{pmatrix} \begin{pmatrix}0&1&1&0\\0&0&0&1\end{pmatrix} \begin{pmatrix}0&0&0&1\\1&0&0&0\end{pmatrix}$$
$$\begin{pmatrix}0&1&1&1\\1&0&0&0\end{pmatrix} \begin{pmatrix}0&0&0&1\\1&1&1&0\end{pmatrix} \begin{pmatrix}0&0&0&1\\0&1&1&0\end{pmatrix} \begin{pmatrix}0&1&1&0\\1&0&0&0\end{pmatrix}$$

Since $\begin{pmatrix}0&1\\1&0\end{pmatrix}$ is an automorphism of Δ_Z, this reduces the possibilities to:

$$\begin{pmatrix}1&0&0&0\\0&0&0&1\end{pmatrix} \begin{pmatrix}1&0&0&1\\0&1&1&0\end{pmatrix} \begin{pmatrix}1&1&1&0\\0&0&0&1\end{pmatrix} \begin{pmatrix}1&0&0&0\\0&1&1&0\end{pmatrix}$$
$$\begin{pmatrix}0&0&0&1\\1&0&0&0\end{pmatrix} \begin{pmatrix}0&0&0&1\\1&1&1&0\end{pmatrix} \begin{pmatrix}0&1&1&0\\1&0&0&0\end{pmatrix}$$

But among them the last three are not algebras, they are not associative, a counter example for the three maps is the evaluation of $(|0\rangle * |0\rangle) * |1\rangle$ versus $|0\rangle * (|0\rangle * |1\rangle)$. The other are the algebras Z, X, H and W.

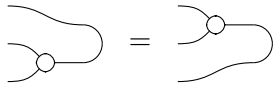
This gives 4 pairs, ZZ, ZX, ZW and ZH. □

From now on we will not take into account the transpose of ZW since we can recover all results involving it by transposing what we find with ZW.

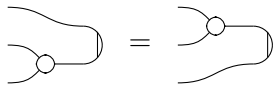
6.2.3 Frobenius algebras

We now characterize all Frobenius algebras in **Lin**. We know from Chapter 5 that this just amounts to find all the compact structures making the monoids flexsymmetric. To find those compact structures we use the following lemma.

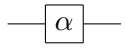
Lemma 10. *Given a monoid*  *and a compact structure* *satisfying:*

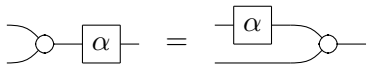


then any other compact structure  *satisfying:*



is of the form:

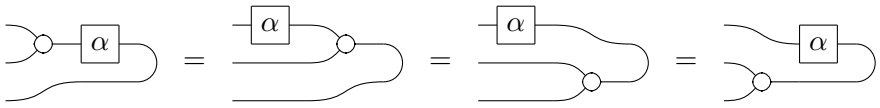
where  *is a* ***phase****, which means it is invertible and satisfies:*



Conversely, all phases define compact structures satisfying the equation.

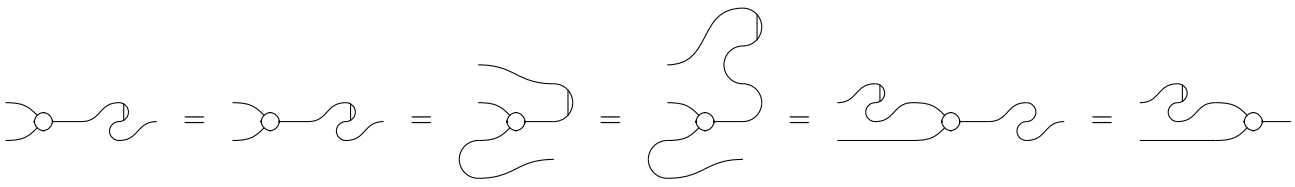
Proof. We start by showing that a compact structure of the form:

satisfies the flexsymmetry equation:



Now given a second compact structure satisfying the flexsymmetry equation. We have:

It only remains to show that  is a phase:



□

Note that the dual result for co-monoids also holds. So we just have to describe the phase groups of the monoids and co-monoids involved in bi-algebra pairs to obtain all the Frobenius algebras of interest for us. The phases of the Z, X, W and H monoids are of the form:

Z	X	H	W
$a, b \in \mathbb{C}^*$	$a, b \in \mathbb{C}^*$	$a, b \in \mathbb{C}^*$	$a \in \mathbb{C}^*$ $b \in \mathbb{C}$
$a\begin{pmatrix}1 & 0\\ 0 & b\end{pmatrix}$	$\frac{a}{2}\begin{pmatrix}1+b & 1-b\\ 1-b & 1+b\end{pmatrix}$	$a\begin{pmatrix}1 & 1-b\\ 0 & b\end{pmatrix}$	$a\begin{pmatrix}1 & 0\\ b & 1\end{pmatrix}$

The phase group of Z, X and H is ${\mathbb{C}^*_\times}^2$. The phase group of W is $\mathbb{C}^*_\times \times \mathbb{C}_+$. We see without surprise that the group of invertible scalars $\mathbb{C}^*_\times$ appears in both groups.

For each monoid Z, X, H and W we define a canonical compact structure making them flexsymmetric:

Z	X	H	W
$\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$ $(1\ 0\ 0\ 1)$	$\frac{1}{2}\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$ $(2\ 0\ 0\ 2)$	$\begin{pmatrix}2\\-1\\-1\\1\end{pmatrix}$ $(1\ 1\ 1\ 2)$	$\begin{pmatrix}0\\1\\1\\0\end{pmatrix}$ $(0\ 1\ 1\ 0)$

6.3 Putting thing together

Now that we know all the bi-algebra pairs and all compact structures making the different parts of those pairs flexsymmetric we look for the compatible compact structures.

6.3.1 Compatibility

All bi-algebra pair we consider are of the form ZA where A is a monoid. We have defined a canonical compact structures for each monoid A. We call the **dualizer of the pair** d_{ZA}, the dualizer between the canonical compact structure of A and the compact structure of Z. We have:

$$d_{ZZ} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \qquad d_{ZX} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \qquad d_{ZH} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \qquad d_{ZX} = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$$

All the other possible compact structures are obtained by composition with a phase. Given a phase α of Z and a phase β of A, we denote $Z^\alpha A_\beta$ the bi-algebra pair ZA together with the pair of compact structures indexed by α and β. For $Z^\alpha A_\beta$ to form a Z^*-algebra we need the two compact structures to be compatible, this corresponds to the equation:

$$\beta \circ d_{ZA} \circ \alpha = \alpha^{-1} \circ d_{ZA}^{-1} \circ \beta^{-1}.$$

We will write phases as couple $(a, b) \in \mathbb{C}^*$ for Z, X and H and as couple $(a, b) \in \mathbb{C}^* \times \mathbb{C}$ for W. We can now describe the compatible pairs.

Theorem 8. *The only Z^*-algebras up to isomorphism in* ***Lin*** *are, with $a, b \in \mathbb{C}^*$: $Z^{(a,b)}Z_{(\frac{1}{a},\frac{1}{b})}$, $Z^{(a,b)}Z_{(-\frac{1}{a},\frac{1}{b})}$, $Z^{(a,b)}Z_{(\frac{1}{a},-\frac{1}{b})}$, $Z^{(a,b)}Z_{(-\frac{1}{a},-\frac{1}{b})}$, $Z^{(a,1)}X_{(\frac{2}{a},1)}$, $Z^{(a,1)}X_{(-\frac{2}{a},1)}$, $Z^{(a,-1)}X_{(\frac{2}{a},1)}$, $Z^{(a,-1)}X_{(-\frac{2}{a},1)}$, $Z^{(a,\frac{4}{a^2b^2})}X_{(b,-1)}$, $Z^{(a,-1)}X_{(b,\frac{4}{a^2b^2})}$, $Z^{(a,\frac{1}{a^2b^2-1})}H_{(b,\frac{1-a^2b^2}{a^2b^2})}$ with $a^2b^2 \neq 1$, and $Z^{(a,\frac{1}{a^2b^2})}W_{(b,0)}$.*

Proof. The candidate Z^*-algebras are $Z^\alpha Z_\beta$, $Z^\alpha X_\beta$, $Z^\alpha W_\beta$ and $Z^\alpha H_\beta$. We check compatibility for all of them.

- $Z^\alpha Z_\beta$: The dualizer of ZZ is the identity. Let $\alpha = (a, b)$ and $\beta = (c, d)$, $a, b, c, d \in \mathbb{C}^*$. Z^α and Z_β are compatible iff

$$c \begin{pmatrix} 1 & 0 \\ 0 & d \end{pmatrix} a \begin{pmatrix} 1 & 0 \\ 0 & b \end{pmatrix} = \tfrac{1}{a} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{b} \end{pmatrix} \tfrac{1}{c} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{d} \end{pmatrix}$$

  This gives the system:

$$\begin{cases} a^2c^2 = 1 \\ b^2d^2 = 1 \end{cases}$$

  The Z^*-algebras are then $Z^{(a,b)}Z_{(\frac{1}{a},\frac{1}{b})}$, $Z^{(a,b)}Z_{(-\frac{1}{a},\frac{1}{b})}$, $Z^{(a,b)}Z_{(\frac{1}{a},-\frac{1}{b})}$ and $Z^{(a,b)}Z_{(-\frac{1}{a},-\frac{1}{b})}$. The dualizer is the identity for $Z^{(a,b)}Z_{(\frac{1}{a},\frac{1}{b})}$.

- $Z^\alpha X_\beta$: The dualizer of ZX is $\frac{1}{2}$, its inverse is 2. let $\alpha = (a, b)$ and $\beta = (c, d)$, $a, b, c, d \in \mathbb{C}^*$. Z^α and X_β are compatible iff

$$c \begin{pmatrix} 1+d & 1-d \\ 1-d & 1+d \end{pmatrix} \tfrac{a}{2} \begin{pmatrix} 1 & 0 \\ 0 & b \end{pmatrix} = \tfrac{1}{a} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{b} \end{pmatrix} \tfrac{2}{c} \begin{pmatrix} 1+\frac{1}{d} & 1-\frac{1}{d} \\ 1-\frac{1}{d} & 1+\frac{1}{d} \end{pmatrix}$$

  This gives the system:

$$\begin{cases} (da^2c^2 - 4)(1 + d) = 0 \\ (a^2c^2bd + 4)(1 - d) = 0 \\ (a^2c^2b^2d - 4)(1 + d) = 0 \end{cases} \Leftrightarrow \begin{cases} \text{if } d = -1 \text{ then } \begin{cases} d = -1 \\ a^2c^2b = 4 \end{cases} \\ \text{if } d = 1 \text{ then } \begin{cases} d = 1 \\ a^2c^2 = 4 \\ b^2 = 1 \end{cases} \\ \text{else } \begin{cases} b = -1 \\ da^2c^2 = 4 \end{cases} \end{cases}$$

  The Z^*-algebras are then $Z^{(a,1)}X_{(\frac{2}{a},1)}$, $Z^{(a,1)}X_{(-\frac{2}{a},1)}$, $Z^{(a,-1)}X_{(\frac{2}{a},1)}$, $Z^{(a,-1)}X_{(-\frac{2}{a},1)}$, $Z^{(a,\frac{4}{a^2b^2})}X_{(b,-1)}$ and $Z^{(a,-1)}X_{(b,\frac{4}{a^2b^2})}$. The dualizer is the identity for $Z^{(a,1)}X_{(\frac{2}{a},1)}$.

- $Z^\alpha H_\beta$: The dualizer of ZH is $\begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}$ and its inverse is $\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$. let $\alpha = (a, b)$ and $\beta = (c, d)$, $a, b, c, d \in \mathbb{C}^*$. Z^α and H_β are compatible iff

$$c \begin{pmatrix} 1 & 1-d \\ 0 & d \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} a \begin{pmatrix} 1 & 0 \\ 0 & b \end{pmatrix} = \tfrac{1}{a} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{b} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \tfrac{1}{c} \begin{pmatrix} 1 & 1-\frac{1}{d} \\ 0 & \frac{1}{d} \end{pmatrix}$$

  This gives the system:

$$\begin{cases} a^2c^2(d+1)=1 \\ a^2c^2bd=-1 \\ a^2c^2b^2d^2=1+d \end{cases} \Leftrightarrow \begin{cases} a^2c^2\neq 1 \\ b=\frac{1}{a^2c^2-1} \\ d=\frac{1-a^2c^2}{a^2c^2} \end{cases}$$

The Z^*-algebras are $Z^{(a,\frac{1}{a^2b^2-1})}H_{(b,\frac{1-a^2b^2}{a^2b^2})}$ with $a^2b^2\neq 1$. The dualizer is the Hadamard gate in the case $a=1$ and $b=\sqrt{2}$.

- $Z^\alpha W_\beta$: the dualizer of ZW is $\begin{pmatrix}0&1\\1&0\end{pmatrix}$. Let $\alpha=(a,b)$ and $\beta=(c,d)$, $a,b,c\in\mathbb{C}^*$, $d\in\mathbb{C}$. Z^α and W_β are compatible iff

$$c\begin{pmatrix}1&0\\d&1\end{pmatrix}\begin{pmatrix}0&1\\1&0\end{pmatrix}a\begin{pmatrix}1&0\\0&b\end{pmatrix}=\tfrac{1}{a}\begin{pmatrix}1&0\\0&\frac{1}{b}\end{pmatrix}\begin{pmatrix}0&1\\1&0\end{pmatrix}\tfrac{1}{c}\begin{pmatrix}1&0\\-d&1\end{pmatrix}$$

This gives the system: $\begin{cases} d=0 \\ a^2c^2b=1 \end{cases}$.

The Z^*-algebras are $Z^{(a,\frac{1}{a^2b^2})}W_{(b,0)}$. The dualizer is the NOT gate in the case $a=1$ and $b=1$.

□

6.3.2 Essentially all Z^*-algebras

Now we have an exhaustive list of all Z^*-algebra in **Lin**. Now we will investigate this list in detail. In fact, most of the calculi of this list are here because of some symmetries that are broken by the definition of Z^*-algebras.

All our phase groups contain $\mathbb{C}^*_\times$ as a subgroup corresponding to floating scalars. So given a Z^*-algebra we can directly obtain another one by multiplying one compact structure by a scalar and dividing the other by the same scalar. This fact appears clearly in the classification when we always see a free a in the phases of Z. We can thus classify the Z^*-algebras up to scalar setting $a=1$, this gives:

Theorem 9. *The only Z^*-algebras up to isomorphism and up to scalars in **Lin** are, with $a,b\in\mathbb{C}^*$: $Z^{(1,b)}Z_{(1,\frac{1}{b})}$, $Z^{(1,b)}Z_{(-1,\frac{1}{b})}$, $Z^{(1,b)}Z_{(1,-\frac{1}{b})}$, $Z^{(1,b)}Z_{(-1,-\frac{1}{b})}$, $Z^{(1,1)}X_{(2,1)}$, $Z^{(1,1)}X_{(-2,1)}$, $Z^{(1,-1)}X_{(2,1)}$, $Z^{(1,-1)}X_{(-2,1)}$, $Z^{(1,\frac{4}{b^2})}X_{(b,-1)}$, $Z^{(1,-1)}X_{(b,\frac{4}{b^2})}$, $Z^{(1,\frac{1}{b^2-1})}H_{(b,\frac{1-b^2}{b^2})}$ with $b^2\neq 1$, and $Z^{(1,\frac{1}{b^2})}W_{(b,0)}$.*

We can now gather the calculus by groups. The existence of different instances for the ZZ, ZX, ZH, and ZW calculi comes from interesting commutation properties between phases of the two algebras.

- $Z^{(1,b)}Z_{(1,\frac{1}{b})}$, $Z^{(1,b)}Z_{(-1,\frac{1}{b})}$, $Z^{(1,b)}Z_{(1,-\frac{1}{b})}$ and $Z^{(1,b)}Z_{(-1,-\frac{1}{b})}$ can be related to each other using the fact that the phases are the same, so they all commutes.

- $Z^{(1,1)}X_{(2,1)}$, $Z^{(1,1)}X_{(-2,1)}$, $Z^{(1,-1)}X_{(2,1)}$ and $Z^{(1,-1)}X_{(-2,1)}$ can be related to each other thanks to what is called *the π-commutation rule*: $(1,\lambda)_Z \circ (1,-1)_X = \lambda(1,-1)_X \circ (1,\frac{1}{\lambda})_Z$ where $(a,b)_Z$ is a phase of Z and $(a,b)_X$ is a phase of X..

- $Z^{(\frac{1}{b},b^2)}X_{(2,-1)}$ and $Z^{(1,-1)}X_{(2b,\frac{1}{b^2})}$ $\Rightarrow$ are related by the Hadamard isomorphism between X and Z and the aforementioned π commutation rule.

- $Z^{(1,\frac{1}{b^2})}W_{(b,0)}$ $\Rightarrow$ also relies to the π commutation rule but where the phase $(1,-1)_X$ is expressed as the dualizer of ZW.

- $Z^{(1,\frac{1}{b^2-1})}H_{(b,\frac{1-b^2}{b^2})}$ with $b^2 \neq 1$ $\Rightarrow$ provides the following commutation rule: $2\frac{\lambda+1}{\lambda}(1,\lambda)_Z \circ H \circ (1,\frac{1}{2}\frac{1}{\lambda+1})_H = (1,2(\lambda+1))_H \circ H \circ (1,\frac{1}{\lambda})_Z$ where $(a,b)_Z$ is a phase of Z, $(a,b)_H$ is a phase of H and H is the Hadamard gate.

6.3.3 Relation to known calculi

We will now compare the calculi we obtain with the literature.

- The ZZ-calculus never has been really considered, as having two identical spiders is not useful. However, its existence is not happenstance: in general, a Frobenius algebra would not make a bialgebra with itself. In this case, it works as Z is a *special* Frobenius algebra.

- The ZX-calculus [64] corresponds to what we call $ZX_{(2,2)}$. This is a particular calculus as the dualizer is trivial: both algebras have the same compact structure (up to scalars). There are a few substantial differences between our calculus and the ZX-calculus. Instead of using all possible phases in $\mathbb{C}^\star$, the authors use phases in the unit circle. Subsequent work [13] introduced so-called lambda boxes to restore all phases. Second, the $ZX_{(2,2)}$-calculus is a bit awkward as the two Frobenius algebras Z and $X_{(2,2)}$ are not isomorphic, but only isomorphic up to a scalar. By rescaling the X algebra, we can obtain a calculus where both algebras are dual, at the price of a slightly different bialgebra rule. The isomorphism corresponds to the Hadamard matrix; as this matrix is symmetric, we can add it to our language without losing flexsymmetry, and we obtain this way the ZX-calculus defined in [64].

- The ZW-calculus as discussed in [74, 81] is exactly what we call ZW. The calculus, however, does not use phases on the black nodes. The original ZW-calculus introduced in [**hadzihasanovic2015diagrammatic**] by the same author is slightly different. Intuitively it corresponds to a different kind of graphical language where the Z and W Frobenius algebras have been subdivided like in Chapter 5 in order to be compatible with a third compact structure. This led to a language with two harvestmen.

- The ZH-calculus as discussed in [66] is exactly what we call $ZH_{(\sqrt{2},-\frac{1}{2})}$. However the authors do not use phases on the white node, and use a different parametrizations of the phases on the black node. The phase they call x is what we would call the phase $(1,1-2x)_H$. This makes the spider rule more awkward in their calculus.

Chapter 7

Entracte: Graphical Linear Algebra

"Linear algebra is the Claude Makélélé of science and mathematics"

Pawel Sobocinski in [22]

This chapter is an introduction to a graphical language which will be used extensively in Chapter 9 and Chapter 10. The idea is to provide a graphical language that is able to express various parts of linear algebra in a graphical way. To do so the interpretations of diagrams are not linear maps as expected but linear relations, an extension of linear maps in a similar way that relations extend functions. This extension to a more exotic semantics reminds the situation of ZX-calculus with respect to quantum circuits. The analogy can go further since the similarity between ZX-calculus and graphical linear algebra is also present on the graphical level. This is this similarity that will be thoroughly exploited in Chapter 9. If the connection between the two languages is widely acknowledged by the two communities, the languages are still evolving quite independently. The focus of the graphical algebra community is mainly to provide semantics to more and more computational models while the ZX community is mostly dedicated to the possible applications to various areas of quantum computing. Here I will present how the concepts developed in Chapter 5 and Chapter 6 for quantum graphical languages also provide some insights on graphical linear algebra. And then we will see how graphical linear algebra can be useful in manipulated quantum graphical languages.

7.1 The language

In the same way, I introduced ZX-calculus in Chapter 3, I will present separately different components of the language and will only later provide a clear and compact axiomatization. So for now I will refer to the graphical language of graphical linear algebra as $\mathbb{GLA}$.

7.1.1 Matrices

We start by defining the generators of $\mathbb{GLA}$ that can be interpreted as linear maps and to do so we introduce the prop $\mathbf{Mat}_R$ where $\mathbb{R}$ is a semi-ring.

Definition 49 ($\mathbf{Mat}_R$)**.** *Given a semi-ring R the monochromatic prop $\mathbf{Mat}_R$ has for arrows $n \to m$ the matrices in $\mathcal{M}_{m\times n}(R)$. The composition is given by the matrix product and the tensor product by the direct product of matrices defined by $A \oplus B \stackrel{\text{def}}{=} \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}$.*

Even is both have matrices as arrows, $\mathbf{Mat}_{\mathbb{C}}$ is very different from **Lin** defined in chapter 3. Informally, there is more space in **Lin** since the Kronecker product has a higher dimension than the direct product. For example, there is only one scalar in $\mathbf{Mat}_R$, the empty matrix with zero rows and columns.

The swap generator has for interpretation:

$$[\![\times]\!] \stackrel{\text{def}}{=} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

The tensor product in $\mathbf{Mat}_{\mathbb{K}}$ is a product so we can define a copy and erasing while this is impossible in **Lin**. Those copy and erasing maps are generators of $\mathbb{GLA}$ defined as:

$$[\![\text{—}\bullet\!<]\!] \stackrel{\text{def}}{=} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad [\![\text{—}\bullet]\!] \stackrel{\text{def}}{=} (.)$$

Where $(.)$ is the empty matrix with one column and zero rows corresponding to the unique R-linear map $R \to \{0\}$. Those generators satisfy the equations of a co-monoid, corresponding to the graphical language $\mathbb{M}^{op}$ we already encountered in Chapter 3.

The addition provides a monoid in $\mathbf{Mat}_R$ defined by:

$$[\![>\!\circ\text{—}]\!] \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 1 \end{pmatrix} \qquad [\![\circ\text{—}]\!] \stackrel{\text{def}}{=} (.)$$

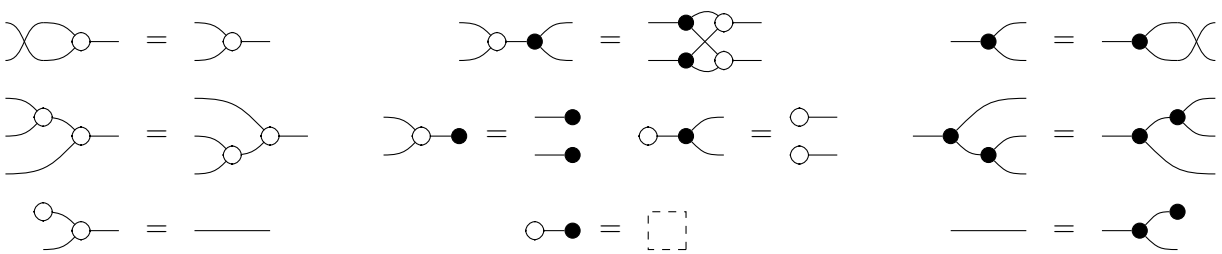
Where $(.)$ is the empty matrix with one row and zero columns corresponding to the unique R-linear map $\{0\} \to R$.

Addition and copy together form the following graphical language.

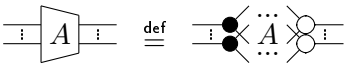
The graphical language $\mathbb{B}$ of bi-algebras

Generators $\mathbb{B}_s$

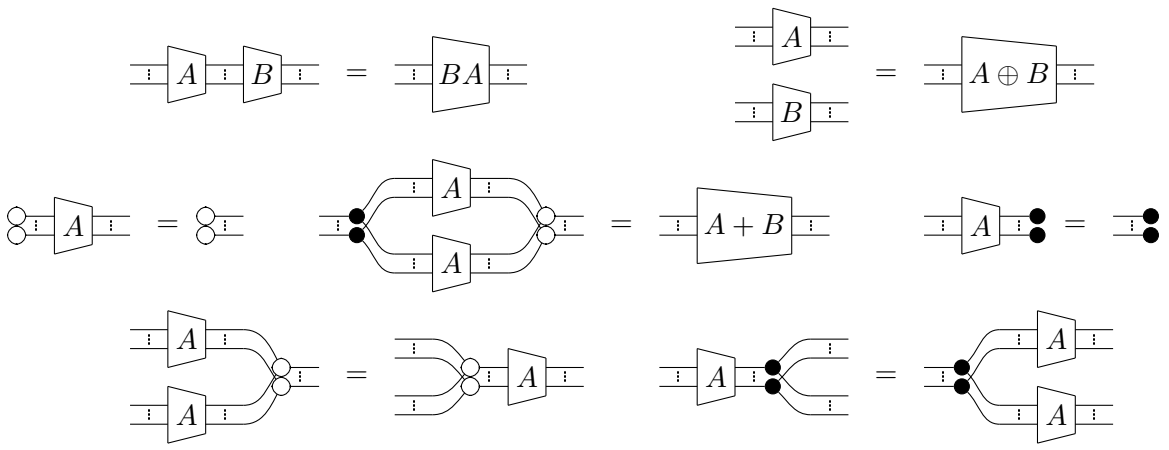
Equations $\mathbb{B}_e$



In fact this graphical language is complete for the prop $\mathbf{Mat}_{\mathbb{N}}$ [82]. We can represent the matrices as bipartite graphs. The matrix then corresponds to the bi-adjacency matrix of the graph. We will represent such graphs as:



A here is a $m \times n$ matrix with integer coefficient and represents a diagram $n \to m$. The black vertices are on the left and the white vertices on the right, hence the orientation of the notation. We have the following relations:



The interaction with the black co-monoid corresponds to copy and erasing. The interaction with the white monoid exactly states the linearity of A, the fact that $A0 = 0$ and $A(x + y) = Ax + Ay$.

7.1.2 Linear relations

We will now extend the language with more generators and equations. To do so we will not work with a general semi-ring R anymore but with a field $\mathbb{K}$.

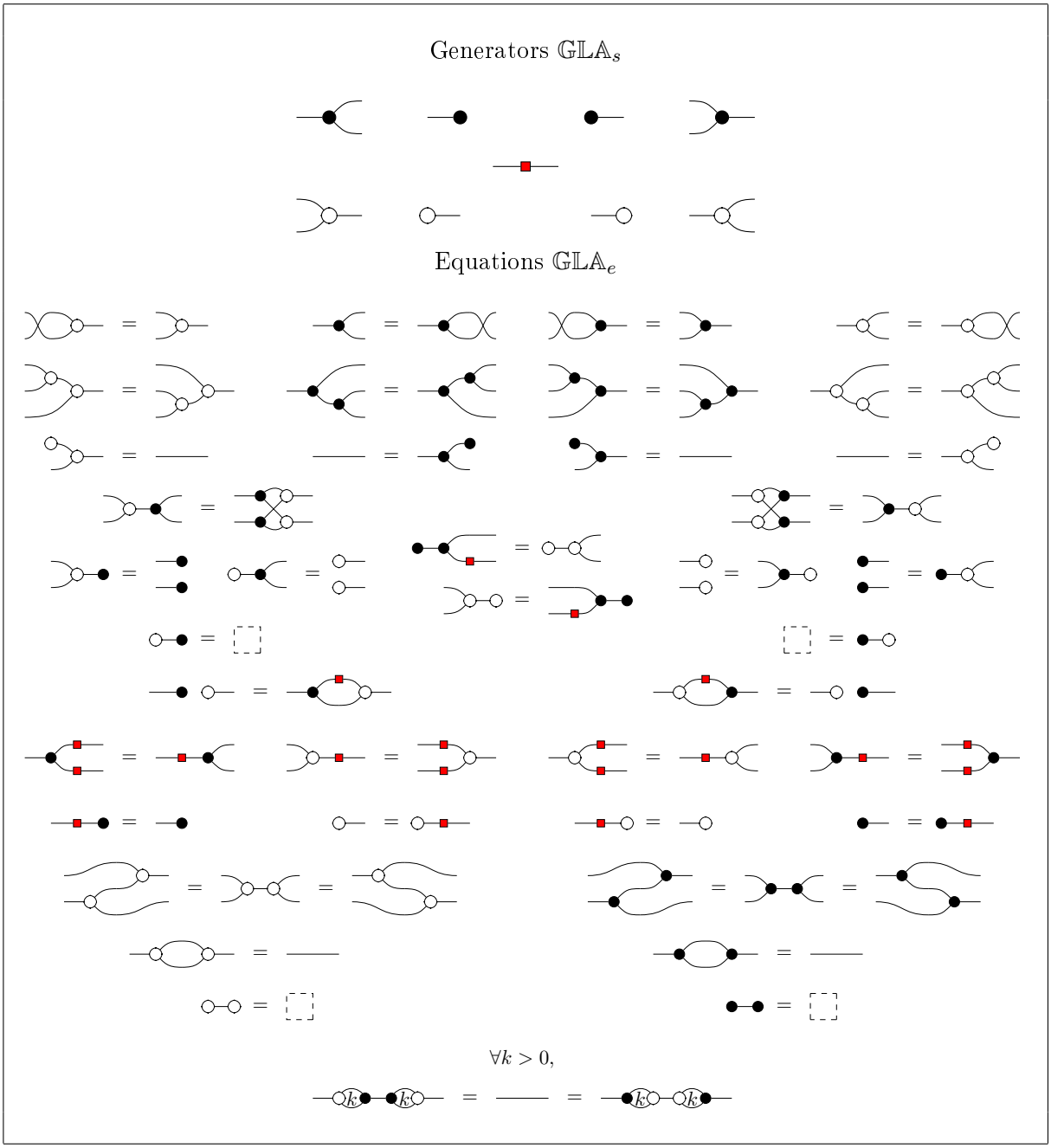
Definition 50 (Linear relations)**.** *A **linear relation** is a relation $\mathcal{R}$ between two $\mathbb{K}$-vector spaces V and W such that 0 is related to 0, i.e., $0\mathcal{R}0$, and $\mathcal{R}$ has to be stable by addition, that is: if $x\mathcal{R}y$ and $x'\mathcal{R}y'$ then $x + x'\mathcal{R}y + y'$.*

An equivalent and more compact definition of linear relations $\mathbb{R} : V \to W$ is a subspace of $V \oplus W$. This defines a prop.

Definition 51 (**LinRel**$_\mathbb{K}$)**.** *Given a field $\mathbb{K}$, the monochromatic prop* **LinRel**$_\mathbb{K}$ *has for arrows $n \to m$ the linear relations $\mathbb{K}^n \to \mathbb{K}^m$.*

We can now define the full graphical language $\mathbb{GLA}$.

The graphical language $\mathbb{GLA}$ of graphical linear algebra



Where the k stands for k wires in parallel. I gave here a huge set of equations that provide a good idea of the admissible rules and the different symmetries that the language enjoys. Of course, this set is not optimal, and more compact presentations have been given in [19] or [22]. The spider notation can reduce it, even more, this will be the object of a later section.

We see that the language feature an antipode $1 \to 1$ and two special Frobenius algebras whose spiders have for interpretations:

$$\left[\!\left[\;\right]\!\right] \stackrel{\text{def}}{=} \{((x,\cdots,x),(x,\cdots,x))\,,\ x\in\mathbb{Q}\}$$

$$\left[\!\left[\;\right]\!\right] \stackrel{\text{def}}{=} \left\{(\overrightarrow{x},\overrightarrow{y})\,,\ \sum_i x_i = \sum_j y_j\right\}$$

$$[\![\;]\!] \stackrel{\text{def}}{=} \{(x,-x)\,,\ x\in\mathbb{Q}\}$$

$$[\![\;]\!] \stackrel{\text{def}}{=} \{((x,y),(y,x))\,,\ x,y\in\mathbb{Q}\}$$

$$[\![\;]\!] \stackrel{\text{def}}{=} \{((x,x),0)\,,\ x\in\mathbb{Q}\}$$

$$[\![\;]\!] \stackrel{\text{def}}{=} \{(0,(x,x))\,,\ x\in\mathbb{Q}\}$$

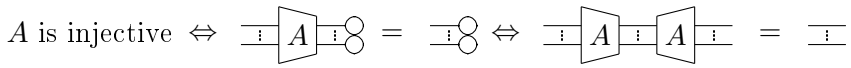
where $\overrightarrow{x} \stackrel{\text{def}}{=} (x_1,\cdots,x_n)$ and $\overrightarrow{y} \stackrel{\text{def}}{=} (y_1,\cdots,y_m)$. Here we also gave the interpretation of the swap and of the compact structure corresponding to the black spider.

This language have been shown to be complete for $\mathbf{LinRel}_{\mathbb{Q}}$ [19]. We can obtain a graphical language complete for $\mathbf{LinRel}_{\mathbb{F}_2}$, where $\mathbb{F}_2$ is the field with two elements, by setting:

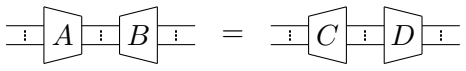
And then the infinite family of equations indexed by k is redundant in this case.

7.1.3 Properties

We state here numerous properties of linear relations that can be expressed purely graphically using graphical linear algebra.



We see here that the linear relations allow us to define backward matrices that do not in general correspond to linear maps. In general, the behaviours of those different kinds of diagrams are governed by the extremely powerful **matrix interaction equation**:



$$\Leftrightarrow \quad Im\begin{pmatrix} C \\ D \end{pmatrix} = Ker\begin{pmatrix} A & B \end{pmatrix}$$

There is a lot more that can be expressed using graphical linear algebra but we will not need more for the application to quantum computing of Chapter 9 and 10. I invite the interested reader to look at the references given at the beginning of this chapter.

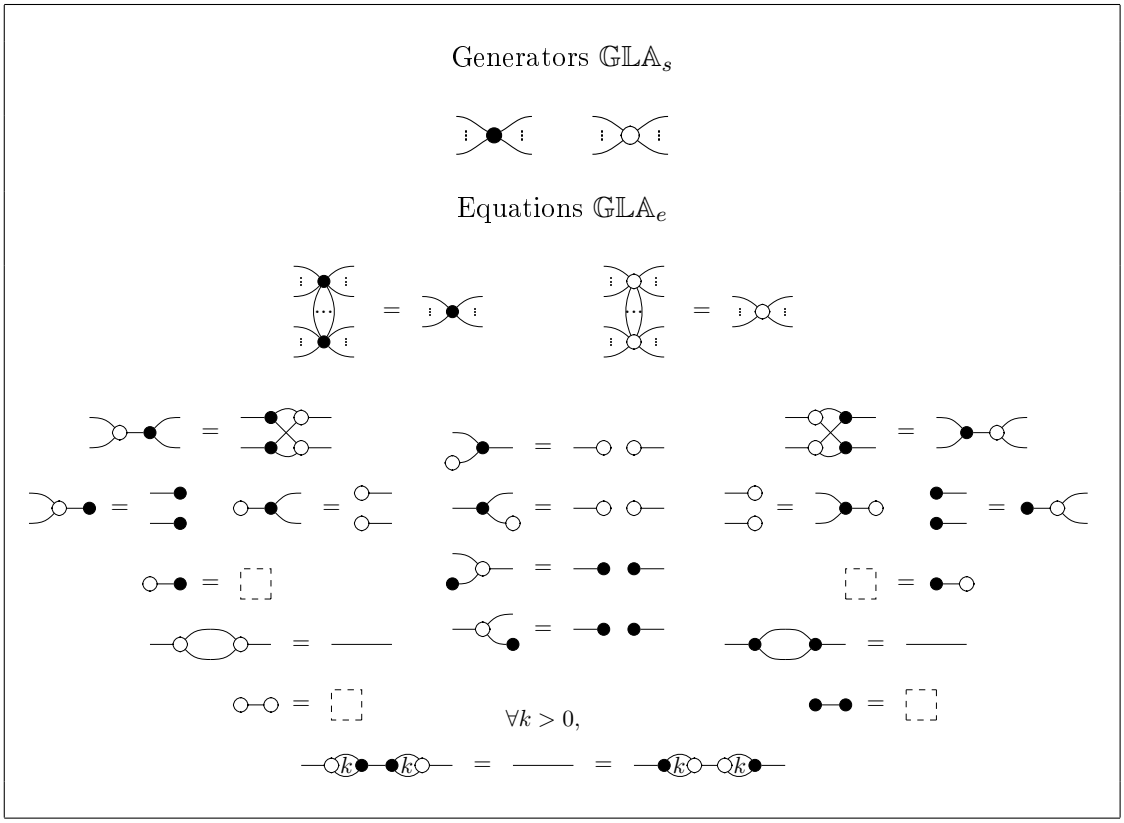
7.2 In hindsight

Now that we have defined $\mathbb{GLA}$ we will see how the concepts introduced in Chapter 5 and 6 can shed new light on this graphical language.

7.2.1 Simplifications

The presentation we gave of *GLA* is clearly not the more compact. It has been shown in [22] how to reduce dramatically the number of equations. The antipode can be derived as a dualizer between the black and white compact structure. We can gather the equations defining the two Frobenius algebras with the spider convention. This gives a simplified presentation.

The graphical language $\mathbb{GLA}$ of graphical linear algebra



In this form, the structure of graphical linear algebra appears very similar to the graphical calculus based on Z^*-algebras. We will investigate this similarity further.

7.2.2 Flexsymmetric graphical linear algebra

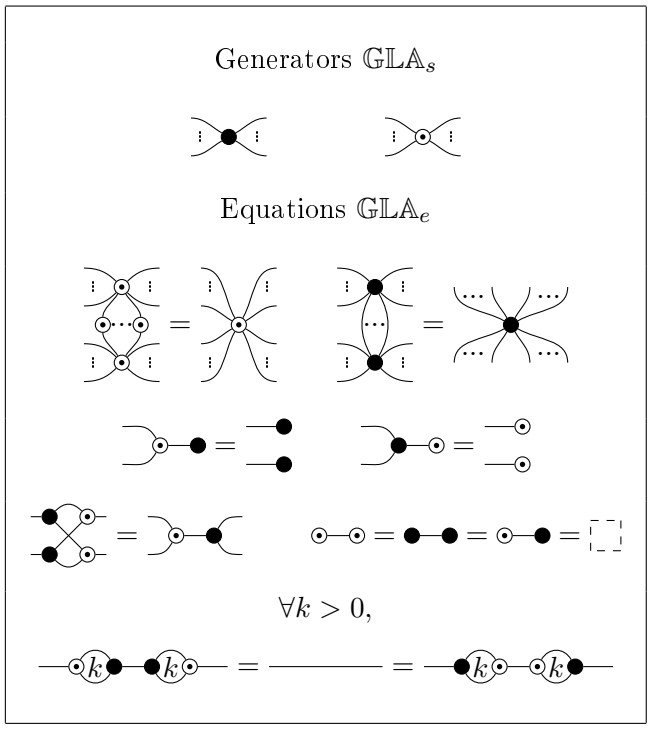
Graphical linear algebra has a compact structure and features two Frobenius algebras. It is natural to try to axiomatize it in flexsymmetric way. In fact, this can be done in a very similar way to the graphical Abelian group algebra example of Chapter 5.

The white spiders are flexsymmetric up to the dualizer  . Softening them gives an harvestmen defined as , with interpretation:

$$\left[\!\!\left[\right]\!\!\right] = \left\{ (\vec{x}, \vec{y}),\ \sum_i x_i + \sum_j y_j = 0 \right\}$$

we see directly in the interpretation how softening has broken the asymmetry between inputs and outputs. One has then a complete flexsymmetric graphical language.

The flexsymmetric graphical language $\mathbb{GLA}$ of graphical linear algebra



The reduction of the number of axioms seems radical but a lot of the original rules are in fact hidden into the spider convention or redundant under flexsymmetry.

7.2.3 As a Z^*-algebra

In its flexsymmetric form $\mathbb{GLA}$ clearly appears as a model of Z^*-algebra in $\mathbf{LinRel}_{\mathbb{Q}}$. In fact, we can prove that it is the only interesting Z^*-algebra there.

It turns out that there are only two monoids in $\mathbf{LinRel}_{\mathbb{K}}$, and they are not isomorphic: the monoid given by the subspace $\{(x,x,x), x \in \mathbb{K}\}$ and the monoid given by $\{(x,y,x+y), x,y \in \mathbb{K}\}$. Their respective phase groups are both trivial. Both these monoids, which we call B and N, actually happen to have Frobenius algebra structures. We then also have two co-monoids B and N

Lemma 11. *There are only four Z^*-algebras in $\mathbf{LinRel}_{\mathbb{K}}$: BB, NN, BN and NB.*

As these are the only potential candidates, we just have to check that they indeed give Z^*-algebras.

Proof. We start by showing that N and B are the only monoids and that their phase groups are trivial.

A subspace M of $\mathbb{K}^3$ is unital iff $\exists u \in \mathbb{K}, \forall x,y \in \mathbb{K}, ((u,x,y) \in M \Leftrightarrow x = y) \wedge ((x,u,y) \in M \Leftrightarrow x = y)$.

The trivial subspaces $\{0,0,0\}$ and $\mathbb{K}^3$ don't satisfy this property.

If M is of dimension one then there is a vector (a, b, c) such that $\forall x, y, z \in \mathbb{K}, (x, y, z) \in M \Leftrightarrow \exists \lambda \in \mathbb{K}, (x, y, z) = (\lambda a, \lambda b, \lambda c)$.

If M has a unit u, given an $x \in \mathbb{K}$ we have $(x, u, x) \in M$ and then $\exists \lambda \mathbb{K}, x = \lambda a, u = \lambda b, x = \lambda c)$. We know that $\lambda \neq 0$ else all triples (x, u, y) would be in M. This gives $a = c$, by symmetry we have also $b = c$. The only unital subspace of dimension one is N. It is also associative and thus is a monoid.

If M is of dimension 2 then there is a vector (a, b, c) such that $\forall x, y, z \in \mathbb{K}, (x, y, z) \in M \Leftrightarrow \exists \lambda \in \mathbb{K}, ax + by + cz = 0$.

If M has a unit u, given any $x \in \mathbb{K}$ we have $(x, u, x) \in M$ and then $ax + bu + cx = 0$. This gives $bu = 0$ and $c = -a$. By symmetry we also have $au = 0$ and $c = -b$. If $a = b = c = 0$ then all triple would be in M. We deduce that $a = b = -c \neq 0$ and $u = 0$. The only unital subspace of dimension 2 is μ_B. It is also associative and thus is a monoid.

Finally B and N are the only monoids in $\mathbf{LinRel}_{\mathbb{K}}$.

Now let α be a phase of μ_N, if $(x, y) \in \alpha$ then the phase's definition gives us that $x = y$, so $\alpha = id$ or $\alpha = (0, 0)$ the only invertible possibility is id. Now let β be a phase of μ_B, if $(x, y) \in \alpha$ then the phase's definition gives us that for all $z \in \mathbb{K}$ $(x + z, y + z) \in \alpha$, thus $\alpha = id$ or $\alpha = \mathbb{K}^2$ the only invertible possibility is id. So both phase groups are trivial.

So there is only one compact structure making respectively B and N flexsymmetric. So BB, NN, BN, and NB are the only candidates and we can check that they form Z^*-algebras.

□

BB and NN are trivial in the sense that they arise from special Frobenius algebras. BN is just a dual version of NB obtained by transposing everything, therefore the graphical calculi of [83], corresponding to BN, is essentially the only Z^*-algebra for this prop.

7.3 Models in Lin

It is now clear that quantum graphical languages share strong bonds with graphical linear algebra. In this section, we will look at how direct this connection can be made by looking for models of $\mathbb{GLA}$. More precisely we will look at prop morphisms $\mathbb{B} \to \mathbf{Lin}$. Those models will be extremely useful in Chapter 9 and 10 where we will use them to describe in a compact way huge graphical structures.

The main idea is that each time we have a bi-algebra then the bipartite graphs obtained from this model correspond to matrices the semi-ring $(\mathbb{N}, +, \times)$.

However, in general, our interpretation $\mathbb{B} \to \mathbf{Lin}$ will not be injective, and then two different integer matrices denote two bipartite graphs that are in fact equivalent in $\mathbf{Lin}$. Sometimes, this quotient happens in a very nice way that only amounts to look at matrices over another semi-ring, but weirder behaviours are also possibles. Thus we will proceed as follows, first, we will look at the 1×1 matrices, this will give us a first quotient allowing us to denote bipartite graphs by matrices over a semi-ring which is a quotient of $(\mathbb{N}, +, \times)$. Then, we will check if this is enough, meaning that two different matrices over this semi-ring correspond to bipartite graphs that are not equivalent in $\mathbf{Lin}$.

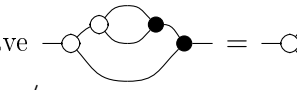
A basis of $\mathbb{C}^{2^n}$ is denoted by the $|x\rangle$ where x is a binary word of size n. e_i is the binary word with 0 everywhere except in the ith coordinate where it is 1. The three bi algebras we will consider all share the same co-monoid defined by:

$$[\![\text{  }]\!] = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } [\![\text{}]\!] = \begin{pmatrix} 1 & 1 \end{pmatrix}.$$

7.3.1 ZW

We start by the most ill behaved example. The ZW bi-algebra gives a model of $\mathbb{B}$, it is defined by:

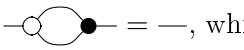
$$[\![\text{  }]\!] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \text{ and } [\![\text{}]\!] = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Looking at the 1×1 matrices, we have  . Thus, we are working with matrices over the semi-ring $(\mathbb{N}, +, \times) \Big/ (2 = 3)$. However, not all $\{0, 1, 2\}$-matrices have a distinct interpretation. If A is a $\{0, 1\}$-matrix then $A|e_j\rangle = |A_j\rangle$, but if A has a 2 in the jth column then for all x such that $x_j = 1$ we have $A|x\rangle = 0$. We cannot distinguish the coefficients in a column with a 2. So the bipartite graphs correspond to matrices with only $\{0, 1\}$-coefficients except in some columns which are full of 2s. We see that it seems that we can't do a lot by using graphical linear algebra inside of the ZW-calculus. Happily, we will be luckier with ZH and ZX.

7.3.2 ZH

The ZH bi-algebra gives a model of $\mathbb{B}$, it is defined by:

$$[\![\text{  }]\!] = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } [\![\text{}]\!] = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

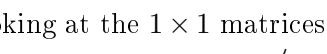
Looking at the 1×1 matrices, we have  = —, which essentially means "$2 = 1$". As a consequence, we are working with matrices over the semi-ring $(\mathbb{N}, +, \times) \Big/ (2 = 1)$. This is exactly the boolean semi-ring $(\vee, \wedge)$. There is no more quotienting since $A|e_j\rangle = |A_j\rangle$ where A_j is the jth column of A. All bipartite graphs corresponding to $\{0, 1\}$-matrix are then different. Here we have an interesting model of $\mathbb{B}$ that allows to consider matrices over the boolean semi-ring inside of the ZH-calculus. Those are the yellow arrows that will be introduced in Chapter 9.

7.3.3 ZX

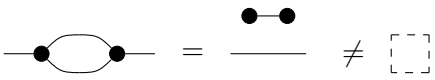
The ZX-calculus is clearly the quantum graphical language sharing the most with graphical algebra. The connection can be made by looking at $\mathbb{GLA}_{\mathbb{F}_2}$ where $\mathbb{F}_2$ is the field with two elements [21].

The ZX bi-algebra is defined by:

$$[\![\text{  }]\!] = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \text{ and } [\![\text{}]\!] = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

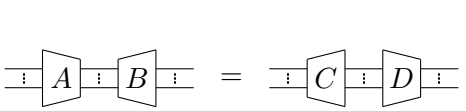
Looking at the 1×1 matrices, we have  . We are working with matrices over the semi-ring $(\mathbb{N}, +, \times) \Big/ (2 = 0)$. This is the field $\mathbb{F}_2$. We can see $\mathcal{F}_2$ as the set $\mathbf{2}$ equipped with the XOR sum $\oplus$. There is no more quotienting since $A|e_j\rangle = |A_j\rangle$. All $\{0,1\}$-matrices have a distinct interpretation. Those are the red arrows of Chapter 9.

In ZX, we can have a common compact structure making at the same time Z and X flexsymmetric, which is not the case for ZH or ZW. So we can also represent the backward matrices by transposing and hence any linear relation. However, we now have to be extremely careful since in our version of ZX-calculus the spider are only special up to a scalar, *i.e.*, they satisfy:



This implies that we don't really obtain a prop morphism $\mathbb{GLA} \to \mathbf{Lin}$ but a prop morphism $\mathbb{GLA} \to \mathbf{Lin}/_{scal}$ where $\mathbf{Lin}/_{scal}$ is the prop $\mathbf{Lin}$ quotiented by identifying all the non zero scalars to 1. One could speak of a projective model.

This interpretation of linear relations in ZX-calculus can be summarized by a modified version of the matrix interaction equation able to handle scalars:



$$(\bigstar)^k \qquad \Leftrightarrow \quad Im\begin{pmatrix} C \\ D \end{pmatrix} = Ker\begin{pmatrix} A & B \end{pmatrix}$$

Where $k \stackrel{\text{def}}{=} dim\left(Ker\begin{pmatrix} C \\ D \end{pmatrix}\right)$ and $[\![\bigstar]\!] = \frac{1}{\sqrt{2}}$. Here the boxes represent red/green bipartite graphs in ZX-calculus. We will come back to this way to see linear relation inside the ZX-calculus in details later in Chapter 8.

Part III

Add-ons

Chapter 8

The Discard Construction

Nan, mais j'assume. Moi j'aime bien.[a]

[a] I take full responsibility, I like that.
Simon Perdrix on drawing grounds [84]

While pure quantum evolutions correspond to linear maps over Hilbert spaces, probability distributions over quantum states as well as some quantum evolutions like measuring a quantum system can be represented by means of density matrices and completely positive maps. The category of completely positive maps has been already studied [85], and in particular, the connection between the pure and mixed state approaches is a central question in categorical quantum mechanics. Selinger introduced a construction called CPM to turn a category for pure quantum mechanics into a category for density matrices and completely positive maps [16]. Another approach to relate pure quantum mechanics to the general one is the notion of environment structure [86, 87, 88]. The CPM-construction and the environment structure approaches have been proved to be equivalent [87]. In [89] completely positive maps are represented by doubling the wires, this can be seen roughly as another way to present the CPM-construction.

In terms of graphical languages, the environment structure approach cannot be used in a straightforward way to extend a graphical language beyond pure quantum mechanics. Roughly speaking the environment structure approach provides second-order axioms which associate with any equation on arbitrary (non necessarily pure) evolutions an equivalent equation on pure evolutions. Such a second-order axiom cannot be easily handled by an equational theory on diagrams. Regarding the CPM-construction, the main property which has been exploited in [89] is that CPM($\mathbf{C}$) is essentially a subcategory of $\mathbf{C}$. Thus one can use a graphical language that has been designed for $\mathbf{C}$ in order to represent morphisms in CPM($\mathbf{C}$): Given a complete graphical language for $\mathbf{C}$, we can use a subset of the pure diagrams to represent the evolutions in CPM($\mathbf{C}$). The main caveat of this approach is that this subset is not necessarily closed under the equational theory on pure diagrams, and as a consequence does not provide a complete graphical language for CPM($\mathbf{C}$).

In [90], it was shown that the category $\mathbf{CPTP}$ of completely positive trace-preserving maps is the universal monoidal category with a terminal unit and a functor from the category of isometries. In this chapter we build upon this result by introducing a new construction, the *discard construction*, which transforms any †-symmetric monoidal category into a symmetric monoidal category equipped with a discard map. Roughly speaking this construction consists of making any isometry causal. Indeed, in quantum mechanics, the isometries (linear maps U such $U^\dagger \circ U = I$) are known to be causal, i.e., applying U and then discarding the subsystem on

which it has been applied is equivalent to discarding the subsystem straight away. Specifically, the discard construction proceeds as follows: first, the discard is added to the subcategory of isometries, making the unit of the tensor a terminal object in this subcategory, as pointed out in [90]. Then the discard construction is obtained as the push-out of the resulting category and the original one.

The discard construction does not always produce an environment structure for the original category, and thus is not equivalent to the CPM construction in general. However, a necessary and sufficient condition for the two constructions to be equivalent is that the initial category has enough isometries. We will see that most of the categories usually used in the context of categorical quantum mechanics, like **Lin**, **Stab** or **Rel**, do have enough isometries. However **Clifford+T** does not.

The discard construction also provides a simple recipe to extend the equational theory of ZX-calculus into a complete axiomatization for mixed-state quantum mechanics.

This chapter has some specificities compared with the others. We will here work with the general symmetric strict monoidal categories introduced in Chapter 0 instead of props. Also, we will not see our extension as a paradigm since the notion of paradigm presented in Chapter 4 is too weak for this construction. Furthermore, I will exceptionally use the top to bottom convention to write string diagrams.

8.1 Mixed state categorical quantum mechanics

In Chapter 2 we have introduced pure quantum mechanics where all evolutions are isometries. In such a model the measurement process cannot be represented since it requires classical non-determinism. In other words, we would need the possibility to consider probabilistic combinations of qubits and the associated stochastic quantum map. Such a model, which is the reunion of classical and quantum is achieved by the density matrix representation.

8.1.1 Density matrices

We will extend the model of quantum computation defined in Chapter 2.

Definition 52. *Density matrices A* ***mixed state*** *will be represented by a matrix ρ in $\mathcal{M}_{2\times 2}(\mathbb{C})$ which is required to satisfy $\rho^\dagger = \rho$, $\mathrm{tr}(\rho) = 1$ and must be* ***positive semi-definite****, in other words for any $x \in \mathbb{C}^2$ we must have $x^\dagger \rho x \geq 0$. Such matrix ρ is called a* ***density matrix****.*

Given a quantum state $|\phi\rangle$ we obtain the corresponding density matrix by multiply it with its dagger:

$$|\phi\rangle \mapsto |\phi\rangle \langle\phi|$$

To obtain the density matrix for a probabilistic mixture of the qubits $|x_i\rangle$ occurring respectively with probability p_i such that $\sum_{i=1}^n p_i = 1$, we take the convex sum of the density matrices corresponding to each qubits : $\sum_{i=1}^n p_i |x_i\rangle \langle x_i|$. It can be checked that this satisfies the properties of a density matrix.

Conversely, any density matrix can be written in this form. So density matrices exactly correspond to probabilistic mixtures of qubits.

Composite systems are defined by taking the Kronecker product of density matrices. A quantum process is then naturally defined as a linear map sending density matrices to density matrices, that is a trace-preserving positive linear map. But this is not sufficient, sadly, being

positive is not stable by tensor product. So we restrict to the trace-preserving positive maps ϕ such that for all $k \in \mathbb{N}$, $\phi \otimes id_{\mathcal{M}_{k\times k}\mathbb{C}}$ is positive. We call them **trace preserving completely positive maps** or **CPTP maps** and they are stable by tensor product. So we can define a prop.

Definition 53 (**CPTP**). *The monochromatic prop **CPTP** has for arrows $n \to m$ the trace preserving completely positive linear maps $\mathcal{M}_{2^n\times 2^n}(\mathbb{C}) \to \mathcal{M}_{2^m\times 2^m}(\mathbb{C})$. The tensor product being the usual tensor product of linear maps.*

In the same way that **Lin** generalizes **Qub** we can define a generalization of **CPTP** by dropping the trace-preserving condition.

Definition 54 (**CPM**). *The monochromatic prop **CPM** has for arrows $n \to m$ the completely positive linear maps $\mathcal{M}_{2^n\times 2^n}(\mathbb{C}) \to \mathcal{M}_{2^m\times 2^m}(\mathbb{C})$. The tensor product being the usual tensor product of linear maps.*

Given a linear map $V : \mathbb{C}^{2^n} \to \mathbb{C}^{2^m}$ we can define a completely positive map:

$$\rho \mapsto V\rho V^\dagger$$

Such maps are called **pure** and provide a prop morphism **Lin** $\to$ **CPM**. If V is an isometry then we obtain a CPTP maps and then a prop morphism **Qub** $\to$ **CPTP**. Some completely positive maps are not pure. An example is given by the trace:

$$\rho \mapsto \mathrm{tr}(\rho)$$

This map is extremely important, we call it the **discard map** and denote it by a ground:

$$[\![\perp]\!] \stackrel{\text{def}}{=} \rho \mapsto \mathrm{tr}(\rho)$$

In fact, in a sense, the discard map is the source of all impurity. The Stinespring dilation theorem tells us that any completely positive map is the composition of a pure map and a discard map.

Theorem 10 (Stinespring dilation theorem). *Given any completely positive linear map $\phi : \mathcal{M}_{2^n\times 2^n}(\mathbb{C}) \to \mathcal{M}_{2^m\times 2^m}(\mathbb{C})$ we can find a linear map $V : \mathbb{C}^{2^n} \to \mathbb{C}^{2^{k+m}}$ such that ϕ is of the form:*

$$\phi(\rho) = \left(\begin{matrix} \mathcal{M}_{2^k\times 2^k}(\mathbb{C}) \\ \perp \end{matrix} \otimes id_{\mathcal{M}_{2^m\times 2^m}(\mathbb{C})} \right) (V\rho V^\dagger)$$

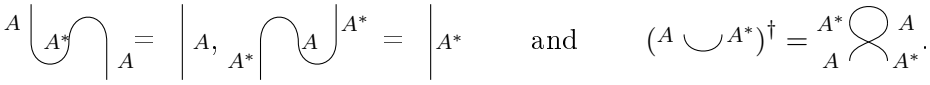
Furthermore, V is unique up to isometries, that is, if another linear map $V' : \mathbb{C}^{2^n} \to \mathbb{C}^{2^{k'+m}}$ satisfies the same property then (up to exchanging V and V') there is an isometry $K : \mathbb{C}^{2^k} \to \mathbb{C}^{2^{k'}}$ such that $(id_{\mathbb{C}^{2^m}} \otimes K) \circ V = V'$.

8.1.2 Dagger compact closed categories

The structure of Hilbert spaces can be abstracted with category theory. A † **strict symmetric monoidal category** (†-SMC) **C** is a strict symmetric monoidal category with an i.o.o. (identity on objects) involutive and contravariant SMC-functor $(.)^\dagger : \mathbf{C} \to \mathbf{C}$. That is, every morphism $f : A \to B$ has a dagger $f^\dagger : B \to A$ such that $f^{\dagger\dagger} = f$. Moreover the dagger respects the swaps $\sigma^\dagger_{A,B} = \sigma_{B,A}$. The dagger is a central notion in categorical quantum computing and can be used to define specific properties of morphisms:

Definition 55. *$f : A \to B$ is an* isometry *if $f^\dagger \circ f = id_A$, i.e.,*  .

Most of the categories we will consider are furthermore compact closed: A dagger compact category (†-CC) is a †-SMC where every object A has a dual object A^* such that for all objects A, there are two morphisms ${}^A\cup^{A^*} : A \otimes A^* \to I$ and ${}_{A^*}\cap_A : I \to A^* \otimes A$ satisfying:



Together with **Qub**, **Lin**, **CPM** and **CPTP** we will also consider the prop **Stab** and **Clifford+T**. **Stab** is the sub-prop of **Lin** which is finitely generated by the Clifford operators: H, S, CNot, the state $|0\rangle$, the projector $\langle 0|$, and the scalar 2 where:

$$\mathrm{H} = \tfrac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \mathrm{S} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \quad \mathrm{CNot} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \langle 0| = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

Those are amongst the most commonly used gates in quantum computation (see [91] for details). **Clifford+T** is the same as **Stab** but with the additional generator $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$. The morphisms of **Clifford+T** are exactly the matrices with entries in the ring $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$ [92]. Contrary to **Stab**, **Clifford+T** is approximately universal in the sense of Chapter 2.

8.1.3 CPM construction and environment structures

Connecting the pure quantum mechanics and mixed state quantum mechanics is a central question in categorical quantum mechanics. Selinger pointed out that any †-CC for pure quantum mechanics can be turned into a category for density matrices and completely positive maps via the CPM construction [16]:

Definition 56. *Given a †-CC* **C**, *let* CPM(**C**) *be the †-CC with the same objects as* **C** *such that*

$$\mathrm{CPM}(\mathbf{C})[A,B] = \left\{ \begin{array}{c} A \quad A^* \\ \boxed{f} \quad \boxed{f^*} \\ B \quad C \quad C^* \quad B^* \end{array} , f \in \mathbf{C}[A, B \otimes C] \right\}, \text{ where } \begin{array}{c} A^* \\ \boxed{g^*} \\ B^* \end{array} := {}_{B^*}\!\left(\begin{array}{c} B \\ \boxed{g^\dagger} \\ A \end{array} \right)^{A^*}.$$

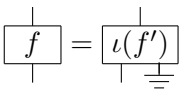
Applying it to **Lin** one obtains the category **CPM** of completely positive maps. The CPM construction can also be applied to **Stab** and **Clifford+T**. Notice that the CPM construction has been then extended to not necessarily compact categories [87].

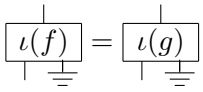
Another approach to relate pure quantum mechanics to the general one is the notion of environment structure [86, 87, 88]. The notion of *purification* is central in the definition of

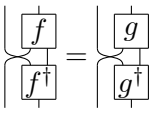
environment structure. Intuitively, it means that (1) there is a discard morphism for every object; (2) any morphism can be purified, i.e., decomposed into a pure morphism followed by a discarding map, and (3) this purification is unique up to a certain equivalence relation. More formally:

Definition 57. *An environment structure for a †-CC* $\mathbf{C}$ *is a CC* $\overline{\mathbf{C}}$ *with the same objects as* $\mathbf{C}$*, an i.o.o SMC-functor* $\iota : \mathbf{C} \to \overline{\mathbf{C}}$ *and for each object* A *a morphism* $\frac{\perp}{=}_A : A \to I$ *such that:*

(1) $\frac{\perp}{=}^{I} = 1_I$*, and for all* $A, B : \mathbf{C}$, $\frac{\perp}{=}^{A} \otimes \frac{\perp}{=}^{B} = \frac{\perp}{=}^{A \otimes B}$.

(2) *For all* $f : A \to B$ *in* $\overline{\mathbf{C}}$*, there is an* $f' : A \to B \otimes X$ *in* $\mathbf{C}$ *such that:* 

(3) *For any* $f : A \to B \otimes X$ *and* $g : A \to B \otimes Y$ *in* $\mathbf{C}$*:* $f \sim_{\text{cp}} g \Leftrightarrow$ 

where the relation $\sim_{\text{cp}}$ *is defined as:* $f \sim_{\text{cp}} g \Leftrightarrow$ 

Notice that $\sim_{\text{cp}}$ is technically not a relation on morphisms but on tuples (A, B, X, f) with $f \in \mathbf{C}[A, B \otimes X]$: $(A, B, X, f) \sim_{\text{cp}} (C, D, Y, g)$ if $A = C, B = D$ and f, g satisfy the graphical condition represented above. By abuse of notation, we write $f \sim_{\text{cp}} g$, as the other components of the tuple will be usually obvious from context. We will do the same for our relation $\sim_{\text{iso}}$ latter.

CPM is actually an environment structure for the category **Lin**, and more generally for any †-CC $\mathbf{C}$, CPM($\mathbf{C}$) is an environment structure for $\mathbf{C}$ and conversely any environment structure for $\mathbf{C}$ is equivalent to CPM($\mathbf{C}$) [87]. Actually one can notice that CPM($\mathbf{C}$)$[A, B]$ is nothing but the set of equivalence classes of $\sim_{\text{cp}}$.

8.2 Discard construction

We introduce a new construction, the *discard construction*, which consists in adding a discard map for every object of a †-SMC, and thus intuitively transforming a category for pure quantum mechanics into a category for general quantum evolutions.

Causality is a central notion in quantum mechanics which has been axiomatised using a discard map as follows [93]: $f : A \to B$ is *causal* if and only if  . Amongst the pure quantum evolutions, the isometries are causal evolutions. The discard construction essentially consists in making any isometry causal. Thus, whereas the CPM construction relies on completely positive maps and the environment structures on the concept of purification, the discard construction relies on causality.

8.2.1 Definition

We introduce the new construction in three steps. First, given a †-SMC, one can consider its subcategory of isometries:

Definition 58. *Given a †-SMC* $\mathbf{C}$*,* $\mathbf{C}_{\text{iso}}$ *is the subcategory with the same objects as* $\mathbf{C}$ *and isometries as morphisms, i.e., for all* $A, B : \mathbf{C}$*,* $\mathbf{C}_{\text{iso}}[A, B] = \{f : \mathbf{C}[A, B], f^\dagger \circ f = 1_A\}$.

Notice that $\mathbf{C}_{\text{iso}}$ is an SMC but usually not a †-SMC. Any †-SMC-functor $F : \mathbf{C} \to \mathbf{D}$ between two †-SMC can be restricted to their subcategories of isometries leading to an SMC-functor $F_{\text{iso}} : \mathbf{C}_{\text{iso}} \to \mathbf{D}_{\text{iso}}$. Thus there is a restriction functor iso : †-**SMC** $\to$ **SMC**. Note that this functor preserves fullness and faithfulness. One always has an inclusion i.o.o. faithful SMC-functor: $i_{\text{iso}} : \mathbf{C}_{\text{iso}} \to \mathbf{C}$.

In quantum mechanics, isometries are causal evolutions, i.e., applying an isometry and then discarding all outputs is equivalent to discarding the inputs straight away. As pointed out in [90], adding discard maps to the category of isometries would make I a terminal object. Such a category is said to be *affine symmetric monoidal category* (ASMC). We define the affine completion of an SMC:

Definition 59. *Given an SMC* $\mathbf{C}$*, we define* $\mathbf{C}^!$ *as* $\mathbf{C}$ *with an additional morphism* $!_A : A \to I$ *for each object* $A : \mathbf{C}$*. We denote the functor* $i^! : \mathbf{C} \to \mathbf{C}^!$ *which is strict monoidal and i.o.o. We further impose that* $1_I = !_I$*, and that for all* $f : \mathbf{C}[A, B]$*,* $!_B \circ i^!(f) = !_A$*. This makes* I *a terminal object in* $\mathbf{C}^!$*, and thus* $\mathbf{C}^!$ *is an ASMC.*

Notice by the way that $!_A \otimes !_B = 1_I \circ (!_A \otimes !_B) = !_I \circ (!_A \otimes !_B) = !_{A\otimes B}$. Again given a functor $F : \mathbf{C} \to \mathbf{D}$, one can define a functor $F^! : \mathbf{C}^! \to \mathbf{D}^!$ by $F^!(!_A) = !_{i^!(F(A))}$ and $F^!(f) = i^!(F(f))$ for the other morphisms. In [90], Huot and Staton show that **CPTPM**, the category of completely positive trace-preserving maps, is equivalent to $\mathbf{Lin}^!_{\text{iso}}$, thus giving a characterisation of it via a universal property. We extend this idea to non-trace preserving maps by proceeding to a local affine completion of the subcategory of isometries.

We define the category $\mathbf{C}^{\fallingdotseq}$ as the pushout of $\mathbf{C}$ and $\mathbf{C}^!_{\text{iso}}$:

Definition 60. *Given a †-SMC* $\mathbf{C}$*,* $\mathbf{C}^{\fallingdotseq}$ *is defined as the pushout in the category of a symmetric monoidal categories:*

$$\begin{array}{ccc} \mathbf{C}_{\text{iso}} & \xrightarrow{\ i_{\text{iso}}\ } & \mathbf{C} \\ {\scriptstyle i^!}\downarrow & & \downarrow{\scriptstyle \iota_{\mathbf{C}}} \\ \mathbf{C}^!_{\text{iso}} & \xrightarrow[\ \iota_{\mathbf{C}^!_{\text{iso}}}\ ]{} & \mathbf{C}^{\fallingdotseq} \end{array}$$

The existence of this pushout follows from the fact that the forgetful functor from strict symmetric monoidal categories to categories **StrictSymMonCat** $\to$ **Cat** preserves co-equalizers, and from [94, Theorem 9.3.9]. As all our functors are i.o.o., we can also describe it simply combinatorially. The objects of $\mathbf{C}^{\fallingdotseq}$ are the same as $\mathbf{C}$. Its morphisms are equivalence classes generated by formal composition and tensoring of morphisms in $\mathbf{C}^!_{\text{iso}}$ and $\mathbf{C}$. The equivalence relation is generated by the equations of both categories augmented with equations $i^!(f) = i_{\text{iso}}(f)$ for all f in $\mathbf{C}_{\text{iso}}$. The functors $\iota_{\mathbf{C}}$ and $\iota_{\mathbf{C}^!_{\text{iso}}}$ are the natural ways to embed $\mathbf{C}$ and $\mathbf{C}^!_{\text{iso}}$. We will see those formal compositions as string diagrams whose components are morphisms of $\mathbf{C}$ and $\mathbf{C}^!_{\text{iso}}$ wired to each other. Two diagrams represent the same morphism if we can rewrite one into the other applying the equations of both categories and $i^!(f) = i_{\text{iso}}(f)$ for all f in $\mathbf{C}_{\text{iso}}$. This forms a well-defined SMC.

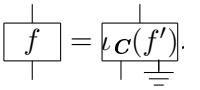
Since the only morphisms in $\mathbf{C}_{\text{iso}}$ which are not identified with the morphisms of $\mathbf{C}$ are those that contain $!_A$, we can see $\mathbf{C}^{\fallingdotseq}$ as $\mathbf{C}$ augmented with discard maps which delete isometries. A more detailed description of pushouts of props can be found in [95].

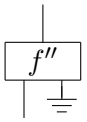
Definition 61. *The discard map on an object* A *is defined in* $\mathbf{C}^{\fallingdotseq}$ *by* $\overset{A}{\underset{\fallingdotseq}{\downarrow}} \overset{\text{def}}{=} \iota_{\mathbf{C}^!_{\text{iso}}}(!_A)$.

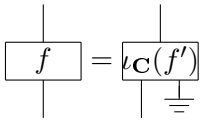
Notice, that for any isometry $f : A \to B$ in $\mathbf{C}^{\fallingdotseq}$,  $= \perp$, thus any isometry is causal.

8.2.2 Enough isometries

In order to compare the $\mathbf{C}^{\fallingdotseq}$ construction with environment structures and the CPM construction we need to study in detail the purification process in $\mathbf{C}^{\fallingdotseq}$. First notice that any morphism of $\mathbf{C}^{\fallingdotseq}$ admits a purification:

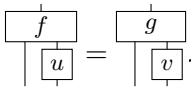
Lemma 12. *Let* $\mathbf{C}$ *be a* †*-SMC. For all* $f : \mathbf{C}^{\fallingdotseq}[A, B]$*, there is an* $X : \mathbf{C}$ *and an* $f' : \mathbf{C}[A, B \otimes X]$ *such that*  .

Proof. Given any morphism $f : \mathbf{C}^{\fallingdotseq}[A, B]$, we take a diagram representing it. Using the naturality of the symmetry we obtain an equivalent diagram in $\mathbf{C}^{\fallingdotseq}$ where all the discards have been pushed to the bottom right:  . There are no discards among the components of the part f'' of this diagram. So it represents a morphism in the range of $\iota_{\mathbf{C}}$ and then there is an $f' : \mathbf{C}[A, B \otimes X]$ such that:

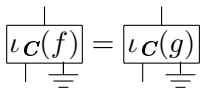


In other words, f' is a purification of f. □

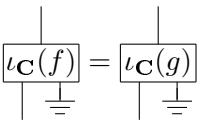
The purification needs not to be unique, however, it satisfies an essential uniqueness condition. To state it we define the relation $\sim_{\text{iso}}$.:

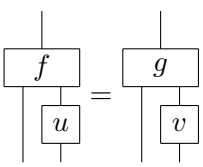
Definition 62. *Let* $\mathbf{C}$ *be a* †*-SMC, and two morphisms* $f : A \to B \otimes X$*,* $g : A \to B \otimes Y$*.* $f \sim_{\text{iso}} g$ *if there are two isometries* $u : X \to Z$ *and* $v : Y \to Z$*, such that*  .

Notice that the relation $\sim_{\text{iso}}$ is not transitive, thus we consider $\sim^{+}_{\text{iso}}$ its transitive closure to make it an equivalence relation. It is easy to show that if $f \sim^{+}_{\text{iso}} g$ then f and g purify the same morphism of $\mathbf{C}^{\fallingdotseq}$. The converse is also true:

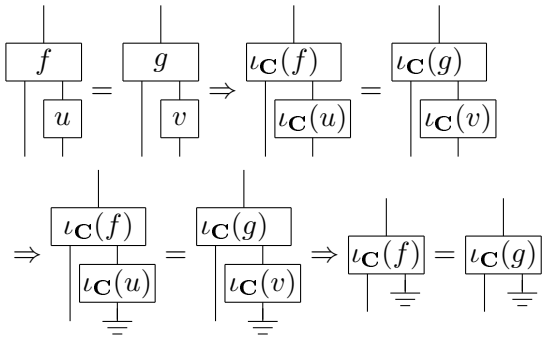
Lemma 13. *For all* $f : A \to B \otimes X$ *and* $g : A \to B \otimes Y$*:* $f \sim^{+}_{\text{iso}} g \Leftrightarrow$ 

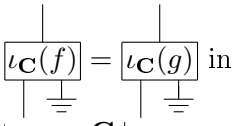
Proof of Lemma 13.

($\Rightarrow$) It is enough to show $f \sim_{\text{iso}} g \Rightarrow$  since equality is transitive.

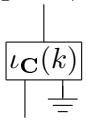
$f \sim_{\text{iso}} g \Leftrightarrow$ there are two isometries $u : X \to Z$ and $v : Y \to Z$ such that 

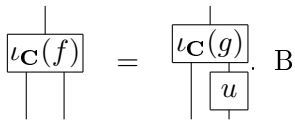
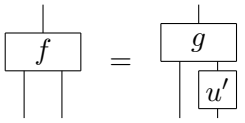
and then:



(⇐) We have  in $\mathbf{C}^{\doteqdot}$. To do the proof, we will have to go back to the definition of the category $\mathbf{C}^{\doteqdot}$ as a pushout. Recall that two terms are equal if one can rewrite one into the other using the equations defining $\mathbf{C}^{\doteqdot}$.

We can assume that, among those steps, the only one involving discards are isometry deletion/creation. Diagrammatically this amounts to saying that the discards are never moved, in fact, one can always move the other morphisms to make them interact with the discards.

Doing this, we ensure that all intermediary diagrams in the chain of equations are of the form  for some k. Therefore, to prove the result for a chain of equations of arbitrary size, it is enough to do it just for one step of rewriting.

Consider then this step of rewriting. There are two cases. Either we have used an equation which, by identification, can be seen as an equation of $\mathbf{C}$, that is which involves no discards. Then by functoriality of ι_C we recover that $f = g$ and therefore $f \sim_{\text{iso}} g$. Or the equation involves a discard which has deleted an isometry u. Then one of the upper part, let's say $\iota_C(f)$, can be written  . But u being an isometry, there exists u' in $\mathbf{C}$ such that $\iota_{\mathbf{C}}(u') = u$. Hence, we have  in $\mathbf{C}$. It follows that $f \sim_{\text{iso}} g$.

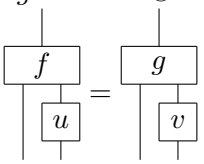
□

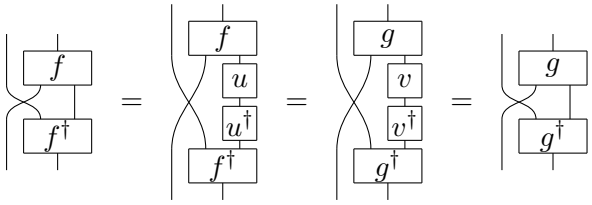
So the purification is unique up to $\sim^{+}_{\text{iso}}$. Lemma 13 also gives an alternative definition of $\mathbf{C}^{\doteqdot}$ which relates more easily to the CPM construction. It is the same construction as CPM with $\sim_{\text{cp}}$ replaced by $\sim^{+}_{\text{iso}}$. In other words $\mathbf{C}^{\doteqdot}[A, B]$ is the set of equivalent classes of $\sim^{+}_{\text{iso}}$.

As we have introduced a new discard construction, a natural question is whether $\mathbf{C}^{\doteqdot}$ is an environment structure for $\mathbf{C}$. To be an environment structure, three conditions are required. The first two are satisfied: $\mathbf{C}^{\doteqdot}$ has a discard morphism for every object and every morphism can be purified. The third one is the uniqueness of the purification: according to the definition of the environment structures, f and g purify the same morphism if and only if $f \sim_{\text{cp}} g$ whereas

according to Lemma 13, f and g purify the same morphism if and only if $f \sim^{+}_{\text{iso}} g$. As a consequence $\mathbf{C}^{\fallingdotseq}$ is an environment structure for $\mathbf{C}$ if and only if $\sim_{\text{cp}} = \sim^{+}_{\text{iso}}$. It turns out that one of the inclusions is always true:

Lemma 14. *For any †-SMC category* $\mathbf{C}$*, we have* $\sim^{+}_{\text{iso}} \subseteq \sim_{\text{cp}}$.

Proof of Lemma 14. Since $\sim_{\text{cp}}$ is transitive it is enough to show that $\sim_{\text{iso}} \subseteq \sim_{\text{cp}}$. Let $f : A \to B \otimes X$ and $g : A \to B \otimes Y$ s.t. $f \sim_{\text{iso}} g$. Then there are two isometries $u : X \to Z$ and $v : Y \to Z$ such that  and then:



So $f \sim_{\text{cp}} g$. □

As a consequence, if $\sim_{\text{cp}} \neq \sim^{+}_{\text{iso}}$, it means that there are some morphisms f, g that are equal in $\sim_{\text{cp}}$ but cannot be proved equal in $\sim^{+}_{\text{iso}}$. Intuitively it means that the category has not enough isometries to prove those terms equal, which leads to the following definition:

Definition 63. *A †-SMC category* $\mathbf{C}$ *has* enough isometries *if the equivalence relations* $\sim_{\text{cp}}$ *and* $\sim^{+}_{\text{iso}}$ *of* $\mathbf{C}$ *are equal.*

Lemma 15. *Given a †-SMC* $\mathbf{C}$*, the following properties are equivalent:*

1. $\mathbf{C}$ *has enough isometries;*
2. $\mathbf{C}^{\fallingdotseq}$ *is an environment structure for* $\mathbf{C}$*;*
3. $\mathbf{C}^{\fallingdotseq} \simeq \text{CPM}(\mathbf{C})$.

Proof of Lemma 15. [(1) ⇔ (2)] First $\mathbf{C}^{\fallingdotseq}$ has the same object as $\mathbf{C}$ and $\iota_{\mathbf{C}} : \mathbf{C} \to \overline{\mathbf{C}}$ is a SM-functor. We need to check the three conditions hold:

- Since $\iota_{\mathbf{C}^{!}_{iso}}$ is strict monoidal one has:

$$\overset{I}{\perp\!\!\!\fallingdotseq} = \iota_{\mathbf{C}^{!}_{iso}}(\ !_I) = \iota_{\mathbf{C}^{!}_{iso}}(id_I) = id_I$$
$$\overset{A}{\perp\!\!\!\fallingdotseq} \otimes \overset{B}{\perp\!\!\!\fallingdotseq} = \iota_{\mathbf{C}^{!}_{iso}}(\ !_A) \otimes \iota_{\mathbf{C}^{!}_{iso}}(\ !_B) = \iota_{\mathbf{C}^{!}_{iso}}(\ !_A \otimes\ !_B)$$
$$= \iota_{\mathbf{C}^{!}_{iso}}(\ !_{A\otimes B}) = \overset{A\otimes B}{\perp\!\!\!\fallingdotseq}$$

  So the first condition is satisfied.

- The second condition is Lemma 12.

- According to Lemma 14, $\sim_{\text{iso}}^{+} \subseteq \sim_{\text{cp}}$, thus the third condition is satisfied if and only if $\sim_{\text{cp}} \subseteq \sim_{\text{iso}}^{+}$.

[(1) $\Leftrightarrow$ (2)] Direct consequence of the fact that $\mathbf{D}$ is an environment structure for $\mathbf{C}$ iff $\mathbf{D}$ is equivalent to CPM($\mathbf{C}$) [87]. □

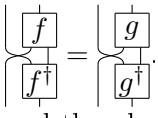
Notice that if $\mathbf{C}$ has enough isometries, the discard construction provides a definition of CPM($\mathbf{C}$) via a universal property. This gives a more direct way to build the environment, avoiding to deal with the equivalence classes of the CPM construction.

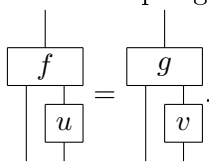
The notion of environment structures has also been generalised to the non-compact case [87]. Even if our construction does not require a compact structure, we chose here to focus on the compact case for two reasons. First, for now, the relation of our construction with environment structures is only clear with this hypothesis. Second the $\mathbf{CP}^{\infty}$ construction of [87] leads to a degenerate category when applied on the subcategory of isometries in **Lin**, this suggests that there might not have general connections between our construction and $\mathbf{CP}^{\infty}$.

Let's focus for a moment on the category Causal CPM($\mathbf{C}$) of causal maps, that is the subcategory of maps cancelled by the discards in CPM($\mathbf{C}$). We have that: $\sim_{\text{cp}} \subseteq \sim_{\text{iso}}^{+} \Rightarrow \mathbf{C}_{\text{iso}}^{!} \simeq$ Causal CPM($\mathbf{C}$). In fact by Lemma 15, CPM($\mathbf{C}$) $\simeq \mathbf{C}^{\fallingdotseq}$, and then the subcategory Causal CPM($\mathbf{C}$) is equivalent to the subcategory of maps cancelled by the discards in $\mathbf{C}^{\fallingdotseq}$ which is equivalent to $\mathbf{C}_{\text{iso}}^{!}$. Causal CPM(**Lin**) being exactly **CPTP**, we have recovered the result of [90].

We consider the usual subcategories of **Lin** used for pure quantum mechanics and show in each case whether the discard construction produces an environment structure or not. First of all, thanks to the Stinespring dilation theorem, $\mathbf{Lin}^{\fallingdotseq}$ is not only an environment structure for **Lin**, but the relation $\sim_{\text{iso}}$ is also transitive in this case:

Proposition 7. $\mathbf{Lin}^{\fallingdotseq}$ *is an environment structure for* **Lin**. *Furthermore* $\sim_{\text{iso}}^{+} = \sim_{\text{iso}}$.

Proof. Let $f : A \to B \otimes X$ and $g : A \to B \otimes Y$ be two linear maps such that $f \sim_{\text{cp}} g$. By definition: 

. It follows that the two superoperators $\rho \mapsto \text{tr}_X(f^{\dagger}\rho f)$ and $\rho \mapsto \text{tr}_Y(g^{\dagger}\rho g)$ are equal and then by the Stinespring dilation theorem (see for example [90]), there are isometries u and v such that 

. In other words $f \sim_{\text{iso}} g$. This shows that $\sim_{\text{cp}} \subseteq \sim_{\text{iso}}$ which is even stronger than having enough isometries. From Lemma 14 it follows that $\sim_{\text{iso}}^{+} \subseteq \sim_{\text{iso}}$. □

Such property is also true of the category **Rel** of sets and relations, giving an alternative description to the category CPM(**Rel**) studied in [96] and [97].

Proposition 8. $\mathbf{Rel}^{\fallingdotseq}$ *is an environment structure for* **Rel**. *Furthermore* $\sim_{\text{iso}}^{+} = \sim_{\text{iso}}$.

Proof. We just have to show that $\sim_{\text{iso}}^{+} \subseteq \sim_{\text{iso}}$ in **Rel**. $\mathcal{R} : A \to B$ is an isometry in **Rel** iff $\forall (x, y) \in A \times A$:

$$\exists z \in B,\ (x|z) \in \mathcal{R} \ \wedge\ (y|z) \in \mathcal{R} \quad \Leftrightarrow \quad x = y$$

In other words, $\mathcal{R}$ must be total and injective in the relational sense. Given two relations $\mathcal{F} : A \to B \times C$ and $\mathcal{G} : A \to B \times D$ we have $\mathcal{F} \sim_{\text{iso}}^{+} \mathcal{G}$ iff $\forall (a, b, a', b') \in A \times B \times A \times B$:

$$\exists c \in C,\ (a|b,c) \in \mathcal{F}\ \wedge\ (a'|b',c) \in \mathcal{F} \quad \Leftrightarrow \quad \exists d \in D,\ (a|b,d) \in \mathcal{G}\ \wedge\ (a'|b',d) \in \mathcal{G}$$

We denote $\mathcal{F}_{|c} = \{(a,b) \in A \times B,\ (a|b,c) \in \mathcal{F}\}$ and $\mathcal{G}_{|d} = \{(a,b) \in A \times B,\ (a|b,d) \in \mathcal{G}\}$. Given two relations $\mathcal{F} : A \to B \times C$ and $\mathcal{G} : A \to B \times D$ we have $\mathcal{F} \sim_{\text{iso}} \mathcal{G}$ iff there exists two isometries $\mathcal{U} : C \to H$ and $\mathcal{V} : D \to H$ such that: $\forall (a,b,h) \in A \times B \times H$:

$$\exists c \in C,\ (a|b,c) \in \mathcal{F}\ \wedge\ (c|h) \in \mathcal{U} \quad \Leftrightarrow \quad \exists d \in D,\ (a|b,d) \in \mathcal{G}\ \wedge\ (d|h) \in \mathcal{V}$$

Let $\mathcal{F} : A \to B \times C$ and $\mathcal{G} : A \to B \times D$ be two relations such that $\mathcal{F} \sim^{+}_{\text{iso}} \mathcal{G}$. We define two relations $\mathcal{U} : C \to \{0,1\} \times C \times D \times A \times B$ and $\mathcal{V} : D \to \{0,1\} \times C \times D \times A \times B$ as: forall $(c,c',d,d',a,b) \in C \times C \times D \times D \times A \times B$,

$$\begin{aligned}(c|0,c',d',a,b) \in \mathcal{U} \quad &\Leftrightarrow \quad c = c'\ \wedge\ \mathcal{F}_{|c} = \emptyset \\ (c|1,c',d',a,b) \in \mathcal{U} \quad &\Leftrightarrow \quad c = c'\ \wedge\ (a|b,c') \in \mathcal{F}\end{aligned}$$

and:

$$\begin{aligned}(d|0,c',d',a,b) \in \mathcal{V} \quad &\Leftrightarrow \quad d = d'\ \wedge\ \mathcal{G}_{|c} = \emptyset \\ (d|1,c',d',a,b) \in \mathcal{V} \quad &\Leftrightarrow \quad d = d'\ \wedge\ (a|b,d') \in \mathcal{G}\end{aligned}$$

U and V are isometries. We only show it for $\mathcal{U}$, the case of $\mathcal{V}$ being perfectly symmetric. Given $(c,i,c',d,a,b) \in C \times \{0,1\} \times C \times D \times A \times B$, $(c|i,c',d,a,b) \in \mathcal{U}$ implies by construction that $c = c'$. So $\mathcal{U}$ is injective. Given any $c \in C$, if $\mathcal{F}_{|c} = \emptyset$ then: $(c|0,c,d,a,b) \in \mathcal{U}$ for all $(d,a,b) \in D \times A \times B$. Else, if $\mathcal{F}_{|c} \neq \emptyset$ then there is $(a,b) \in A \times B$ such that $(c|1,c,d,a,b) \in \mathcal{U}$ for all $d \in D$. So $\mathcal{U}$ is total.So $\mathcal{U}$ is an isometry.

Given $(a,b,i,c',d',a',b') \in A \times B \times \{0,1\} \times C \times D \times A \times B$, we need to prove:

$$\exists c \in C,\ (a|b,c) \in \mathcal{F}\ \wedge\ (c|i,c',d',a',b') \in \mathcal{U} \quad \Leftrightarrow \quad \exists d \in D,\ (a|b,d) \in \mathcal{G}\ \wedge\ (d|i,c',d',a',b') \in \mathcal{V}$$

We distinguish to cases:

- If $i = 0$:

  $$\exists c \in C,\ (a|b,c) \in \mathcal{F}\ \wedge\ (c|0,c',d',a',b') \in \mathcal{U} \quad \Leftrightarrow \quad \exists d \in D,\ (a|b,d) \in \mathcal{G}\ \wedge\ (d|0,c',d',a',b') \in \mathcal{V}$$

  Unfolding the definition of $\mathcal{U}$ and $\mathcal{V}$ gives:

  $$\exists c \in C,\ (a|b,c) \in \mathcal{F}\ \wedge\ \mathcal{F}_{|c} = \emptyset \quad \Leftrightarrow \quad \exists d \in D,\ (a|b,d) \in \mathcal{G}\ \wedge\ \mathcal{G}_{|d} = \emptyset$$

  Both assertions are false so the equivalence holds.

- If $i = 1$:

  $$\exists c \in C,\ (a|b,c) \in \mathcal{F}\ \wedge\ (c|1,c',d',a',b') \in \mathcal{U} \quad \Leftrightarrow \quad \exists d \in D,\ (a|b,d) \in \mathcal{G}\ \wedge\ (d|1,c',d',a',b') \in \mathcal{V}$$

  Unfolding the definition of $\mathcal{U}$ and $\mathcal{V}$ gives:

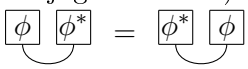
$$\exists c \in C,\ (a|b,c) \in \mathcal{F}\ \wedge\ (a'|b',c) \in \mathcal{F} \quad \Leftrightarrow \quad \exists d \in D,\ (a|b,d) \in \mathcal{G}\ \wedge\ (a'|b',d) \in \mathcal{G}$$

But this is exactly $\mathcal{F} \sim^{+}_{\text{iso}} \mathcal{G}$.

□

Notice that in general, the property of having enough isometries does not transfer to full subcategories: If $\mathbf{D}$ is a full subcategory of $\mathbf{C}$, we might have $f \sim^{+}_{\text{iso}} g$ in $\mathbf{C}$ but $f \not\sim^{+}_{\text{iso}} g$ in $\mathbf{D}$. This could happen for two reasons: First the chain of intermediate morphisms that prove that $f \sim^{+}_{\text{iso}} g$ might live outside of $\mathbf{D}$. Second, the isometries that "prove" that $f \sim^{+}_{\text{iso}} g$ in $\mathbf{C}$ might have codomain outside of $\mathbf{D}$.

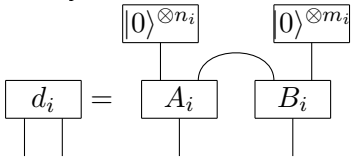
If our category is not a full subcategory, then everything falls apart, and finding conditions that guarantee that $\mathbf{C}^{\doteq}$ is an environment structure for $\mathbf{C}$ is not easy.

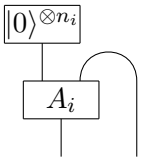
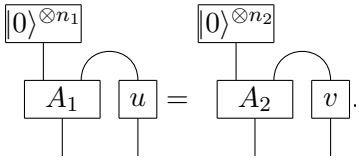
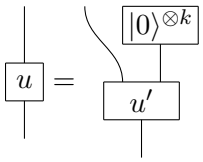
For subcategories of **Lin**, necessary conditions can be given. This category has the peculiarity that $\cdot^*$ is the identity on objects and that $f_{**} = f$ for all morphisms ($\cdot^*$ maps a matrix to its conjugate matrix). In particular, for any state $\phi : I \to I \otimes X$, we have $\phi^* \sim_{\text{cp}} \phi$. Indeed  .

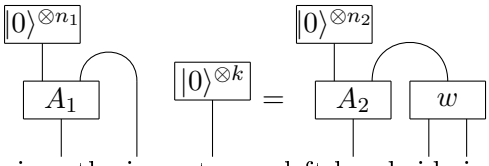
So a necessary condition for a subcategory of **Lin** to behave nicely is that for all states ϕ, we have $\phi^* \sim^{+}_{\text{iso}} \phi$. This is the case in **Stab**: Given a stabilizer state ϕ, there always exists a unitary U in **Stab** s.t. $U\phi = \phi^*$. In fact:

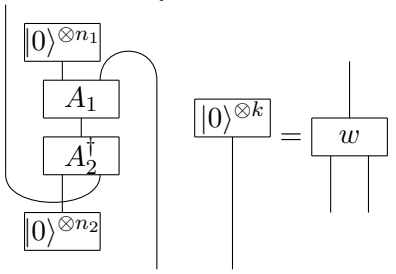
Proposition 9. $\mathbf{Stab}^{\doteq}$ *is an environment structure for* **Stab**.

The main idea of the proof is to use the map/state duality and structural results about bipartite stabilizer states [98].

Proof. First of all, since **Stab** is compact closed, using the map/state duality, proving the result for states is sufficient. Since all the non-zero scalars are invertible in **Stab** we can furthermore without loss of generality focusing on normalized states. Consider two states $d_1 : A \otimes X$ and $d_2 : A \otimes Y$ in **Stab** such that $d_1 \sim_{\text{cp}} d_2$. The point of focusing on normalized states is that we can decompose using [98] so that  where A_i and B_i are unitaries in **Stab**.

Defining $A'_i \stackrel{\text{def}}{=}$  we have that $d_i \sim_{\text{iso}} A'_i$ since we just have deleted isometries. So, by transitivity, to prove $d_1 \sim^{+}_{\text{iso}} d_2$ we just have to show $A'_1 \sim_{\text{iso}} A'_2$. But since $d_1 \sim_{\text{cp}} d_2$ in **Stab** we also have $d_1 \sim_{\text{cp}} d_2$ in **Lin** and so by Lemma 7, $d_1 \sim^{+}_{\text{iso}} d_2$ in **Lin**. By transitivity $A'_1 \sim^{+}_{\text{iso}} A'_2$ in **Lin** and so by Lemma 7 $A'_1 \sim_{\text{iso}} A'_1$ in **Lin**. So there are two unitaries u and v such that  . In **Lin** any isometry can be written as an unitary with ancillas. In other words there is an unitary u' such that:  , composing by $u'^{\dagger}$ on both side and

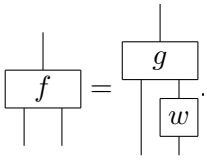
denoting $w = u'^{\dagger} \circ v$ one has: 

. It only remains to show that the isometry w is in **Stab** since the isometry on left hand side is clearly in it. This is given by: 

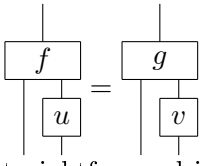
so $A'_1 \sim_{\text{iso}} A'_1$ in **Stab** and then $d_1 \sim_{\text{cp}} d_2$. □

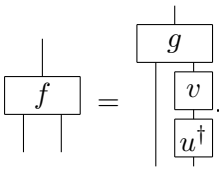
No such unitary exists in general in **Clifford+T**: For almost all states ϕ, there is no unitary U (and actually no morphism at all) s.t. $U\phi = \phi^*$. **Clifford+T** therefore has not got enough isometries:

Proposition 10. $(\mathbf{Clifford{+}T})^{\fallingdotseq}$ *is not an environment structure for* **Clifford+T**. *More precisely, there exists a state ϕ s.t. $\phi \sim_{\text{cp}} \phi^*$ but $\phi \not\sim^{+}_{\text{iso}} \phi^*$. One can take for example $\phi = 1 + 2i$ (in this case ϕ is a state with no input and outputs, hence a scalar).*

Proof. First note that, in any †-SMC category, if $f \sim^{+}_{\text{iso}} g$ then there is a morphism (usually not an isometry) w such that 

.

This is true if $f \sim_{\text{iso}} g$: From 

we immediately get 

.

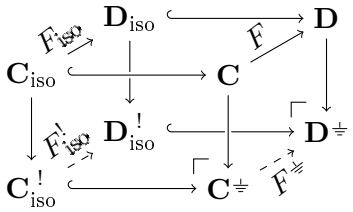
The result then follows by a straightforward induction.

Now take $\phi = 1 + 2i$ and $\phi^* = 1 - 2i$. The scalars are in **Clifford+T** since their entries are in $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$, and are clearly $\sim_{\text{cp}}$ equivalent. Now let's suppose $1 + 2i \sim^{+}_{\text{iso}} 1 - 2i$. Then by the previous remark, there exists a morphism u such that $(1-2i)u = 1+2i$. But the only possibility for u is $\frac{4i-3}{5}$, which is not in $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$, a contradiction. □

Note that for all categories above, we have $\sim^{+}_{\text{iso}} = \sim_{\text{iso}}$. That it holds in **Lin** is a consequence of the Witt extension theorem: Every isometry $f : A \to B$ is equal to a unitary $g : B \to B$ pre-composed with a canonical embedding from A to B. It it well known in **Stab** and it is true in **Clifford + T** by [99, Lemma 5].

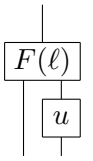
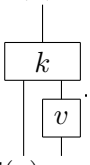
8.2.3 Completeness

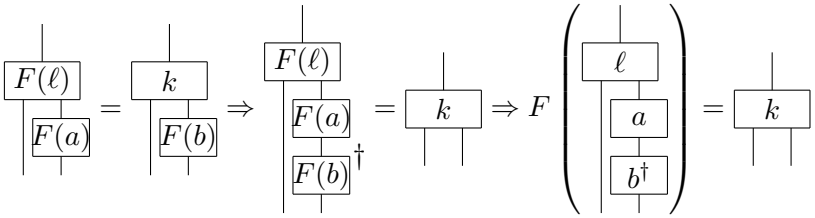
We now focus on the behaviour of interpretation functors with respect to the discard construction. The discard construction defines a functor $(\_)^{\fallingdotseq} : \dagger\text{-}\mathbf{SMC} \to \mathbf{SMC}$. Indeed, given a †-SMC functor F, F_{iso} and $F^{!}_{\text{iso}}$ uniquely define a functor $F^{\fallingdotseq}$ by push-out.



The following lemma and theorem are the main tools to apply the discard construction to graphical languages:

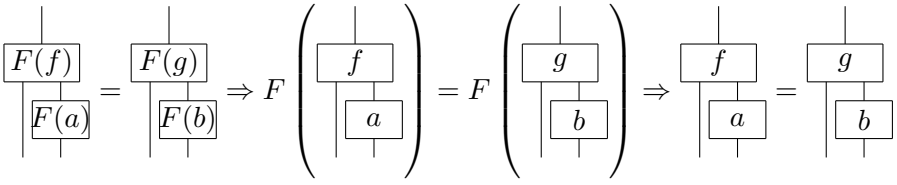
Lemma 16. *If F is faithful and if $F_{\text{iso}} : \mathbf{C}_{\text{iso}} \to \mathbf{D}_{\text{iso}}$ is full and surjective on objects, then $F(f) \sim^+_{\text{iso}} F(g) \Rightarrow f \sim^+_{\text{iso}} g$.*

Proof. First, remark that if $F(\ell) \sim_{\text{iso}} k$, then there exists h s.t. $F(h) = k$. Indeed, under the hypothesis, there are two isometries u and v such that:  =  . Since F_{iso} is full and surjective on objects, there are two isometries a and b such that $F(a) = u$ and $F(b) = v$.



The first implication uses the fact that $F(b)$ is an isometry. So k is in the image of F.

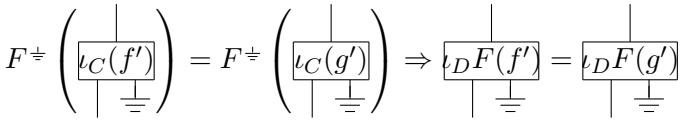
By the first remark, it is therefore sufficient to prove the result if $F(f) \sim_{\text{iso}} F(g)$. Since F_{iso} is full and surjective on objects, there are two isometries a and b such that $F(a) = u$ and $F(b) = v$. Therefore

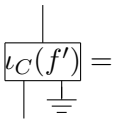


The second one holds because F is faithful. The last equation is the definition of $f \sim_{\text{iso}} g$. □

Theorem 11. *Let $\mathbf{C}$ and $\mathbf{D}$ be two †-SMCs and $F : \mathbf{C} \to \mathbf{D}$ a †-SMC-functor. If F is faithful and if $F_{\text{iso}} : \mathbf{C}_{\text{iso}} \to \mathbf{D}_{\text{iso}}$ is full and surjective on objects, then $F^{\fallingdotseq} : \mathbf{C}^{\fallingdotseq} \to \mathbf{D}^{\fallingdotseq}$ is faithful. If furthermore F is full then $F^{\fallingdotseq}$ is full and faithful.*

Proof. Let f and g be two morphisms such that $F^{\fallingdotseq}(f) = F^{\fallingdotseq}(g)$. By Lemma 12, f and g can be purified:



The implication follows from the upper face of the commutative cube. By Lemma 13 we have $F(f') \sim^{+}_{\text{iso}} F(g')$. By Lemma 16, $f' \sim^{+}_{\text{iso}} g'$. Then Lemma 13 gives  that is $f = g$, F is faithful. □

Notice that the hypothesis on F_{iso} is very strong, we want it to be surjective on objects as we do not want to lose even one isometry. If F is only required to be essentially surjective then our theorem holds only if the isomorphisms between objects are also unitary. In fact, the proper framework to express this condition would be to consider †-categories and not categories as being fundamental in the spirit of [100].

Reformulating for graphical languages this gives:

Corollary 1 (of Theorem 11)**.** *Given a †-CC* $\mathbf{C}$ *with enough isometries, if* $\mathcal{G}$ *is a †-CC universal complete graphical language for* $\mathbf{C}$ *then* $\mathcal{G}^{\perp}$ *is a universal complete language for* CPM($\mathbf{C}$).

This provides a general recipe. We start with a universal complete graphical language $\mathcal{G}$. We build $\mathcal{G}^{\perp}$, by Theorem 11, $[\![.]\!]^{\perp} : \mathcal{G}^{\perp} \to \mathbf{C}^{\perp}$ is full and faithful. Furthermore $\mathbf{C}^{\perp} \simeq \text{CPM}(\mathbf{C})$. $\mathcal{G}^{\perp}$ as a prop can be presented by adding one new generator to the signature Σ and one equation for each isometry of $\mathcal{G}$. Note that we add one equation for each isometry in $\mathcal{G}$ and that's all, there is no recursive process involved. In general, if one is provided with a spanning set of the isometries, the number of equations can be drastically reduced. We just need one equation for each element of this set. We then obtain a universal complete graphical language. For example if one finds an axiomatisation of quantum circuits complete for $\mathbf{Lin}_{iso}$ then the discard construction will apply since $\mathbf{Lin}_{iso}$ obviously has enough isometries.

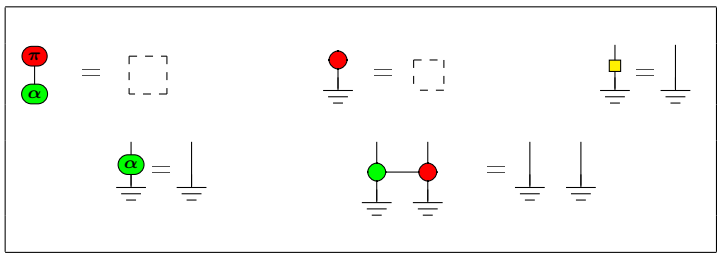
8.3 Application to ZX-calculus

We will now briefly review the extension of the ZX-calculus of Chapter 3. It is universal and complete for **Lin**. We will apply the recipe with a well-chosen spanning set and provide the additional axioms involving .

8.3.1 The ZX-Calculus with discard

We now come back to the ZX calculus of Chapter 3, Theorem 11 provides a recipe for transforming the language for mixed states and CPMs. The resulting language $\text{ZX}^{\perp}$ can be seen as a prop with the generators of the ZX-Calculus, augmented with and with the axiomatisation enriched with $\{\perp \circ D = \perp \mid D^{\dagger} \circ D = I\}$. We actually do not need an infinite axiomatisation. Indeed, the set of isometries of the ZX-Calculus can be finitely generated.

Using ($e^{i\alpha}$, $|0\rangle$, H, $R_Z(\alpha)$, CNot) as spanning set of the isometries [91], we obtain only five axioms:

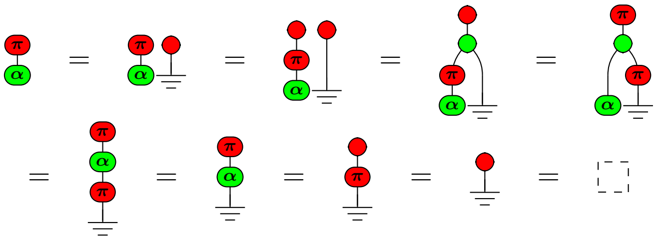


Here, to obtain more elegant axioms in interaction with the well-tempered normalization used in Chapter 3, I redefined the semantic of the ground as:

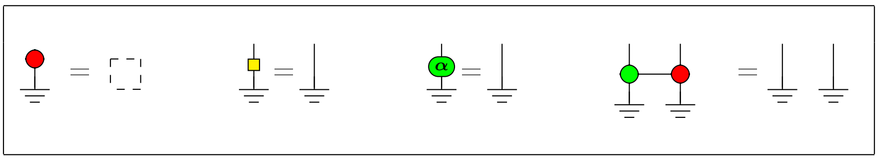
$$[\![⏚]\!] \overset{\mathsf{def}}{=} \rho \mapsto \frac{1}{\sqrt{2}}\,\mathrm{tr}(\rho)$$

The ground is then no longer normalized. So one needs to add the relevant scalars in the diagrammatic characterization of isometries. Notice that in the case of unitaries, when the input space has the same dimension as the output space, those additional scalars cancel, and then the equation is exactly the same.

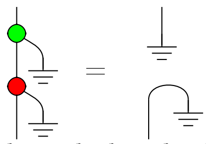
In fact, the first axiom follows from the others :

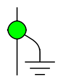
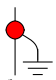


The final axiomatisation is then:



Notice that the last axiom, which means that CNot is causal, is equivalent to the following equation, first introduced in [88]:



In this equation, the bottom part of the right hand side is nothing but a maximally mixed state (i.e., the states $\binom{1}{0}$ and $\binom{0}{1}$ uniformly at random). Moreover,  and  can be interpreted as measurements in the complementary basis, thus the equation means that measuring a system according to two complementary bases is equivalent to discarding the system and replacing it with a maximally mixed state.

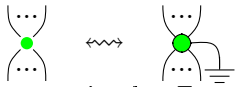
8.3.2 ZX-calculus with bastard spiders

In this section, contrary to the previous ones, we do not apply the ground construction to a graphical language, instead, we consider the extension of the ZX-calculus introduced in [89] for representing mixed states and classical channels. The main idea is to add two additional spiders, a new kind of green and a new kind of red spiders, the so-called *bastard spiders* depicted as follows:

 and 

They respectively represent decoherence in the Z and X basis. The behaviour of a bastard spider would be represented in the $ZX^{⏚}$-calculus as a standard spider with a ground attached

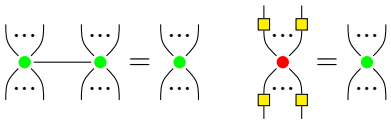
to it:



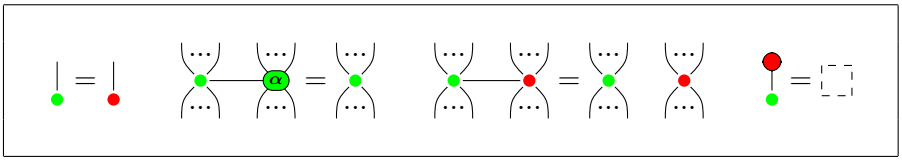
which respectively represent decoherence in the Z and X basis.

We give here an axiomatisation of those bastard spiders and show that it is equivalent to the discard construction, and then complete for quantum operations.

The new generators behave the same way as vanilla red and green spiders with respect to fusion and Hadamard gate:

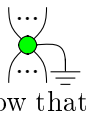


They also satisfy the following axioms:

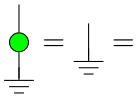


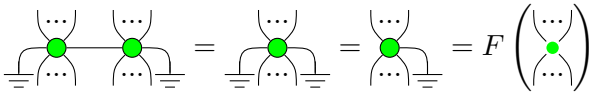
We call the ZX-calculus augmented by those generators and equations ZX^b.

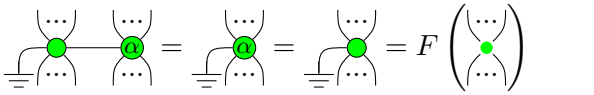
Lemma 17. $ZX^b \simeq ZX^{\div}$

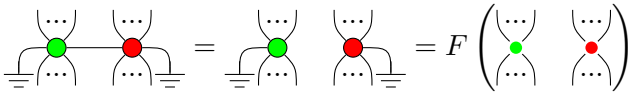
Proof of Lemma 17. Since both categories have the same objects we exhibit a full and faithful functor $F : ZX^b \to ZX^{\div}$ defined as the identity on the ZX part and which acts on the bastard spiders as: $F\left(\right) =$  and the colour swap for the red bastard spiders.

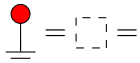
First we need to show that F is well defined. In other words that the images of the additional equations of ZX^b are still equal in $ZX^{\div}$:

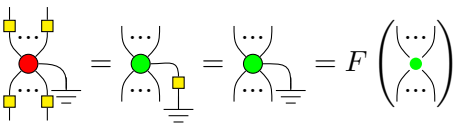
$F\left(\right) =$  $= F\left(\right)$

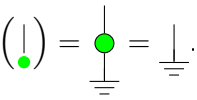
$F\left(\right) =$ 

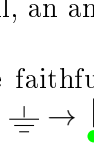
$F\left(\right) =$ 

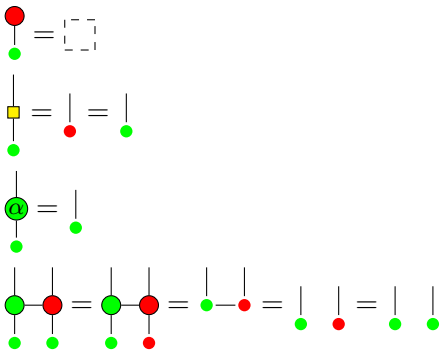
$F\left(\right) =$ 

$F\left(\right) =$  $= F\left(\right)$

$F\left(\right) =$ 

F is full, an antecedent of being given by  : F  .

For the faithfulness we show that the additional equations of $ZX^{\doteq}$ hold in ZX^{b} via the translation  :



So F is an equivalence of category. □

Remark: [89] also includes a new type of classical wire. However, we choose to drop it, first because it is not necessary to fully describe quantum operation, but mainly because even if a new type of wire could have been properly handled by the use of coloured props, the composition of bastard spiders of distinct colours through classical wire is ill-defined, which is inconsistent with a presentation with generators and equations.

Chapter 9

The Scalable Notations

"I've paid one PhD for each of those dots!"
Bob Coecke

We have seen in Chapter 3 that ZX-calculus diagrams are as powerful as matrices in terms of expressiveness. In Chapter 8 we saw that this equivalence between the two formalisms extends to mixed states quantum mechanics. In some situations, diagrams are a lot more compact than matrices and then allow short and intuitive computation that would involve very large matrices. However, in practice, we often work with matrices in an abstract way, allowing to reason on arbitrarily large families of quantum computations. Typical examples are the quantum algorithms that require the description of different quantum circuits for each possible size of the input.

A diagrammatic take on such families of circuits involves usually very big diagrams which are hard to draw, hard to read, or worst, feature the infamous informal three dots (note this has already been done several times in this thesis). This appeared quickly as a big weakness of diagrams: they do not scale well. Those considerations started the "dot war" and numerous solutions were proposed to tackle those scalability issues. Examples are the bang boxes introduced in [101] that provide a way to duplicate and erase part of a diagram. They form a meta-language admitting a first-order logic able to handle families of equations between diagrams. More generally, context-free grammars for graphical languages have been developed [102].

In this chapter, I will present yet another possibility: **scalable notations**. The main idea behind this is to stay strictly in the prop formalism without adding anything too exotic. This has the advantage that we can define everything in the framework described in Chapter 1. Scalable notations are quite primitive compared to the mathematical elegance of bang boxes or diagrammatic grammars. As a consequence, they are easier to manipulate but in general less powerful. However, we will see in Chapter 10 that they still are expressive enough to handle some basic quantum algorithms.

The first prototype of scalable notations can be found in [18], where the authors demonstrated that the ZX-calculus can be used in practice to design and verify quantum error-correcting codes. We have presented a formalisation of those notations in [7]. I will give here a slightly updated presentation.

9.1 Divide and gather

Before stating the construction allowing to apply the scalable construction on actual graphical language, we will start by presenting separately the different ingredients that will be combined later.

9.1.1 Types

The idea is to add to a monochromatic prop new types representing collections of wires. Typically we will be able to work with a tensor of n wires or with only one wire of size n representing those n wires in a compact way. Formally, with a $\mathbb{N}^*$-coloured prop, where $\mathbb{N}^*$ stands for $\mathbb{N}\backslash\{0\}$. Let $\mathbf{P}$ be a monochromatic prop and $\mathcal{S}\mathbf{P}$ be the $\mathbb{N}^*$-coloured prop that we construct. To differentiate clearly the two props we will use different notations.

In a $\mathbb{N}^*$-coloured props, a wire of colour $n \in \mathbb{N}$ will be denoted $[n]$. $[0]$ denotes the tensor unit. We write the tensor product $\otimes$ and denotes $[n]^m$ the tensor product $[n] \otimes \cdots \otimes [n]$ of m copies of $[n]$. By convention we set $[n]^0 \stackrel{\text{def}}{=} [0]$. We see that all object in $\mathcal{S}\mathbf{P}$ are of the form $\bigotimes_{i=1}^{m}[n_i]$. Given an object a of $\mathcal{S}\mathbf{P}$, its **size** $|a| \in \mathbb{N}$ is defined inductively as $|a \otimes b| \stackrel{\text{def}}{=} |a| + |b|$ and $|[n]| \stackrel{\text{def}}{=} n$. We say that a wire is **small** if it is of size 1. We say that a wire is big if it is of any size. When it comes to string diagrams we take the convention to denote the small wire by thin string and the big wire by bold strings. In general we will only indicate the colours on the wires if necessary, and in this case, we will use the strike wire notation.

9.1.2 The wire calculus

We can clearly see the types of $\mathbf{P}$ as types of $\mathcal{S}\mathbf{P}$ with $n \mapsto [1]^n$. The interest of $\mathcal{S}\mathbf{P}$ is that we can now consider a type $[n+m]$ which is different from the type $[n] \otimes [m]$. However it is possible to make those types interact together using two generators called **dividers** and **gatherers**.

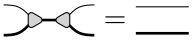
They are depicted as follows:

The $\mathbb{N}^*$-coloured signature containing the dividers $\delta_n : [n+1] \to [1] \otimes [n]$ for any size $n \in \mathbb{N}^*$ is denoted Δ. By convention $\delta_0 \stackrel{\text{def}}{=} id_{[1]}$ so it is not necessary to make it a part of Δ.

In the same way, the $\mathbb{N}^*$-coloured signature containing the gatherers $\gamma_n : [1] \otimes [n] \to [n+1]$ for any size $n \in \mathbb{N}^*$ is denoted Γ. By convention $\gamma_0 \stackrel{\text{def}}{=} id_{[1]}$ so it is not necessary to make it a part of Γ.

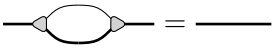
We define two families of equations over $\Delta + \Gamma$.

The family of **elimination** rules *Elim* gather the equations:



For any size $n \in \mathbb{N}^*$.

The family of **expansion** rules *Exp* gather the equations:



For any size $n \in \mathbb{N}^*$. Note that those rules are compatible with the convention $\delta_0 = \gamma_0 = id_{[1]}$. So we can write in full generality for all $n \in \mathbb{N}$:

$$\gamma_n \circ \delta_n = id_{[n+1]} \text{ and } \delta_n \circ \gamma_n = id_{[1]} \otimes id_{[n]}.$$

We now define the calculus of wires.

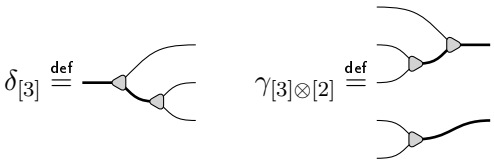
Definition 64 (𝕎-calculus)**.** *The **wire calculus** 𝕎 is defined as the $\mathbb{N}^*$-coloured graphical language:*

$$\mathbb{W} \stackrel{\text{def}}{=} (\Delta + \Gamma) \Big/ (Elim + Exp)$$

In 𝕎, the role of dividers and gatherers is perfectly symmetric, thus we have $\overset{\bullet}{\mathbb{W}} \simeq \overset{\bullet}{\mathbb{W}}^{op}$. So each time something is shown for dividers it also holds for gatherers by duality. The expansion and elimination rules exactly state that gatherers and dividers are isomorphisms hence $\overset{\bullet}{\mathbb{W}}$ is a groupoid. We also see that all the generators preserve the size. So there is cannot be any morphism of type $a \to b$ when $|a| \neq |b|$. The converse is also true.

We define inductively, for any object a of a $\mathbb{N}^*$-coloured prop, $\delta_a : a \to [1]^{|a|}$ as $\delta_{[0]} \stackrel{\text{def}}{=} id_{[0]}$, $\delta_{[1]} \stackrel{\text{def}}{=} id_{[1]}$, $\delta_{[n+1]} \stackrel{\text{def}}{=} \big([1] \otimes \delta_{[n]}\big) \circ \delta_n$ and $\delta_{a \otimes b} \stackrel{\text{def}}{=} \delta_a \otimes \delta_b$.

Similarly, we define inductively, for any object a of a $\mathbb{N}^*$-coloured prop, $\gamma_a : [1]^{|a|} \to a$ as $\gamma_{[0]} \stackrel{\text{def}}{=} id_{[0]}$, $\gamma_{[1]} \stackrel{\text{def}}{=} id_{[1]}$, $\gamma_{[n+1]} \stackrel{\text{def}}{=} \gamma_n \circ \big([1] \otimes \gamma_{[n]}\big)$ and $\gamma_{a \otimes b} \stackrel{\text{def}}{=} \gamma_a \otimes \gamma_b$.



So given two types a and b such that $|a| = |b|$ we have always an arrow $\gamma_b \circ \delta_a : a \to b$. So in general:

$$\overset{\bullet}{\mathbb{W}}\ [a, b] \neq \emptyset \qquad \Leftrightarrow \qquad |a| = |b|.$$

We can even go further in the description of the structure of 𝕎.

9.1.3 Rewiring theorem

We now show a kind of coherence theorem for 𝕎.

Theorem 12 (Rewiring theorem)**.** *Given two types a and b we have $|a| \neq |b| \Leftrightarrow \overset{\bullet}{\mathbb{W}}\ [a, b] = \emptyset$ and $|a| = |b| \Leftrightarrow \overset{\bullet}{\mathbb{W}}\ [a, b] \simeq \mathbb{P}_{\mathbb{N}}[|a|, |b|]$*

In other words, the arrows in $\overset{\bullet}{\mathbb{W}}$ are basically permutations over the simple wires contained in the input and output types.

Proof. We will show that for any diagram $\omega : a \to b$ we have $\omega = \Gamma_b \circ \sigma \circ \Delta_a$ where σ is a permutation of type $[1]^{|a|} \to [1]^{|b|}$. This will provide the expected bijection $\omega \mapsto \sigma$ between $\overset{\bullet}{\mathbb{W}}\ [a, b]$ and $\mathbb{P}_{\mathbb{N}}[|a|, |b|]$.

To do so we define, for each wire in the diagram, its **situation** as a couple of elements of $\{i, o, d, g\}$. The situation of a wire describes what the wire links to what: i stands for input, o for output, d for non identity divider, and g for non identity gatherer. For example, a wire which links an input to an output has situation (i, o) and a wire linking a gatherer to a divider has situation (g, d). The possible situations for a small wire are: (i, o), (i, g), (d, o), and (d, g). The possible situations for a big wire are the same plus (i, d), (d, d), (g, o), (g, g) and (g, d).

We say that a diagram is **expanded** if it contains no big wire in one of the **bad** situations which are (g, d) and (d, g).

The expanded condition enforces a unique structure. In fact, the only expanded diagrams are exactly the one of the form $\gamma_b \circ \sigma \circ \delta_a$. Thus it only remains to show that any diagram can be rewritten into an expanded one.

We proceed by induction on the size of the biggest big wire in a bad situation.

If there are no such wires then we are already in expanded form. Else, we consider the biggest wires in a bad situation. If a wire is in situation (g, d) then the elimination rule can be applied and decreases strictly the size of the big wires in bad situations. If a wire is in situation (g, d) then the expansion rule can be applied and decreases strictly the size of the big wires in bad situations. Thus all the wires in bad situations can be removed and replaced by wire with a strictly smaller size. Then by induction, all diagrams can be rewritten into expanded form.

□

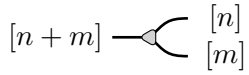
This gives us a clear understanding of what $\overset{\bullet}{\mathbb{W}}$ looks like as a category. The rewiring theorem works as a coherence result in the sense that any well-typed equation involving only dividers and gatherers holds. This gives us total freedom to rewire the way we want as soon as we preserve the order of the wires.

The way the dividers and gatherers are defined, taking the wires one by one, is useful to come up with a normal form but is still quite restrictive. We can define generalized dividers and gatherers able to relate a wire of size $n + m$ to one wire of size n and one wire of size m.

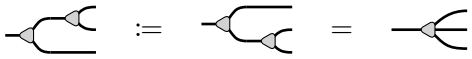
$\delta_{n,m} : [n + m] \to [n] \otimes [m]$ is defined as $\delta_{n,m} \overset{\text{def}}{=} \left(\gamma_{[n]} \otimes \gamma_{[m]}\right) \circ \delta_{[n+m]}$.

$\gamma_{n,m} : [n] \otimes [m] \to [n + m]$ is defined as $\gamma_{n,m} \overset{\text{def}}{=} \gamma_{[n+m]} \circ \left(\delta_{[n]} \otimes \delta_{[m]}\right)$.

We depict them as:



The rewiring theorem ensures that this is the only way to define those generalizations without permuting wires. We also know exactly what kind of equation they satisfy: all the well types one preserving the order of wires. In particular, an associativity-like law holds for generalized dividers and gatherers allowing us to define n-ary generalizations.



In full generality we can unambiguously define the rewiring maps $a \to b$ for a and b satisfying $|a| = |b|$ as $\gamma_b \circ \delta_a$.

We now proceed to make a monochromatic prop interacting with dividers and gatherers through the scalable construction.

9.2 Scalable construction

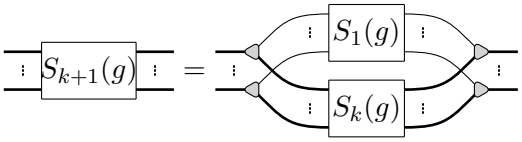
We will now describe how to construct a $\mathbb{N}^*$-coloured graphical language from a monochromatic one. The idea is to extend the monochromatic generators to big wires and then make them interact with the dividers and gatherers.

9.2.1 Definition

Given a monochromatic signature Σ, let $S_k\Sigma$ be the $\mathbb{N}^*$-coloured signature defined as $S_k\Sigma([k]^n, [k]^m) \stackrel{\text{def}}{=} \Sigma(n,m)$. We denote $S\sigma \stackrel{\text{def}}{=} \biguplus_{k\in\mathbb{N}^*} S_k\Sigma$. In $S\sigma$, each generator $x \in \Sigma(n,m)$ admits a **scaled version** of size k, $S_k s \in S\sigma([k]^n, [k]^m)$. By convention we set $S_0 x = id_0$ but do not include it in the signature.

Given a family of equations E over Σ we have a family of equations $S_kE \stackrel{\text{def}}{=} S_k \circ E$ over $S_k\Sigma$ defined by composing a family of equations with a signature map in the same way we did in Chapter 5. We define the family of equations $SE \stackrel{\text{def}}{=} \biguplus_{k\in\mathbb{N}^*} S_kE$. Intuitively, SE ensures that the equations valid for generators in Σ also hold for their scaled version in $S\Sigma$.

We define the family of **distribution rules** *Dist* over the signature $S\Sigma + \Delta + \Gamma$ as the union of all equations defined for each generator $x \in \Sigma(n,m)$ and $k \in \mathbb{N}^*$ as:



Definition 65 (Scalable graphical language)**.** *Given a monochromatic graphical language $\mathcal{L}$, we define an $\mathbb{N}^*$-coloured graphical language $\mathcal{SL}$ as:*

$$\mathcal{SL} \stackrel{\text{def}}{=} (S\mathcal{L}_s + \mathbb{W}) \Big/ (S\mathcal{L}_e + Dist_\Sigma)$$

9.2.2 Properties

Numerous properties follow from the construction. First, we define the scaling maps.

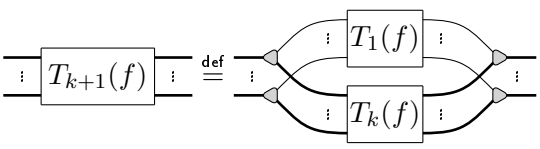
Definition 66 (scaling functors)**.** *For every $k \in \mathbb{N}^*$, the **scaling functor** $S_k : \overset{\bullet}{\mathcal{L}} \to \overset{\bullet}{\mathcal{SL}}$ is defined by $1 \mapsto [1]$ and $x \mapsto S_k x$.*

Note that this definition is sound since $S_k \circ E$ are among the equations of $\mathcal{SL}$ for any $k \in \mathbb{N}^*$. We set the convention $S_0 : x \to id_{[0]}$. We see that $\mathcal{SL}$ contains a copy of $\mathcal{L}$ for each k.

These functors ensure that any equation between diagrams still holds at a large scale. And one application of the scaled rule is in fact hiding k parallel applications of the original rule.

We can increase the size of scaled generators using the thickening functors.

Definition 67 (Thickening functors)**.** *We define inductively the **thickening functors** $T_k : \overset{\bullet}{\mathcal{SL}} \to \overset{\bullet}{\mathcal{SL}}$ as $T_k([n]) \stackrel{\text{def}}{=} [kn]$, $T_1 \stackrel{\text{def}}{=} id_{\overset{\bullet}{\mathcal{SL}}}$ and*



We have $T_l \circ T_k = T_{lk}$ and the distribution rules imply $T_l \circ S_k = S_{kl}$.

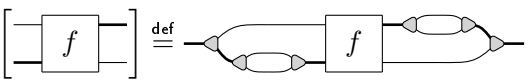
To come back from $\mathcal{SL}$ to $\mathcal{L}$ we can use the wire stripping functor.

Definition 68 (Wire stripper functor)**.** *The **wire stripper** functor $|\cdot| : \overset{\bullet}{\mathcal{SL}} \to \overset{\bullet}{\mathcal{L}}$ is defined on objects as $a \mapsto |a|$, where $|a|$ is the previously defined size of the type a, and on arrows by $|\delta_k| = id_k$, $|\gamma_k| = id_k$ and $|S_1 x| = x$ for all generators $x \in \mathcal{L}_s$.*

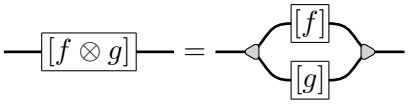
Using the distribution rules any scaled generators can be expressed as divides, gatherers, and generators S_1 so it is enough to define the behaviour of $|\cdot|$ on those generators. The soundness follows from

Note that we have $|\cdot| \circ S_1 = id_{\mathcal{L}}$. Note that the transformations $|\cdot| \circ S_k$ have been called monoidal multiplexing in [103] where it is shown that they amount to take a tensor product of k copies of the original diagram and add a permutation on the inputs and outputs. A way to understand those additional permutations is to see a scaled diagram as a parallel juxtaposition of copies of the same diagram in a three-dimensional space that has been then projected into a two-dimensional space. Then, parallel wires have no choice but to cross, hence the permutations.

Definition 69 (Boxing functor)**.** *The* ***boxing functor*** $[\_] : \mathcal{SL} \to \mathcal{SL}$ *is defined as* $a \mapsto [|a|]$ *on objects, where* $[|a|]$ *is the type of a unique big wire whose size is the one of the type* a*, and as* $f \mapsto \gamma_{[|b|]} \circ \delta_b \circ f \circ \gamma_a \circ \delta_{[|a|]}$ *on morphisms* $f : a \to b$*.*

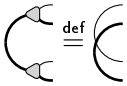


If the S_k, T_k and $|\cdot|$ are strict monoidal functors it is not the case of $[\cdot]$ that satisfies:

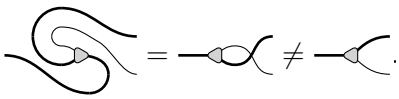


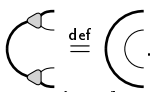
We expect most of the properties of $\mathcal{L}$ to be reflected in $\mathcal{SL}$. Here are some specific examples. If $\mathcal{L}$ is a dagger category then $\mathcal{SL}$ inherits this structure by setting ${\delta_k}^\dagger \stackrel{\text{def}}{=} \gamma_k$. Then we have ${\gamma_k}^\dagger = \delta_k$ and the expansion and elimination equations state that dividers and gatherers are unitary maps.

If $\mathcal{L}$ is compact closed then so is $\mathcal{SL}$. Using the scaled version of the cups and caps, we have:



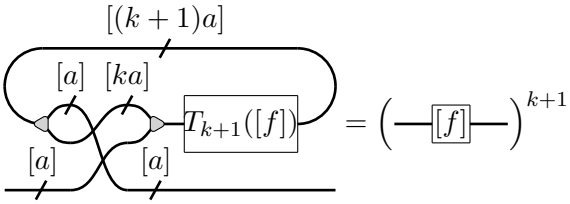
and then ${\gamma_k}^t = \delta_k$. However, note that some intuitive topological moves do not hold:



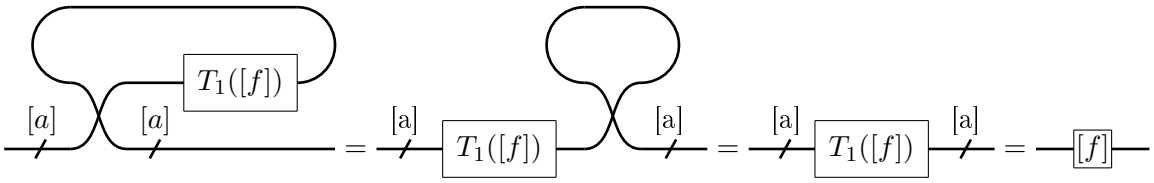
Another possibility is to take  . Then we recover the topology but we lose the correspondence between equations on simple wires and their scaled version.

When we have a compact structure, then an iteration mechanism is available.

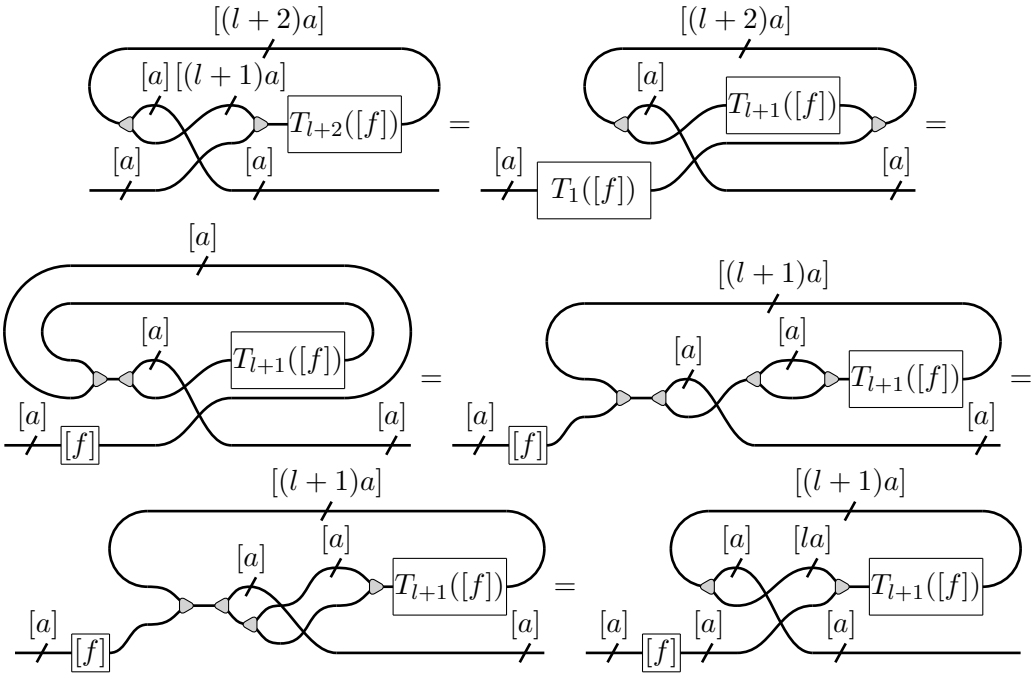
Lemma 18 (Iteration)**.** *Given any diagram* $f : a \to a$ *in* $\mathcal{SD}$*:*



Proof. By induction, for $k = 0$:



For $k > 0$, let $k = l + 1$:



□

This allows to represent for loops graphically. In practice, we need to compute the thickening of the boxed diagram. Thickening a scaled generator is by definition very easy, this only increases the size of every wires and generators. However, thickening a divider or a gatherer involves a permutation of the wires. This allows to apply the iteration mechanism as soon as we have a good representation of permutations.

9.2.3 Completeness

All the properties of $\mathcal{SL}$ follow from a structure theorem which can be seen as an extension to $\mathcal{SL}$ of the rewiring theorem.

Lemma 19 (Normal form)**.** *All diagrams $f : a \to b$ in $\mathcal{SL}$ are of the form :*

$$f = \gamma_b \circ S_1(|f|) \circ \delta_a$$

Proof. We use the same method as in the proof of the rewiring theorem. In $\mathcal{SL}$, there are new wire situations: s represents a simple generator and S. The new possible situations for a simple wires are (i, s), (d, s), (s, s), (s, g) and (s, o). For a big wire the new situations are (i, S), (d, S), (g, S), (S, S), (S, g), (S, d) and (S, o).

A diagram of $\mathcal{SL}$ is in expanded form if it contains no big wire in one of the bad situations which are (g,d), (d,g), (i,S), (d,S), (g,S), (S,S), (S,g), (S,d) and (S,o).

So an expanded diagram contains no big generators and is of the form $\omega = \gamma_b \circ \nu \circ \delta_a$. Where ν contains no big wire, so $S_1(|\nu|) = \nu$. Moreover

$$S_1(|\omega|) = S_1(|\gamma_b \circ \nu \circ \delta_a|) = S_1(|\gamma_b|) \circ S_1(|\nu|) \circ S_1(|\delta_a|) = S_1(|\nu|) = \nu.$$

So for an expanded diagram $\omega = \gamma_b \circ S_1(|\omega|) \circ \delta_a$.

It remains to show that any diagram can be rewritten in expanded form. We proceed by induction on the size of the biggest big wire in a bad situation.

If there is no such wire then we are already in expanded form. Else, we consider the biggest wires in a bad situation.

First we apply the expansion rule to remove all the biggest wires in situation (i,S), (d,S), (S,S), (S,g) and (S,o). If a wire is in situation (g,d) then the elimination rule can be applied and decreases strictly the size of the big wires in bad situations. If a wire is in situation (g,d) then the expansion rule can be applied and decreases strictly the size of the big wires in bad situations. If a wire is in situation (g,S) or (S,d) we apply the corresponding unfold equation. This decreases strictly the size of the big wires in bad situations. Thus all the wires in bad situations can be removed and replaced by wire with a strictly smaller size. Then by induction, all diagrams can be rewritten into expanded form.

□

From this result it follows that the scalable construction enjoys a universal property:

Lemma 20 (Universal property of $\mathcal{SL}$)**.** *The following diagram is a pull-back square:*

$$\begin{array}{ccc} \overset{\bullet}{\mathcal{SL}} & \xrightarrow{|\cdot|} & \overset{\bullet}{\mathcal{L}} \\ ! \downarrow & & \downarrow ! \\ \mathbf{1}_{\mathbb{N}^*} & \xrightarrow[|\cdot|]{} & \mathbf{1}_{\{*\}} \end{array}$$

Where $\mathbf{1}_{\mathbb{N}^*}$ is that terminal $\mathbb{N}^*$-coloured prop and $\mathbf{1}_{\{*\}}$ is the terminal monochromatic prop.

Proof. The diagram clearly commutes. Now let $f : \mathbf{K} \to \mathbf{1}_{\mathbb{N}^*}$ and $g : \mathbf{K} \to \overset{\bullet}{\mathcal{L}}$ be two functors such that $|\cdot| \circ f = ! \circ g$. We define a functor $h : \mathbf{K} \to \overset{\bullet}{\mathcal{SL}}$. Given a morphism $t \in \mathbf{K}[a,b]$ we take $h(t) = \gamma_{f(a)} \circ S_1(g(t)) \circ \delta_{f(b)}$. This is well defined since $|f(a)| = g(a)$. We have:

$$|h(t)| = |\gamma_{f(a)} \circ S_1(g(t)) \circ \delta_{f(b)}| = |S_1(g(t))| = g(t)$$

and

$$!(h(t)) = !(\gamma_{f(a)} \circ S_1(g(t)) \circ \delta_{f(b)}) = !_{f(a),f(b)} = f(t).$$

Now let $l : \mathbf{K} \to \overset{\bullet}{\mathcal{SL}}$ be another functor such that $|\cdot| \circ l = g$ and $! \circ l = f$. we have $l(t) \in \overset{\bullet}{\mathcal{SL}}\ [l(a), l(b)]$, the structure theorem gives us: $l(t) = \gamma_{l(a)} \circ S_1(|l(t)|) \circ \delta_{l(b)} = \Gamma_{f(a)} \circ S_1(g(t)) \circ \Delta_{f(b)}$. So $l = h$. The diagram is a pull-back square. □

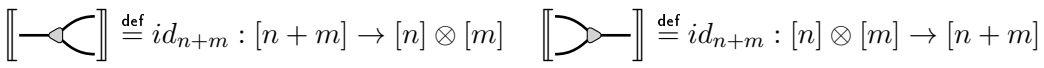
The universal property allows to lift interpretation functors.

Definition 70 (Scaled interpretation). *Given a monochromatic graphical language $\mathcal{L}$ with an interpretation $[\![\cdot]\!] : \overset{\bullet}{\mathcal{L}} \to \boldsymbol{P}$ we defined the* ***scaled interpretation*** $[\![\cdot]\!]_{\mathcal{S}} : \overset{\bullet}{\mathcal{SL}} \to \boldsymbol{P} \times_{\mathbf{1}_{\{*\}}} \mathbf{1}_{\mathbb{N}^*}$ *as* $[\![f]\!]_{\mathcal{S}} \overset{\text{def}}{=} (|f|, !_{a,b})$.

Given $f : a \to b$ in $\mathcal{SL}$, all the information in $[\![f]\!]_{\mathcal{S}}$ is contained in $|f|$ and the type of f. We will simplify the notation and usually write to describe the interpretation of a scaled diagram $f : a \to b$:

$$[\![f]\!] = |f| : a \to b$$

The dividers and gatherers have interpretation:



We can also directly lift the universality and completeness results.

Lemma 21. *$[\![\cdot]\!]_S$ is faithful (respectively full), iff $[\![\cdot]\!]$ is faithful (respectively full).*

Proof. By definition $\mathbf{P} \times_{\mathbf{1}_{\{*\}}} \mathbf{1}_{\mathbb{N}^*}$ is the pull-back of $! : \mathbf{P} \to \mathbf{1}_{\{*\}}$ and $|\cdot| : \mathbf{1}_{\mathbb{N}^*} \to \mathbf{1}_{\{*\}}$. Thus $[\![\cdot]\!]$ lifts to a unique functor $[\![\cdot]\!]_{\mathcal{S}} : \overset{\bullet}{\mathcal{SL}} \to \mathbf{P} \times_{\mathbf{1}_{\{*\}}} \mathbf{1}_{\mathbb{N}^*}$ satisfying $|\cdot| \circ [\![\cdot]\!]_{\mathcal{S}} = [\![\cdot]\!] \circ |\cdot|$. This functor is an $\mathbb{N}^*$-coloured prop morphism. Furthermore since $!$ and $|\cdot|$ are jointly monic then $[\![\cdot]\!]_{\mathcal{S}}$ is faithful iff $[\![\cdot]\!]$ is faithful. Iff $[\![\cdot]\!]$ is full then we can reach any map $(f, !_{a,b})$ in $\mathbf{P} \times_{\mathbf{1}_{\{*\}}} \mathbf{1}_{\mathbb{N}^*}$ by taking $[\![\gamma_b \circ S_1(|f|) \circ \delta_a]\!]_{\mathcal{S}}$ so $[\![\cdot]\!]_{\mathcal{S}}$. □

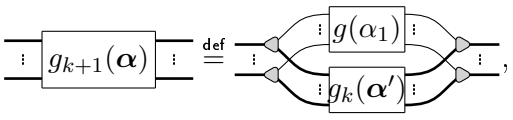
These results point out that the scalable construction is syntactic and has no impact on the semantics. In fact, we even have an equivalence of categories.

Lemma 22. *We have $\overset{\bullet}{\mathcal{L}} \simeq \overset{\bullet}{\mathcal{SL}}$ as symmetric strict monoidal categories.*

Proof. We consider the wire striper functor $|\cdot| : \overset{\bullet}{\mathcal{SL}} \to \overset{\bullet}{\mathcal{L}}$. It is clearly essentially surjective. It is also full and faithful since it induces a bijection between $\overset{\bullet}{\mathcal{SL}}\ [a, b]$ and $\overset{\bullet}{\mathcal{L}}\ [|a|, |b|]$ by the normal form theorem. So it is an equivalence of category and $\overset{\bullet}{\mathcal{L}} \simeq \overset{\bullet}{\mathcal{SL}}$. □

9.3 Arrows

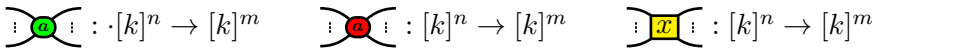
We now give more concrete examples of what scalable notations can do. The idea here is that we can use big wires to encapsulate large graphical structures with nice behaviour into large generators. In particular, we will be able to restate the results of graphical linear algebra presented in Chapter 7. We specify the scalable construction to the case of ZX-calculus and ZH-calculus. In this case we have spiders which are families of monochromatic generators $(g(\alpha) : \mathbf{a} \to \mathbf{b})_{\alpha \in A}$ indexed by some parameter $\alpha \in A$. In such a situation, we can index the scaled version by an element $\boldsymbol{\alpha} \in A^k$. This is defined inductively by $g_1(\alpha) \overset{\text{def}}{=} g(\alpha)$ and:



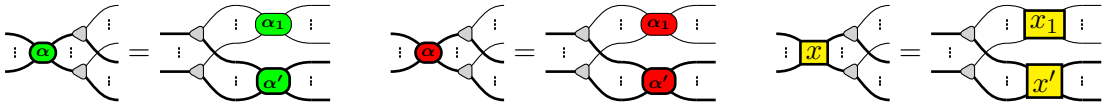
with $\boldsymbol{\alpha} = (\alpha_1, \boldsymbol{\alpha}') \in A^{k+1}$ and $\boldsymbol{\alpha}' \in A^k$.

The associated scaled rule, if any, should be *a priori* defined in the same way. For example, in the ZX-calculus, the phases α and β of two spiders add up when they fuse, so the lists of phases $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ add up pointwise when scaled spiders fuse.

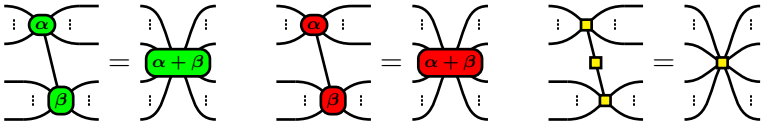
In what follows, we will use a mix of ZX-calculus and ZH-calculus. The green and red families of spiders are indexed by phase vectors in $\left(\mathbb{R}/2\pi\mathbb{Z}\right)^k$, by convention the phase is 0 if not given. The yellow family of harvestmen is indexed by complex vectors, by convention the phase is a vector of -1 if not given. They are respectively depicted:



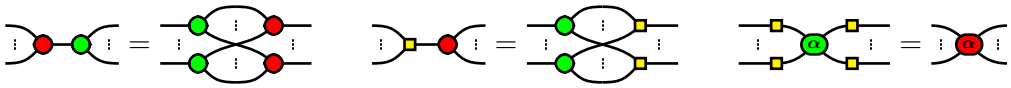
Denoting α_1 (resp. x_1) the head of the phase vector α (resp. complex vector x) and α' (resp. x') its tail, the arachnids interact with dividers and gatherers with:



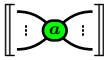
All of them are flexsymmetric and satisfy fusion rules:



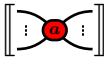
Note we only define fusion of yellow boxes indexed by phase -1. We recall that the three arachnids interact with each other in the following way:



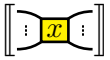
The green and red families of spiders and the yellow family of harvestmen indexed phase vectors are depicted respectively:



$$\stackrel{\text{def}}{=} 2^{k\frac{n+m-2}{4}} \sum_{x \in \mathbf{2}^k} e^{i\pi(x \cdot \boldsymbol{\alpha})} |x\rangle^{\otimes n} \langle x|^{\otimes m} : [k]^n \to [k]^m$$



$$\stackrel{\text{def}}{=} 2^{k\frac{2-n-m}{4}} \sum_{x_i \in \mathbf{2}^k} \prod_{j=1}^{k} \frac{1+e^{i\pi\left(\alpha_j + \sum_{i=1}^{n+m} x_{i,j}\right)}}{2} |x_1 \cdots x_n\rangle\langle x_{n+1} \cdots x_{n+m}| : [k]^n \to [k]^m$$



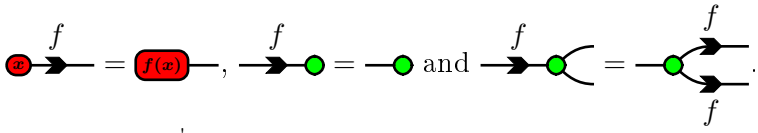
$$\stackrel{\text{def}}{=} 2^{-k\frac{n+m}{4}} \sum_{y_i \in \mathbf{2}^k} \prod_{j=1}^{k} x_j^{\bigwedge_{i=1}^{n+m} y_i} |y_1 \cdots y_n\rangle\langle y_{n+1} \cdots y_{n+m}| : [k]^n \to [k]^m$$

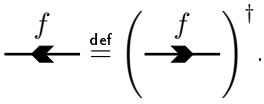
Notice that the vectors for the green and red spiders have coefficient in $2\pi\mathbb{R}/\mathbb{R}$ and then the addition of phase vectors is done component-wise modulo 2π. However, we will by a slight abuse of notation use $\{0, 1\}$ vectors to denote $\{0, \pi\}$ vectors. So the state $|x\rangle$ is represented up to a scalar as a red node indexed by the phase vector x.

9.3.1 Function arrows

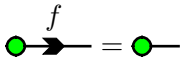
Given a boolean function $f : \mathbf{2}^n \to \mathbf{2}^m$ we naturally turn it into a linear map $\mathbb{C}^{\mathbf{2}^n} \to \mathbb{C}^{\mathbf{2}^m}$ and define a **function arrow**.

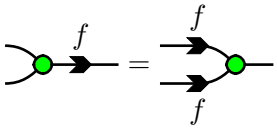
$$\left[\!\!\left[\; f \; \right]\!\!\right] = |x\rangle \mapsto 2^{\frac{m-n}{4}} |f(x)\rangle$$

Any function arrow satisfies: 

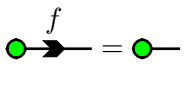
Furthermore we define: 

Lemma 23. *Given a function arrow f:*

$$f \text{ is } \textbf{\textit{balanced}} \overset{def}{\Leftrightarrow} \forall x, y \in 2^m \; |f^{-1}(\{x\})| = |f^{-1}(\{y\})| \quad \Leftrightarrow$$ 

$$f \text{ is } \textbf{\textit{injective}} \overset{def}{\Leftrightarrow} \forall x, y \in 2^n \; (f(x) = f(y) \Rightarrow x = y) \quad \Leftrightarrow$$ 

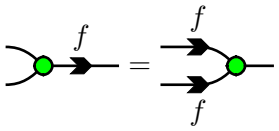
Proof.



$$\Leftrightarrow \quad \left[\!\!\left[\; f \; \right]\!\!\right] = [\![\;]\!]$$

$$\Leftrightarrow \quad \sum_{x \in \mathbf{2}^n} |f(x)\rangle = \sum_{y \in \mathbf{2}^m} 2^{\frac{n-m}{2}} |y\rangle$$

$$\Leftrightarrow \quad \forall x, y \in 2^m \; |f^{-1}(\{x\})| = |f^{-1}(\{y\})|$$



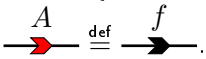
$$\Leftrightarrow \quad \left[\!\!\left[\; f \; \right]\!\!\right] = \left[\!\!\left[\; f \; \right]\!\!\right]$$

$$\Leftrightarrow \quad \forall x, y \in 2^n \; \delta_{x=y} |f(x)\rangle = \delta_{f(x)=f(y)} |f(x)\rangle$$

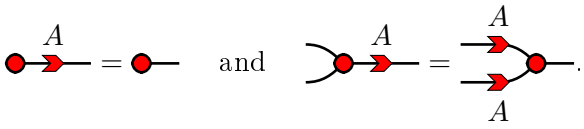
$$\Leftrightarrow \quad \forall x, y \in 2^n \; (f(x) = f(y) \Rightarrow x = y)$$

□

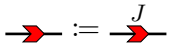
9.3.2 Red arrows

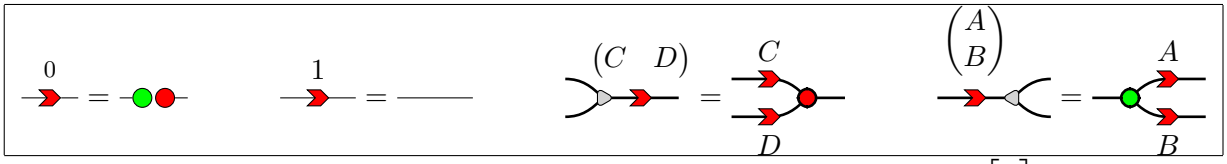
A function $f : 2^n \to 2^m$ can be seen as a map $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$, if this map is $\mathbb{F}_2$-linear then it can be described by a matrix $A \in \mathcal{M}_{m \times n}(\mathbb{F}_2)$. The **red matrix arrows** indexed by A is then defined by 

. Those arrows have been extensively studied in [104] and [8]. Those red matrix arrows have interesting properties, in fact they correspond to the embedding of the matrices of Chapter 7 into the ZX calculus. Being $\mathbb{F}_2$-linear translates to:



They interact with the dividers and gatherers as:

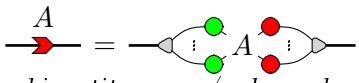
We take the convention to omit the all-ones matrices:  where $\forall i, j, J_{i,j} = 1$.
We can axiomatise those matrices as:



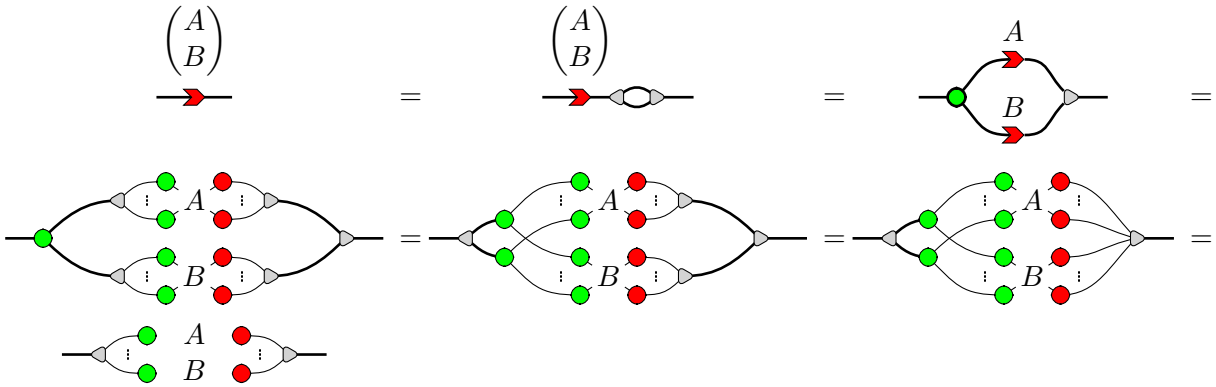
Axioms for matrices, where $A \in \mathbb{F}_2^{a\times n}$, $B \in \mathbb{F}_2^{b\times n}$, $C \in \mathbb{F}_2^{m\times c}$ and $D \in \mathbb{F}_2^{m\times d}$. $\begin{bmatrix} A \\ B \end{bmatrix}$ and $[CD]$ are block matrices.

From those axioms, all properties of red matrix arrows follow.

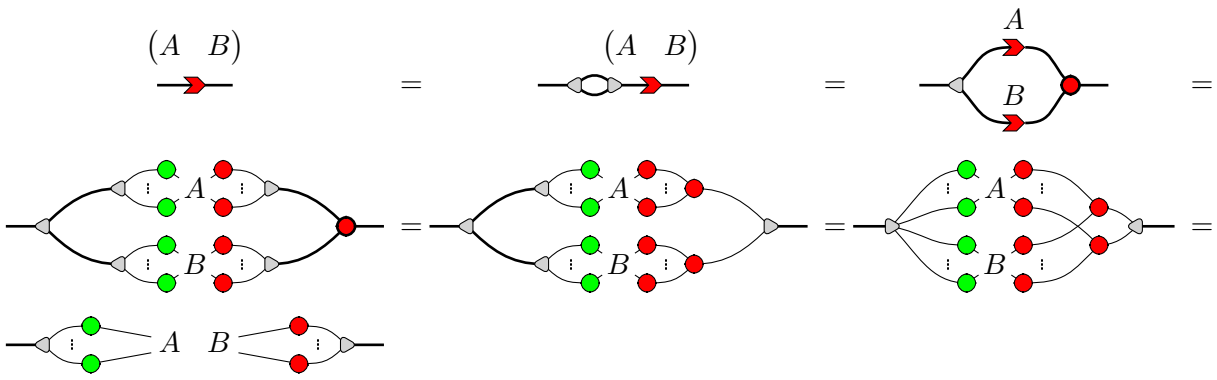
By universality of the scalable construction, we know that the matrix arrows are already expressible as diagrams. The matrix generator is actually a compact representation of a green/red bipartite graphs whose bi-adjacency matrix is A:

Lemma 24. *For any $A \in \mathbb{F}_2^{m\times n}$,*  *where A represents in the RHS diagram the adjacency matrix of the bipartite green/red graph.*

Proof. By induction on the size $m \times n$ of A. If $n = m = 1$, the result is exactly the zero and one axioms. if $m > 1$:



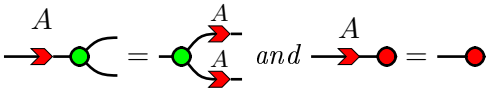
if $n > 1$:



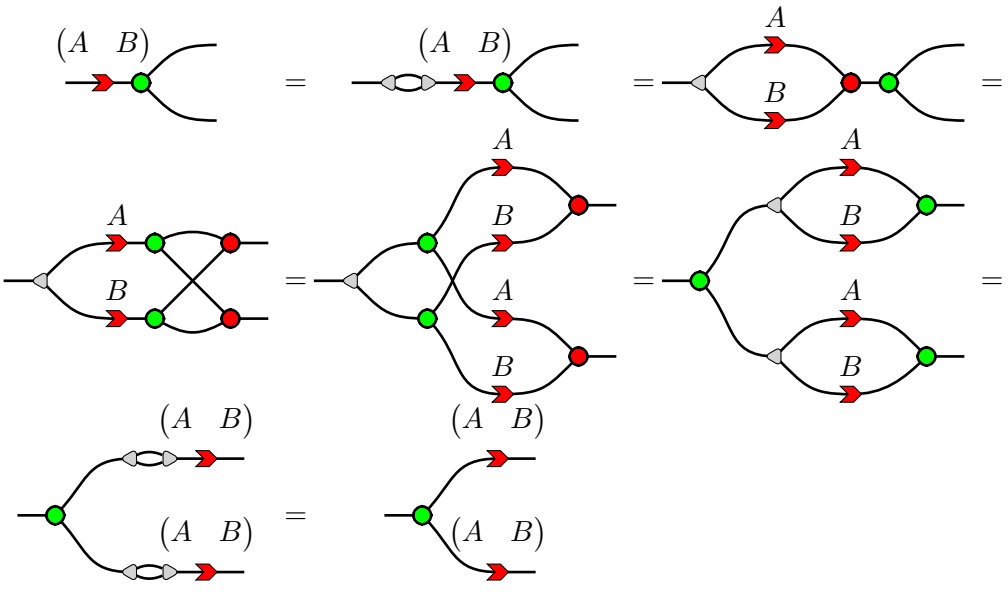
□

Remark 1. *All those properties are in fact direct translations of the graphical linear algebra of Chapter 7. Notice however that our axiomatisation of the matrices strongly relies on their interaction with the divider and the gatherer, which are not present in [19].*

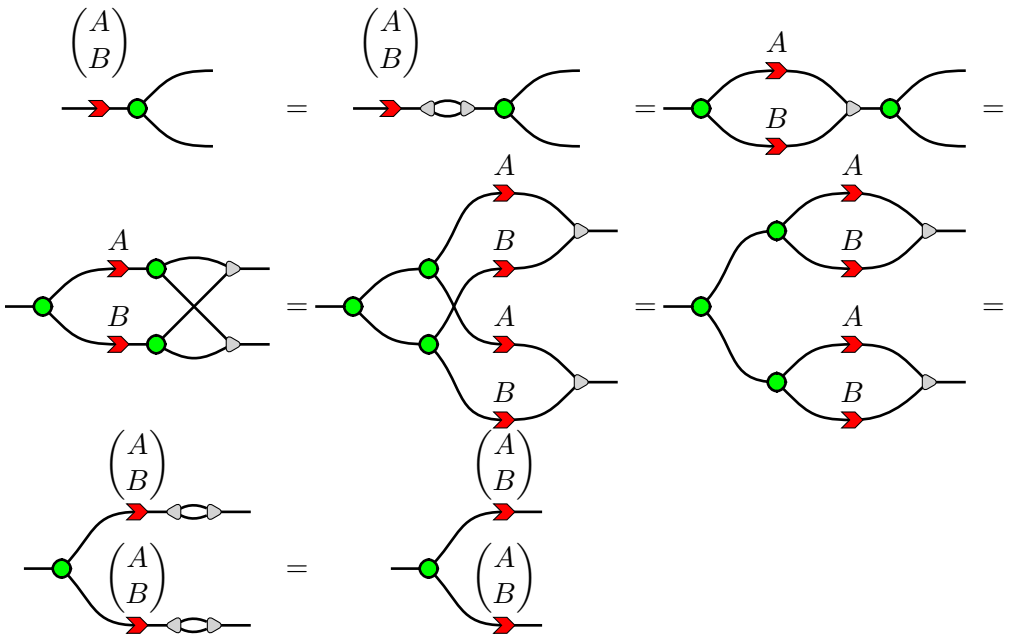
As said before red matrix arrows are copied and erased by green nodes.

Lemma 25. *For any* $A \in \mathbb{F}_2^{m\times n}$, 

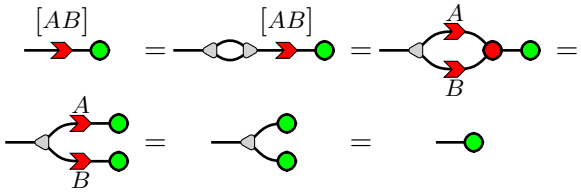
Proof. We start with the copy. By induction on the size $m \times n$ of A. If $n = m = 1$ this directly follows from the rules of ZX-calculus. If $n > 1$:



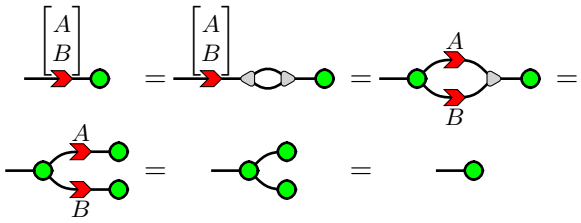
If $m > 1$:



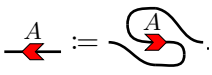
Then we prove erasing. By induction on the size $m \times n$ of A. If $n = m = 1$ this directly follows from the rules of ZX-calculus. If $n > 1$:

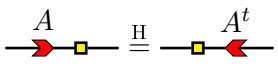


If $m > 1$:



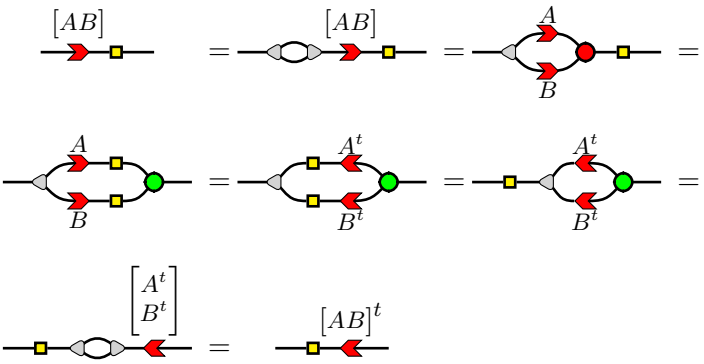
□

We define backward matrices as follows: 
.

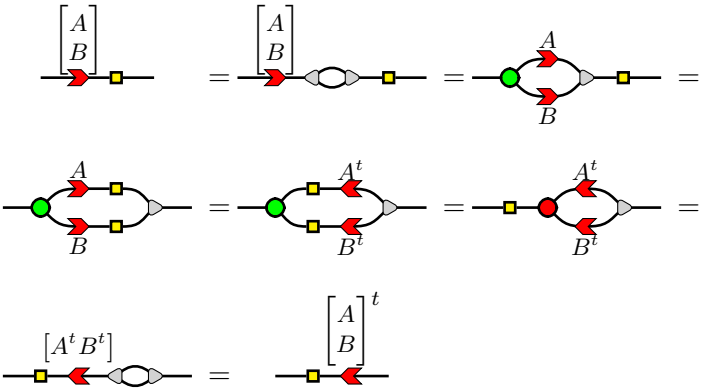
Lemma 26. $\forall A \in \mathbb{F}_2^{m \times n}$, 

where A^t is the transpose of A.

Proof. By induction on the size $m \times n$ of A. If $n = m = 1$ this directly follows from the rules of ZX-calculus. If $n > 1$:

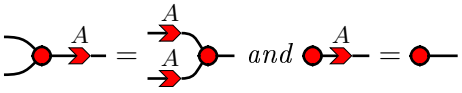


If $m > 1$:



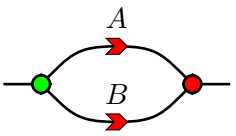
□

As a consequence, conjugating by Hadamard ( ) reverses the orientation and transposes the matrix. Since conjugating by Hadamard colour-swaps the spiders and preserves the other generators of the language, one can derive from any equation a new one (up to scalars) which consists in colour-swapping the spiders, transposing the matrices, and then changing their orientation. For instance, the following Lemma gives that matrices are co-copied and co-erased by red nodes:

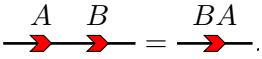
Lemma 27. *For any* $A \in \mathbb{F}_2^{m\times n}$,  *and*

Notice that if the colour-swap symmetry is already present in Chapter 7, the Hadamard gate here really internalizes it directly into the graphical language.

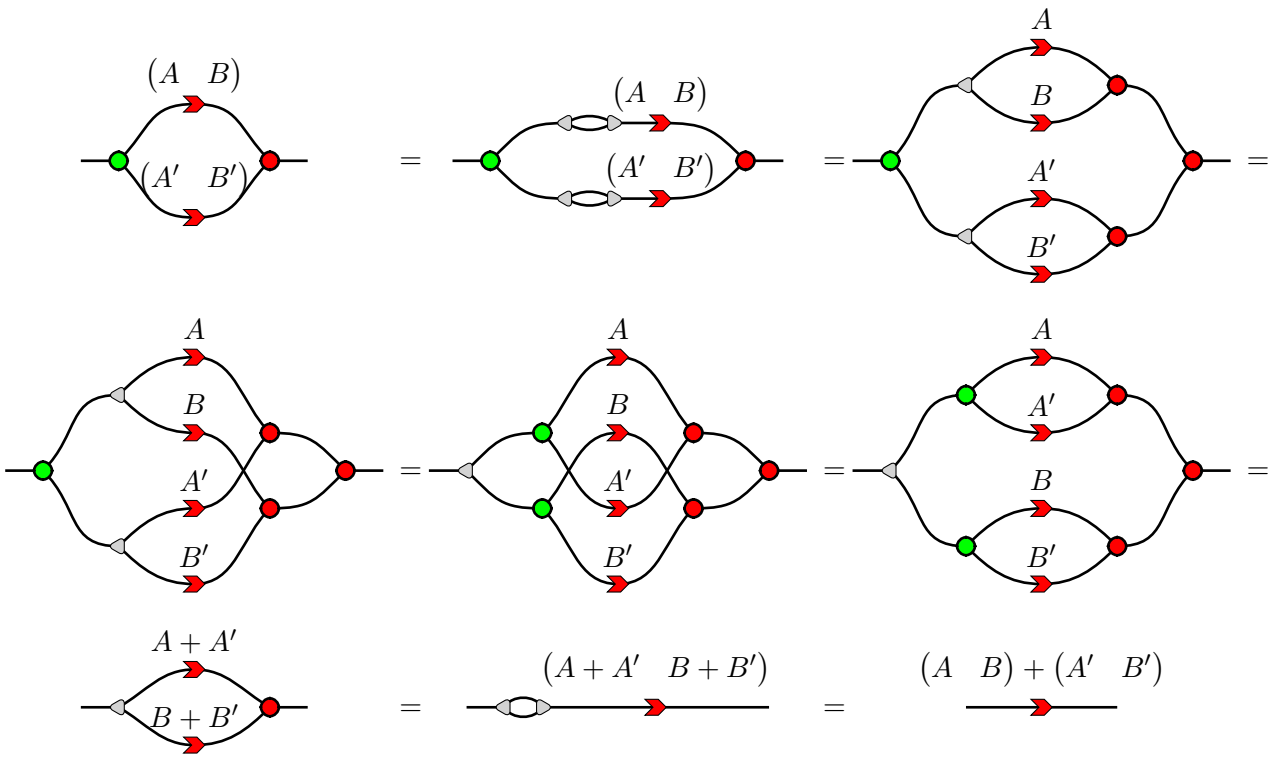
Basic matrix operations like addition and multiplication (in $\mathbb{F}_2$) can be implemented graphically:

Lemma 28. *For any* $A, B \in \mathbb{F}_2^{m\times n}$, *and any* $C \in \mathbb{F}_2^{k\times m}$, 

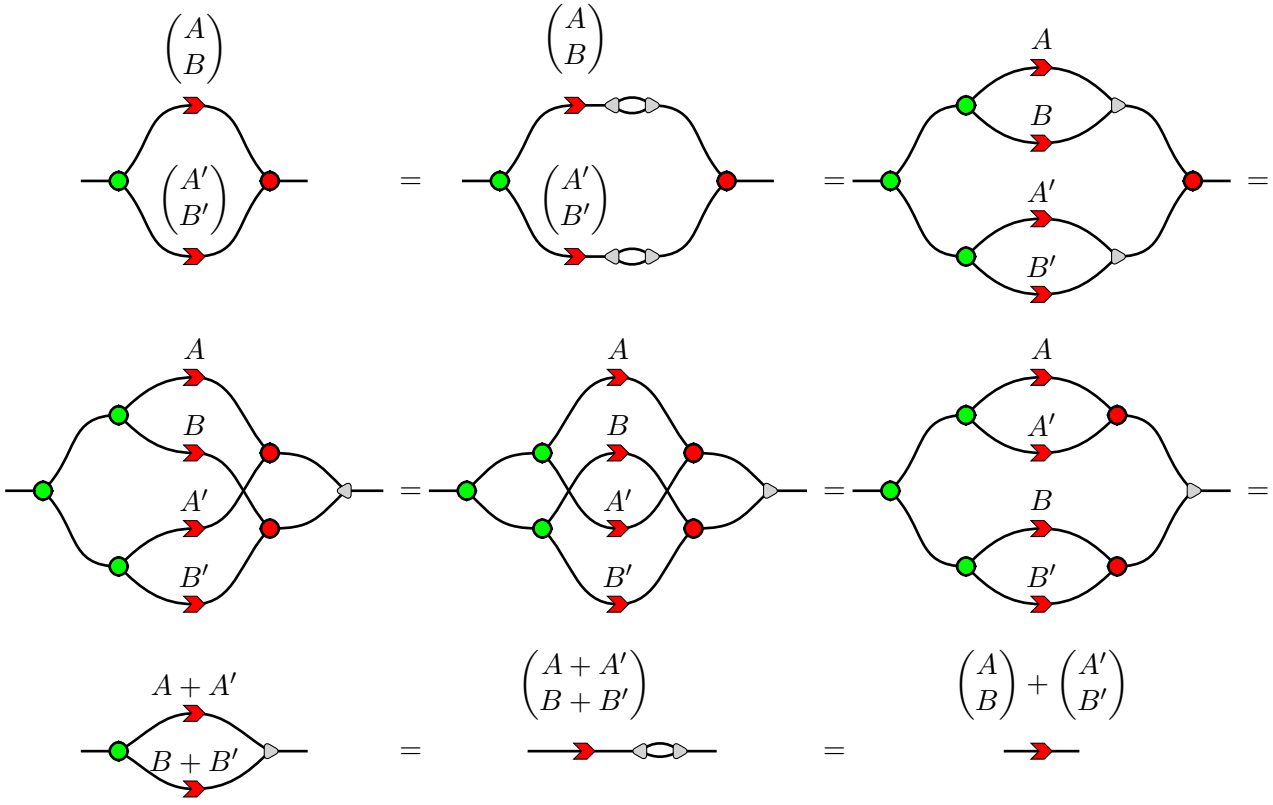
$= A + B$ *and*



Proof. We start with addition. By induction on the size $m \times n$ of A and B. If $n = m = 1$, this is the Hopf rule of ZX-calculus. If $n > 1$:

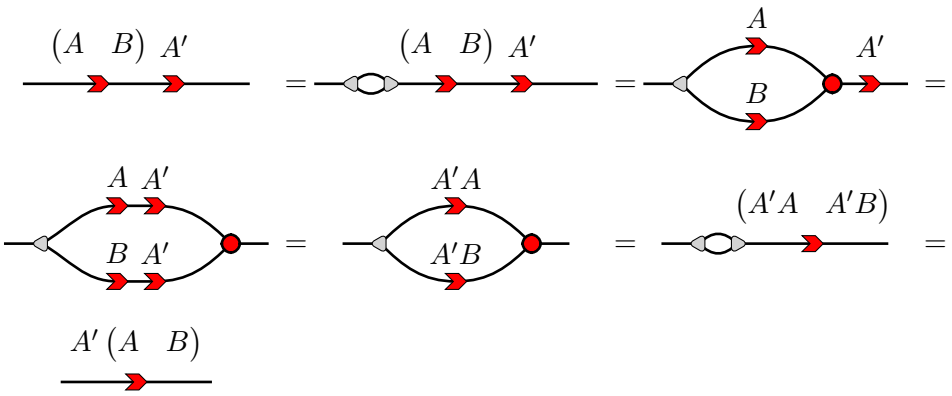


If $m > 1$:

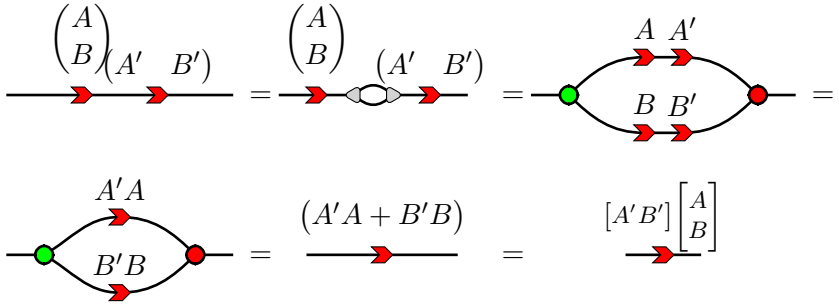


Now we prove multiplication. By induction on the size $b \times a$ of A and $c \times b$ of B.

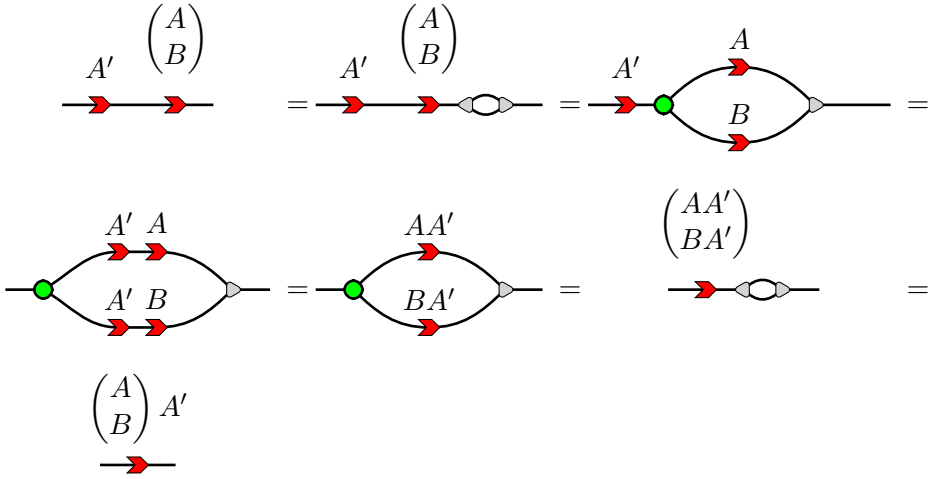
If $a = b = c = 1$ this follows from the zero and one axioms and the scalar rule. If $a > 1$:



If $b > 1$:



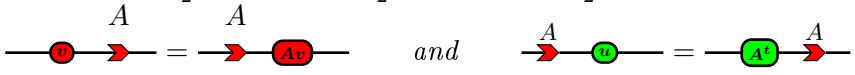
If $c > 1$:



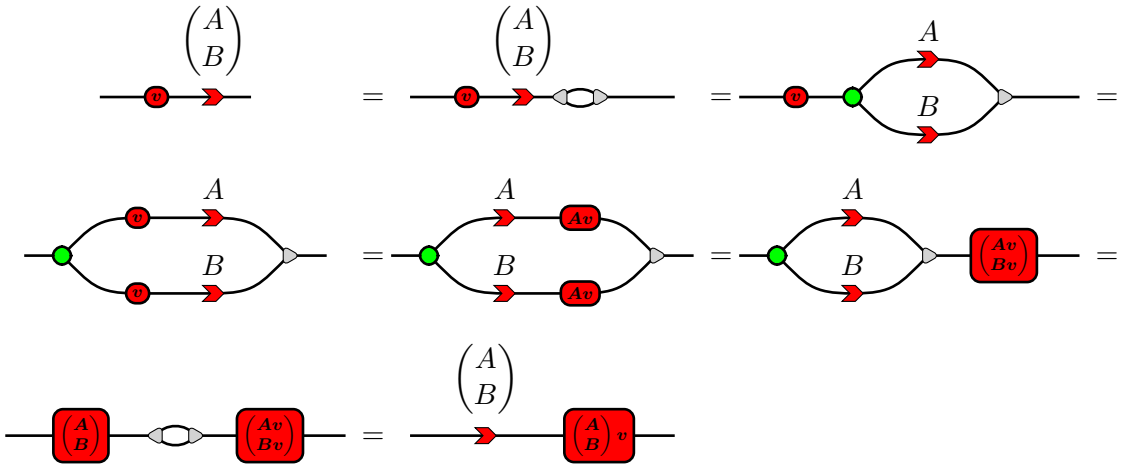
□

Whereas all the previous properties about matrices are angle-free, some spiders whose angles are multiples of π can be pushed through matrices as follows:

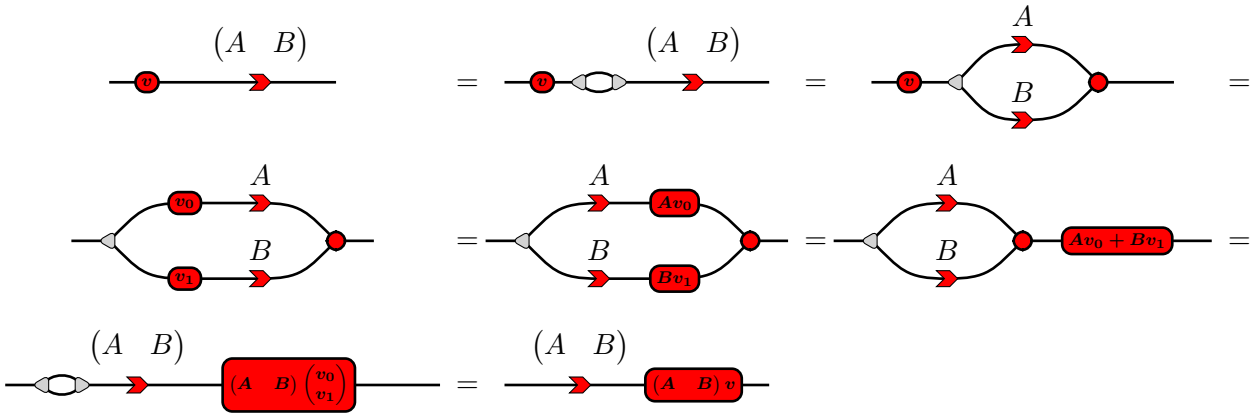
Lemma 29. *For any $A \in \mathbb{F}_2^{m\times n}$, any $v \in \mathbb{F}_2^n$ and any $u \in \mathbb{F}_2^m$,*



Proof. By induction on the size $m \times n$ of A. If $n = m = 1$ they are two possibilities: if $A = 1$ then this is trivial, and if $A = 0$ then the π is erased by the green node and the equation also true. If $m > 1$:



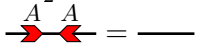
if $n > 1$:

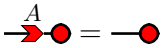
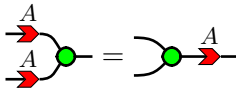


□

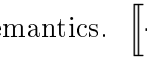
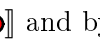
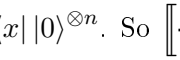
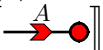
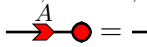
Injective matrices enjoy some specific properties:

Lemma 30. *For any $A \in \mathbb{F}_2^{m\times n}$, the following properties are equivalent:*

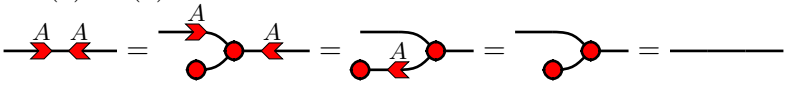
(1) *A is injective.* (3) 

(2)  (4) 

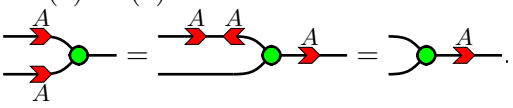
Proof. We show this by circular implications:

(1) ⇒ (2): We use the semantics. $\left[\!\!\left[\text{  } \right]\!\!\right] = |x\rangle \mapsto 2^{\frac{n}{4}} \langle Ax| \, |0\rangle^{\otimes m}$. The basis being orthonormal: $\langle Ax| \, |0\rangle^{\otimes m} = \delta_{Ax,0}$ but since A is injective $\delta_{Ax,0} = \delta_{x,0}$. Besides $\left[\!\!\left[\text{  } \right]\!\!\right] = |x\rangle \mapsto 2^{\frac{n}{4}} \langle x| \, |0\rangle^{\otimes n}$. So $\left[\!\!\left[\text{  } \right]\!\!\right] = \left[\!\!\left[\text{  } \right]\!\!\right]$ and by completeness:  .

(2) ⇒ (3):

 .

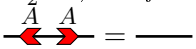
(4) ⇒ (5):

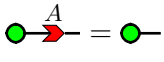
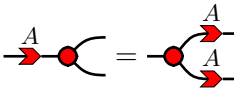


(5) ⇒ (1): We come back to the semantics: $\left[\!\!\left[\right]\!\!\right] = |x\rangle\,|y\rangle \mapsto \delta_{Ax,Ay}\,|Ax\rangle$ and $\left[\!\!\left[\right]\!\!\right] = |x\rangle\,|y\rangle \mapsto \delta_{x,y}\,|Ax\rangle$. So for all x and y $\delta_{Ax,Ay} = \delta_{x,y}$, in other words A is injective. □

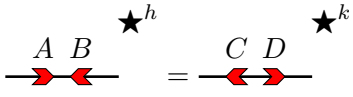
By Hadamard conjugation, we obtain some dual properties for surjective matrices:

Lemma 31. *For any $A \in \mathbb{F}_2^{m\times n}$, the following properties are equivalent:*

(1) *A is surjective.* (3) 

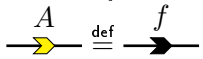
(2)  (4) 

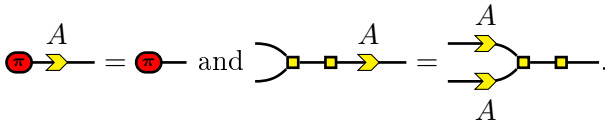
In fact, those last properties of red matrix arrows can be summed up into one meta rule, the translation of the matrix interaction rule of Chapter 7:

 $\Leftrightarrow \quad Im\begin{pmatrix} C \\ D \end{pmatrix} = Ker\begin{pmatrix} A & B \end{pmatrix}$

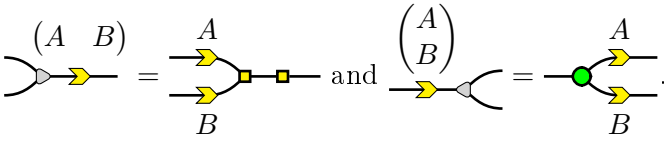
With $k \stackrel{\text{def}}{=} dim\left(Ker\begin{pmatrix} C \\ D \end{pmatrix}\right)$ and $h \stackrel{\text{def}}{=} dim\left(coKer\begin{pmatrix} A & B \end{pmatrix}\right)$.

9.3.3 Yellow arrows

Here we investigate the translation of graphical linear algebra into ZH calculus that I presented briefly at the end of Chapter 7. As noted in [8], the possibility to index arrows by matrices is linked to a bi-algebra structure. If the red/green bi-algebra leads to red matrix arrows, the yellow/green bi-algebra gives us another family of matrix arrows over the boolean semi-ring $\mathbb{B} \stackrel{\text{def}}{=} (\{0,1\}, \wedge, \vee)$. A function $f : 2^n \to 2^m$ can be seen as a map $f : \mathbb{B}^n \to \mathbb{B}^m$, if this map is a homomorphism of $\mathbb{B}$-semi module, that is $f(a \wedge b) = f(a) \wedge f(b)$ and $f(1) = 1$, then it can be described by a matrix $A \in \mathcal{M}_{m\times n}(\mathbb{B})$. The **yellow matrix arrow** indexed by A is then defined by  . Being a homomorphism of $\mathbb{B}$-semi module translates to:



Yellow matrix have less properties than red ones. However we still have:



Chapter 10

Drawing quantum computing

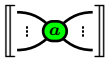
If you take nothing else from this blog: quantum computers won't solve hard problems instantly by just trying all solutions in parallel.

Scott Aaronson in [105]

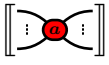
Now that we have introduced the discard construction in Chapter 8 and the scalable notations in Chapter 9 it's time to put everything together and try to apply those abstract constructions to concrete examples. The idea in this chapter is to test the scalable notations on various quantum processes. This reveals at the same time the strengths and weaknesses of the notations. From those examples, we can see where the notations work well and where more developments have to be done.

In this chapter, we will use a mixture of the discard ZX-calculus and discard ZH-calculus together with scalable notations. So we have at our disposal red and yellow matrix arrows.

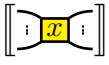
Setting $[\![\bigstar]\!] \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}}$. The spiders and harvestmen of type $[k]^n \to [k]^m$ have interpretations:



$$\stackrel{\text{def}}{=} \rho \mapsto V\rho V^\dagger \text{ with } V \stackrel{\text{def}}{=} 2^{k\frac{n+m-2}{4}} \sum_{x\in\mathbf{2}^k} e^{i(x\cdot a)} \, |x\rangle^{\otimes n}\langle x|^{\otimes m}$$

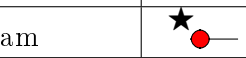
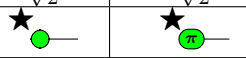


$$\stackrel{\text{def}}{=} \rho \mapsto V\rho V^\dagger \text{ with } V \stackrel{\text{def}}{=} 2^{k\frac{2-n-m}{4}} \sum_{x_i\in\mathbf{2}^k} \prod_{j=1}^{k} \frac{1+e^{\,i\left(a_j+\pi \sum_{i=1}^{n+m} x_{i,j}\right)}}{2} \, |x_1\cdots x_n\rangle\langle x_{n+1}\cdots x_{n+m}|$$



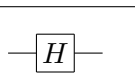
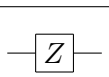
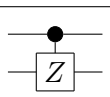
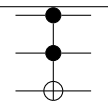
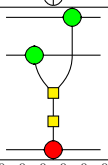
$$\stackrel{\text{def}}{=} \rho \mapsto V\rho V^\dagger \text{ with } V \stackrel{\text{def}}{=} 2^{-k\frac{n+m}{4}} \sum_{x_i\in\mathbf{2}^k} \prod_{j=1}^{k} x^{\prod_{i=1}^{n+m} x_{i,j}} \, |x_1\cdots x_n\rangle\langle x_{n+1}\cdots x_{n+m}|$$

Where the x_i are binary words of size k and $x_{i,j}$ is the j-th bit of x_i. By convention, if the phase is not given it is 0 for red and green spiders and -1 for yellow ones. Usually, quantum algorithms are presented as quantum circuits built from elementary quantum gates. Our language is expressive enough to represent all of them. The main idea is that our generators decompose quantum gates into more fundamental parts which equational theory is better understood. The most common states are represented:

State	$\|0\rangle$	$\|1\rangle$	$\frac{\|0\rangle+\|1\rangle}{\sqrt{2}}$	$\frac{\|0\rangle-\|1\rangle}{\sqrt{2}}$
Diagram				
Density Matrix	$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}$

We will also use the mirror image of those states corresponding to effects. By doing so we will obtain post-selected circuits and we will be able to compute amplitudes.

The following table provides the representation of the most common quantum gates. They are all pure maps, i.e., operators of the form: $\rho \mapsto V\rho V^\dagger$. We just give the corresponding matrix V.

Name	H	Not	Z	Swap	C-Not	C-Z	Toffoli
Gate							
Diagram							
V	$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$

We will also use the **discard map** and its transpose which is the non normalized **completely mixed state**:

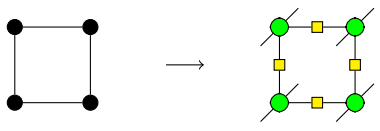
$$[\![\dashv \vdash]\!] \stackrel{\text{def}}{=} \rho \mapsto \frac{1}{\sqrt{2^n}} Tr(\rho) \qquad [\![\vdash \dashv]\!] \stackrel{\text{def}}{=} \rho \mapsto \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbf{2}} |x\rangle \langle x|$$

We recall that the first corresponds to discarding data and the second is a uniform probabilistic mixture of states.

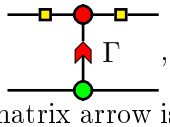
10.1 Graph states

Graph-states were among the first example of application of the scalable notation in [104]. We provide here a much nicer representation. A C-Z gate has interpretation $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$ and graphical representation  . A composition of C-Z gates on n qubits is called a **graph-state operator**. In fact given a graph (V, E) with $|V| = n$, the associated graph operator $G : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ is defined as the composition of the C-Z gates on the qubits i and j for each $(i, j) \in E$. Usually we define the graph-state $|G\rangle \stackrel{\text{def}}{=} G\,|+\rangle^{\otimes n}$ instead of the graph operator. See [106] for more informations on graph-states.

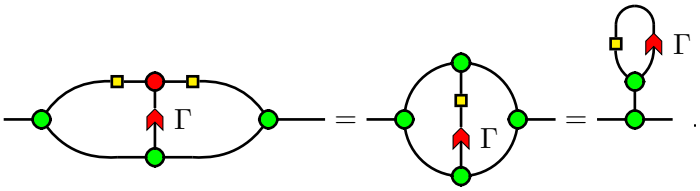
Example 16. *Taking the square graph the corresponding graph state operator is represented as:*



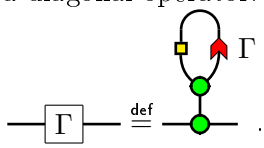
10.1.1 Graphical representation

If G is a bipartite graph (V_0, V_1, E) with $|V_0| = a$ and $|V_1| = b$ then we can write the graph operator 

, where Γ is the bi-adjacency matrix of G defined by $\Gamma_{i,j} = \delta_{(i,j)\in E}$. Here the red matrix arrow is applying C-Nots that Hadamard gates turn into C-Z as expected. Given a non-bipartite graphs (V, E), we build a bipartite graph (V, V, E'). We fix an ordering of the vertices in V and define: $E' \stackrel{\text{def}}{=} \{(i,j), i, j \in V, i > j \text{ and } (i,j) \in E\}$. The ordering ensures that each edge appears only once. Then the bi-adjacency matrix Γ is upper triangular and satisfies $\Gamma + \Gamma^t = A$, the adjacency matrix of G. Thus we call Γ the **half adjacency matrix** of G. We then fuse together the copies of the same vertex in the bipartite graph operator with green nodes:



We recognize the typical form of a diagonal operator.

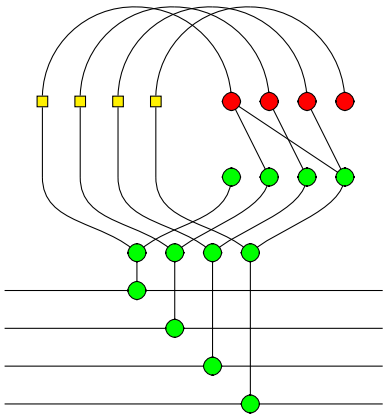
We denote graph state operators 

.

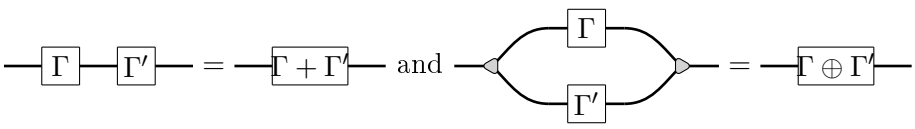
Example 17. *A half adjacency matrix for the square graph is:*

$$\Gamma \stackrel{\text{def}}{=} \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The corresponding diagonal operator is:



We have:



We can now provide graphical versions of the properties of graph operators.

10.1.2 Stabilizer properties

The graph state corresponding to the graph (V, E) can also be characterize as the unique state preserved by all the operators $\mathcal{V}_i$, defined for each $i \in V$ as:

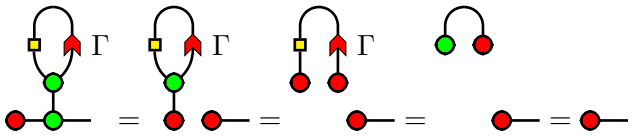
$$\mathcal{V}_i \stackrel{\text{def}}{=} X_i \circ Z_{(\Gamma+\Gamma^t)i}$$

$G\,|x\rangle = \sum^{max} |x\rangle$ and for each vertex $i \in V$, $G\,|i\rangle = X_i \circ Z_{(\Gamma+\Gamma^t)i} \circ G$

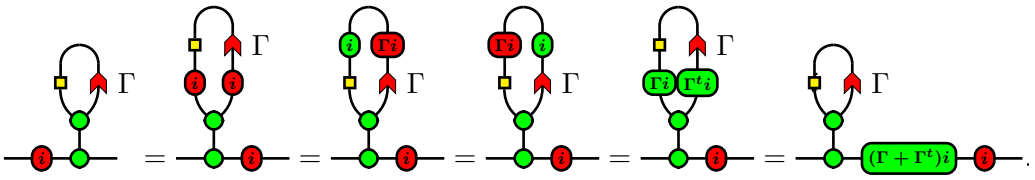
This follows directly from the two following properties of graph operators.

Lemma 32. *The graph state operator corresponding to the graph (V, E) satisfies $G\,|0\rangle = |0\rangle$ and $G \circ X_i = X_i \circ \mathcal{V}_i \circ G$.*

Proof. We have:



and

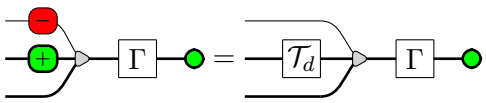


□

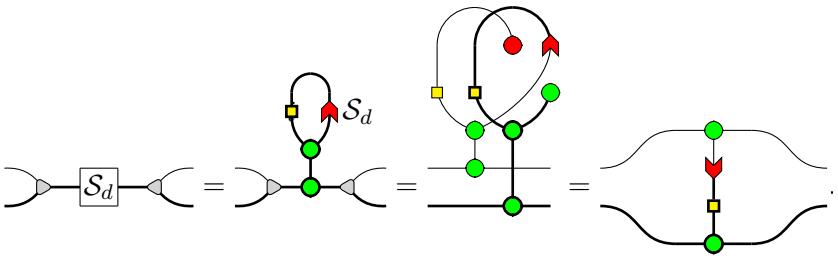
This can be seen as a second way to ensure that our construction correctly implements graph states.

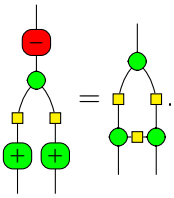
10.1.3 Local complementation

Graph states can be modified by applying phase gates locally on the vertices. More precisely we have $X_u(-\frac{\pi}{2}) \circ Z_{N_u}(\frac{\pi}{2}) \circ G\,|+\rangle = (G * u)\,|+\rangle$ where $G * u$ is the graph G locally complemented in u. Our proof is a scalable reformulation in scalable notations of the one of [24]. We denote $\mathcal{T}_d$ the half adjacency matrix of the complete graph on d vertices, which is the strict upper triangular $d \times d$ matrix full of ones. We assume that the vertices in G are ordered such that we first have u, then the neighbourhood of u, and then the other vertices. Denoting $+$ the $\frac{\pi}{2}$ phase and $-$ the $-\frac{\pi}{2}$ phase we then want to prove:

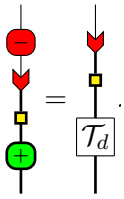
Lemma 33. 

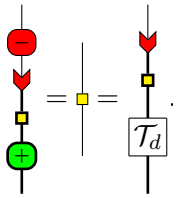
Proof. First we consider the case of star graphs. In this situation the half adjacency matrix that we call $\mathcal{S}_d$ has the form: $\begin{pmatrix} 0 & \\ 1 & \\ \vdots & (0) \\ 1 & \end{pmatrix}$. We have:

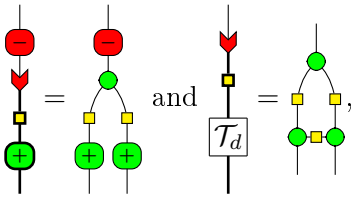


Admitting the triangle lemma proved in [24]:  .

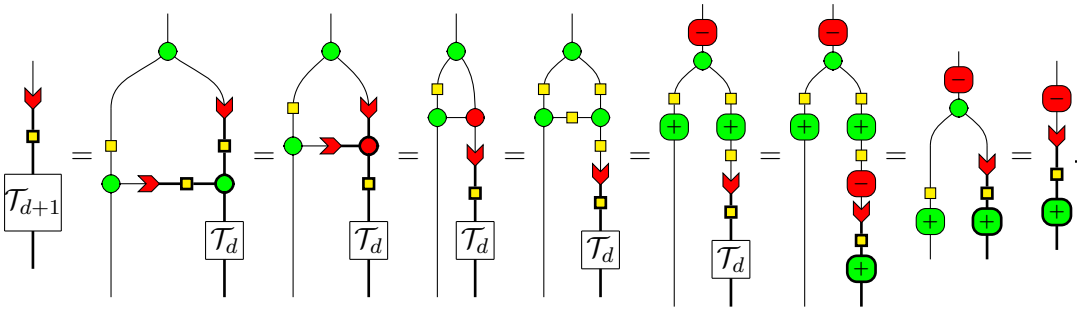
We prove the generalize version:

Lemma 34. 

Proof. For $n = 1$, we have 

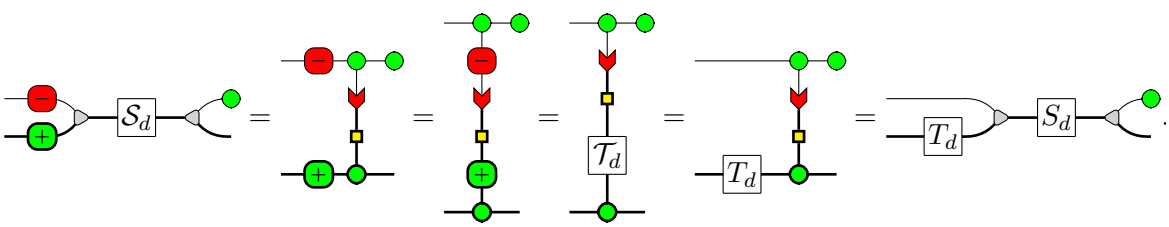
. For $n = 2$, we have 

, this is exactly the triangle lemma. For $n \geq 3$:

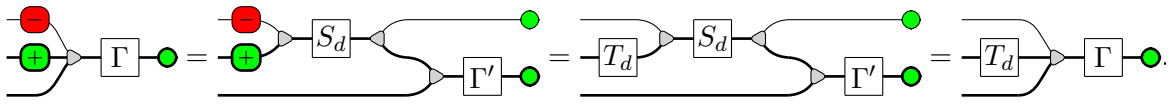


□

Using the generalized triangle lemma we have:



And finally:

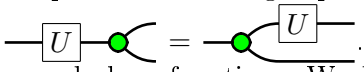


□

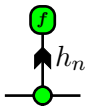
10.2 Diagonal gates

We now take some hindsight and consider diagonal gates that more general than graph states. The main point of this section is to reproduce with scalable notations some results that have been first proved without them or using bang boxes in [26] and [25].

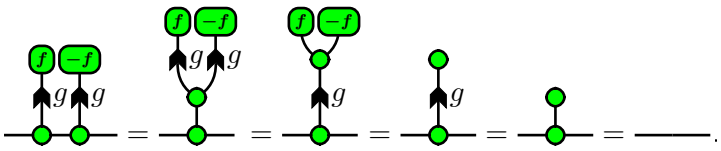
10.2.1 Definition

A **diagonal gate** is a unitary map that is diagonal in the computational basis. In other words, it is a linear map $\mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ such that for each $x \in \mathbf{2}^n$, $|x\rangle$ are eigenvectors with module one eigenvalue. The diagonal gates on n qubits form an Abelian group isomorphic to $\mathbb{U}^{2^n}$. A **phase function** is a semi-boolean function $f : 2^n \to \mathbb{R}$. To each phase function we can associate a diagonal gate $e^{i\pi f} : [n] \to [n]$ defined by $e^{i\pi f} : |x\rangle \mapsto e^{i\pi f(x)} |x\rangle$. The correspondence is not one to one, $e^{i\pi f} = e^{i\pi g}$ if and only if there is a $k \in \mathbb{N}$ such that $f = g + 2k$. We have $e^{i\pi(f+g)} = e^{i\pi f} e^{i\pi g}$. Graphically, being a diagonal gate is equivalent to being a phase of the scaled green spider. That is to be a unitary satisfying : 

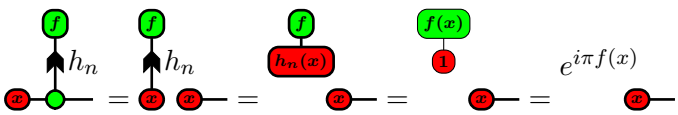
. We can represent graphically any diagonal gate with function arrows and phase functions. We define the **set function** $h_n : \mathbf{2}^n \to \mathbf{2}^{\mathbf{2}^n}$ as: $\forall x, s \in \mathbf{2}^n, h(x)_s \stackrel{\text{def}}{=} \delta_{x=s}$. We can also define h_n inductively by $h_0() = 1$ and $h_{n+1}(x_0 x') = \begin{cases} h_n(x')0\cdots 0 \text{ if } x_0 = 0 \\ 0\cdots 0 h_n(x') \text{ if } x_0 = 1 \end{cases}$.

Denoting f the vector $(f(x))_{x\in 2^n}$, $e^{i\pi f}$ is pictured: 

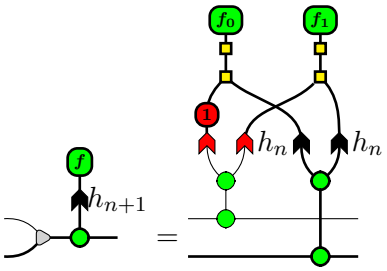
. It has clearly the form of a phase. Moreover, any map of this form is unitary:



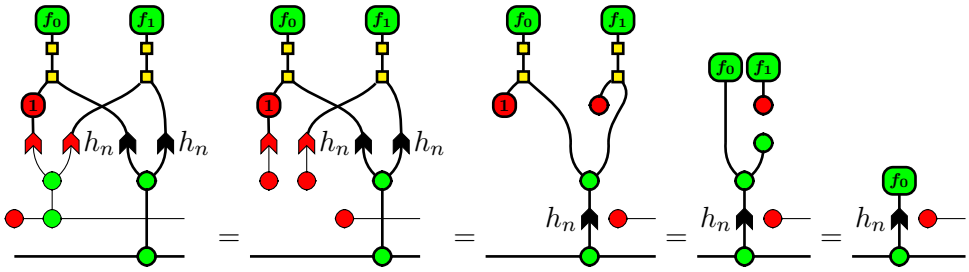
We can check it is a correct representation of $e^{2\pi f}$:



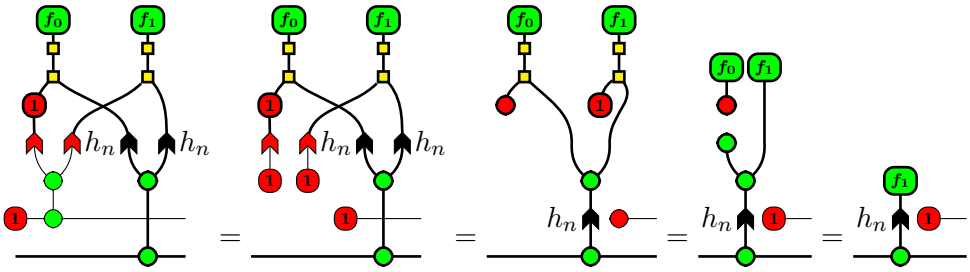
Defining $f_0(x) \stackrel{\text{def}}{=} f(0x)$ and $f_1(x) \stackrel{\text{def}}{=} f(1x)$ we have:



We can see that:



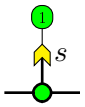
and



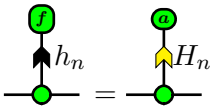
We now focus on families of diagonal gates that admit specific representations.

Hyper-graph operator

When an edge in a graph operator is represented by a controlled Z gate on two vertices, a hyper-edge in a hyper-graph operator is represented by a multi-controlled Z gate on a subset of the vertices. They can be easily represented in ZH-calculus by a Hadamard node. A phase function corresponding to an hyper-edge is defined by $\forall s, x \in 2^n \xi_s(x) \overset{\text{def}}{=} \delta_{s \leq x}$.

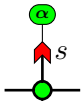
We can depict ξ_s as:  where s is the characteristic vector of the subset s. A composition of such gates is called a **hyper-graph operator**. We can represent them compactly in scalable notations. The matrix $H_n \in \{0,1\}^{2^n} \times \{0,1\}^n$ is defined inductively by: $H_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $H_{n+1} \overset{\text{def}}{=} \begin{pmatrix} 0 & \\ \vdots & H_n \\ 0 & \\ 1 & \\ \vdots & H_n \\ 1 & \end{pmatrix}$.

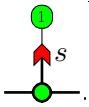
An hyper-graph operator corresponding to the hyper-graph G on n vertices is entirely defined by the phase function $g : 2^n \to \mathbb{R}$ such that $g(s) = \delta_{s \in G}$. H_n is a stack of all possible $s \in 2^n$. We can then draw the hyper-graph operator G:  . We can generalize this to any non $\{0,1\}$ phase functions $f : \mathbf{2}^n \to \mathbb{R}$ of the form: $f(s) = \sum_{x \in \mathbf{2}^n} a_x \xi_s(x)$. We have:



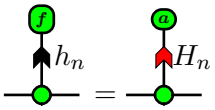
We will see later that in fact, any function arrow admits a representation of this form.

Phase gadgets

Phase gadgets are diagonal gates depicted by 

. They can be represented by the phase functions defined by $\forall s, x \in 2^n, \Omega_s(x) \stackrel{\text{def}}{=} x \cdot s$. Ω_s is depicted: 

. Like for hyper-graphs operator we can use the set matrix to depict a composition of phase gadget. We can represent any phase function $f : \mathbf{2}^n \to \mathbb{R}$ of the form: $f(s) = \sum_{x \in \mathbf{2}^n} a_x \Omega_s(x)$. We have:



We now apply those representations to graphical transforms.

10.2.2 Graphical transforms

The graphical Fourier theory was introduced in [25]. It was then stated in a mix of bang-boxes and ellipsis. We restate it here in an ellipsis-free way using scalable notations. We hope this new presentation allows us to grasp more clearly the underlying phenomena. The theory can be extended to other semi-boolean transforms. We do it there with the Möbius transform.

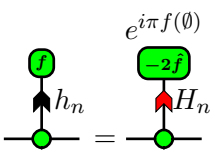
Walsh Fourier transform

Defining $\chi_s(x) \stackrel{\text{def}}{=} 1 - 2\Omega_s(x)$, the χ_s form an orthonormal basis of $\mathbb{R}^{\mathbf{2}^n}$ with respect to the scalar product $\langle f|g\rangle \stackrel{\text{def}}{=} \frac{1}{2^n} \sum_{x \in \mathbf{2}^n} f(x)g(x)$. The Walsh Fourier transform of a phase function f is defined by $\hat{f} \stackrel{\text{def}}{=} \frac{1}{2^n} \sum_{s \in 2^n} f(s)$:

$$f = \sum_{s \in \mathbf{2}^n} \hat{f}(s)\chi_s = \sum_{s \in \mathbf{2}^n} \hat{f}(s)\,(1 - 2\Omega_s) = \sum_{s \in \mathbf{2}^n} \hat{f}(s) - 2 \sum_{s \in \mathbf{2}^n} \hat{f}(s)\Omega_s = f(\emptyset) - 2 \sum_{s \in \mathbf{2}^n} \hat{f}(s)\Omega_s$$

We can use this formula to rewrite the diagonal gate associated with f as a composition of phase gadgets.

Lemma 35. *For all* $f : \mathcal{2}^n \to \mathbb{R}$



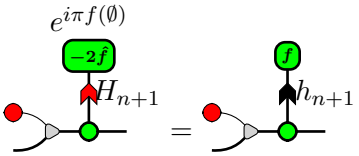
We provide graphical proof.

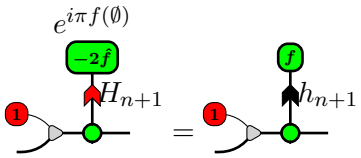
Proof. Denoting $f_0(x) \stackrel{\text{def}}{=} f(0x)$ and $f_1(x) \stackrel{\text{def}}{=} f(1x)$. Let the Walsh matrix be $W \stackrel{\text{def}}{=} \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. The Walsh Fourier transform of a phase function $f : 2^n \to \mathbb{R}$ is $\hat{f} \stackrel{\text{def}}{=} W^{\otimes n} f$.

$$\begin{pmatrix} \hat{f}_0 \\ \hat{f}_1 \end{pmatrix} = W^{\otimes n+1} \begin{pmatrix} f_0 \\ f_1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} I_n & I_n \\ I_n & -I_n \end{pmatrix} \begin{pmatrix} W^{\otimes n} f_0 \\ W^{\otimes n} f_1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} I_n & I_n \\ I_n & -I_n \end{pmatrix} \begin{pmatrix} \widehat{f_0} \\ \widehat{f_1} \end{pmatrix} = \begin{pmatrix} \frac{\widehat{f_0}+\widehat{f_1}}{2} \\ \frac{\widehat{f_0}-\widehat{f_1}}{2} \end{pmatrix}$$

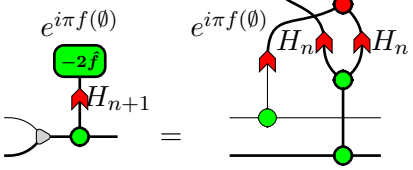
So:

$$\hat{f}_0 = \frac{\widehat{f_0}+\widehat{f_1}}{2} \qquad \hat{f}_1 = \frac{\widehat{f_0}-\widehat{f_1}}{2} \qquad \widehat{f_0} = \hat{f}_0 + \hat{f}_1 \qquad \widehat{f_1} = \hat{f}_0 - \hat{f}_1$$

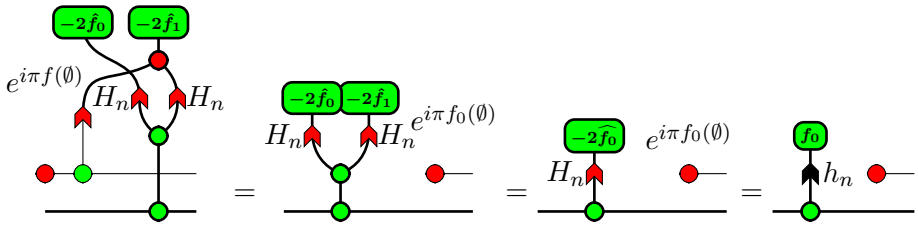
By induction, for $n = 1$ this is direct, for $n \geq 2$: We show 

and 

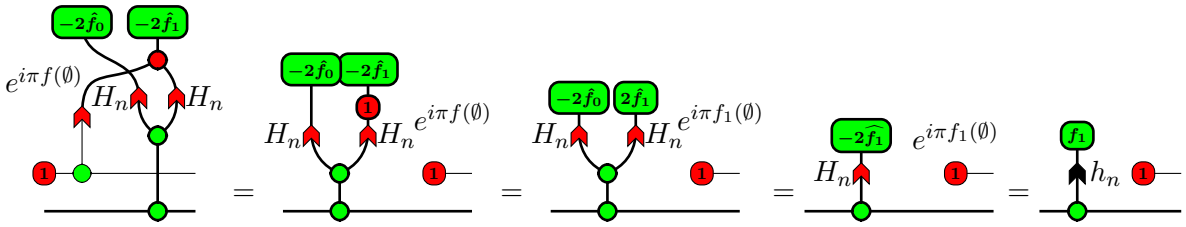
.

We have: 

so:



and



□

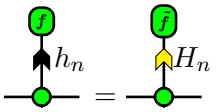
This equality has been proved in [25].

Möbius transform

The ξ_s also form a basis of $\mathbb{R}^{\mathbf{2}^n}$ associated with the Möbius transform, see [107] for details. The Möbius transform of a phase function f is defined by $\tilde{f}(x) = \sum_{s \leq x} (-1)^{|x|+|s|} f(s)$. We have:

$$f = \sum_{s \in 2^n} \tilde{f}(s) \xi_s$$

Lemma 36. *For all $f : \mathbf{2} \to \mathbb{R}$:*

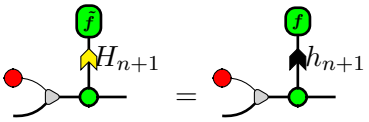


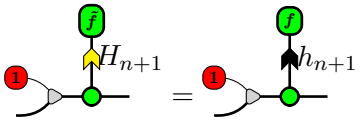
Proof. Let the Möbius matrix be $M \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$. The Möbius transform of a phase function $f : 2^n \to \mathbb{R}$ is $\tilde{f} \stackrel{\text{def}}{=} M^{\otimes n} f$.

$$\begin{pmatrix} \tilde{f}_0 \\ \tilde{f}_1 \end{pmatrix} = M^{\otimes n+1} \begin{pmatrix} f_0 \\ f_1 \end{pmatrix} = \begin{pmatrix} I_n & 0 \\ -I_n & I_n \end{pmatrix} \begin{pmatrix} M^{\otimes n} f_0 \\ M^{\otimes n} f_1 \end{pmatrix} = \begin{pmatrix} I_n & 0 \\ -I_n & I_n \end{pmatrix} \begin{pmatrix} \widetilde{f_0} \\ \widetilde{f_1} \end{pmatrix} = \begin{pmatrix} \widetilde{f_0} \\ \widetilde{f_1} - \widetilde{f_0} \end{pmatrix}$$

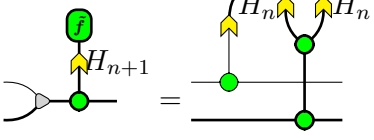
So:

$$\tilde{f}_0 = \widetilde{f_0} \qquad \tilde{f}_1 = \widetilde{f_1} - \widetilde{f_0} \qquad \widetilde{f_0} = \tilde{f}_0 \qquad \widetilde{f_1} = \tilde{f}_0 + \tilde{f}_1$$

By induction, for $n = 1$ it is direct, for $n \geq 2$: We show 

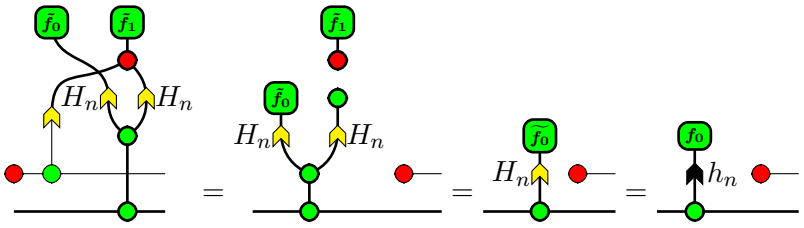
and 

.

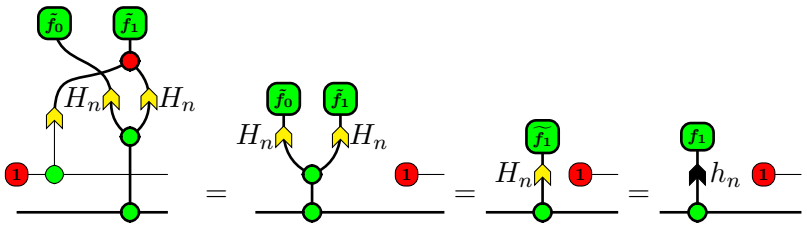
We have: 

=

so:



and



□

10.2.3 Spider nests

A spider nest identity is a composition of spiderlike diagrams, typically generalized hyperedges, and phase gadgets, with one big spider and a lot of very small ones. Furthermore, this composition must be the identity. We end this note by deriving some of them from graphical transforms. We restrict to **symmetric phase functions**, that is $f(x)$ only depends of the Hamming weight of x. We write $F : \mathbf{n} \to \mathbb{R}$ the function such that $f(x) = F(|x|)$.

Binomial transform

The Möbius transform of a symmetric semi-boolean function is the **binomial transform**:

$$\tilde{f}(x) = \sum_{s \leq x} f(s)(-1)^{|s|+|x|} = (-1)^{|x|} \sum_{k=0}^{|x|} \binom{|x|}{k} (-1)^k F(k)$$

We define $\tilde{F}(m) \stackrel{\text{def}}{=} \sum_{k=0}^{m} \binom{m}{k} (-1)^{m-k} F(k)$. We have $\tilde{f}(x) = \tilde{F}(|x|)$ and $F(m) = \sum_{k=0}^{m} \binom{m}{k} \tilde{F}(k)$.

We recover the spider nest identity of [27] by computing the Möbius transform of the phase function $G(m) = \alpha \frac{1-(-1)^m}{2}$ which represent one phase gadget on n qubits with phase α.

$$\tilde{G}(m) = \sum_{k=0}^{m} \binom{m}{k} (-1)^{m-k} G(k) = \frac{\alpha}{2} \left(\sum_{k=0}^{m} \binom{m}{k} (-1)^{m-k} - \sum_{k=0}^{m} \binom{m}{k} (-1)^{m-k} (-1)^k \right) = \frac{\alpha}{2} \left(\delta_{m=0} - (-2)^m \right).$$

$\tilde{G}(0) = 0$ so we have no floating scalars. For $k \geq 1$, $\tilde{G}(k) = \alpha(-2)^{m-1}$. Setting $\alpha = \frac{1}{4}$ we see that only the first terms $\tilde{G}(1) = \frac{1}{4}$, $\tilde{G}(2) = \frac{-1}{2}$ and $\tilde{G}(3) = 1$ are relevant for the phase gate.

This has been proved by induction in [27].

Kravchuk transform

The case of the Fourier transform is more complex:

$$\hat{f}(x) = \frac{1}{2^n} \sum_{s \in \mathbf{2}^n} f(s)(-1)^{s \cdot x} = \frac{1}{2^n} \sum_{k \in \mathbf{n}} F(k) \sum_{|s|=k} (-1)^{s \cdot x}.$$

$\sum_{|s|=k} (-1)^{s \cdot x}$ is also a symmetric boolean function equal to the **Kravchuk polynomial**:

$$\mathcal{K}_k^n(|x|) \stackrel{\text{def}}{=} \sum_{j=0}^{k} \binom{|x|}{j} \binom{n-|x|}{k-j} (-1)^j.$$

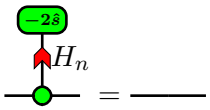
To see this, consider j as the number of ones in comon between x and s. The Kravchuk polynomials satisfy: $\binom{n}{m}\mathcal{K}_k^n(m) = \binom{n}{k}\mathcal{K}_m^n(k)$ and $\sum_{i=0}^{n}\binom{n}{i}\mathcal{K}_k^n(i)\mathcal{K}_l^n(i) = 2^n\binom{n}{k}\delta_{k=l}$.

The Walsh Fourier transform of a symmetric semi-boolean function is then the **Kravchuk transform**: $\hat{F}(m) \stackrel{\text{def}}{=} \frac{1}{2^n}\sum_{k=0}^{n} F(k)\mathcal{K}_k^n(m)$. We have: $\hat{f}(x) = \hat{F}(|x|)$ and $F(m) = \sum_{k=0}^{n}\hat{F}(k)\mathcal{K}_k^n(m)$. See [108] for details on transforms of symmetric semi-boolean functions.

We can compute the transform of the phase function $H(m) \stackrel{\text{def}}{=} \beta\delta_{m=n}$ which corresponds to the generalised hyperedge on n qubits with phase β.

$$\hat{H}(m) = \tfrac{1}{2^n}\sum_{k=0}^{n} H(k)\mathcal{K}_k^n(m) = \tfrac{1}{2^n}\sum_{k=0}^{n}\beta\delta_{k=n}\mathcal{K}_k^n(m) = \tfrac{\beta}{2^n}\mathcal{K}_n^n(m) = \tfrac{\beta}{2^n}(-1)^m.$$

Combining this result with the spider nest identity of the previous section it is possible to derive the spider nest identity from [26] as it is done in [27]. We end this note by giving an alternative proof by inversion. We first sketch a method to check spider nest identity. We want to show that for a symmetric phase function $\hat{s} : \mathbf{2}^n \to \mathbb{R}$:



We know $\hat{S}$ by reading the coefficients in the phase gadgets. We compute S using the inversion formula. Then we check if all values of S are equal modulo 2. If it is the case this means that the corresponding phase gadget is $e^{i\pi S(0)}I_n$. But this scalar is exactly the one appearing in the graphical Fourier transform. So simplifying on both sides gives us exactly what we want.

We apply this method to the spider nest identity of [26]. Here, only the Kravchuk polynomials for $k = 0, 1, 2, 3$ and n are needed:

$$\mathcal{K}_0^n(m) = 1 \quad \mathcal{K}_1^n(m) = -2m + n \quad \mathcal{K}_2^n(m) = 2m^2 - 2nm + \tfrac{n^2-n}{2} \quad \mathcal{K}_n^n(m) = (-1)^m$$

$$\mathcal{K}_3^n(m) = -\tfrac{4}{3}m^3 + 2nm^2 + (-n^2 + n - \tfrac{2}{3})m + \tfrac{n^3-3n^2+2n}{6}$$

We want to inverse the phase function:

$$\hat{S}(0) = 0\ \hat{S}(1) = \tfrac{(n-2)(n-3)}{16}\ \hat{S}(2) = -\tfrac{n-3}{8}\ \hat{S}(3) = \tfrac{1}{8}\ \hat{S}(n) = -\tfrac{1}{8}\ \hat{S}(k) = 0 \text{ for } k \neq 0, 1, 2, 3, n.$$

Lemma 37. *Forall $m \in [\![0, n]\!]$, $S(m) = S(0) \mod 2$*

Proof.

$$\begin{aligned} S(m) &= \sum_{k=0}^{n}\hat{S}(k)\mathcal{K}_k^n(m) = \hat{S}(1)\mathcal{K}_1^n(m) + \hat{S}(2)\mathcal{K}_2^n(m) + \hat{S}(3)\mathcal{K}_3^n(m) + \hat{S}(n)\mathcal{K}_n^n(m) \\ &= \frac{(n-2)(n-3)}{16}\mathcal{K}_1^n(m) - \frac{n-3}{8}\mathcal{K}_2^n(m) + \frac{1}{8}\mathcal{K}_3^n(m) - \frac{1}{8}\mathcal{K}_n^n(m) \\ &= \frac{-m^3}{6} + \frac{3m^2}{4} - \frac{5m}{6} + \frac{n^3}{48} - \frac{n^2}{8} + \frac{11n}{48} - \frac{(-1)^m}{8} \end{aligned}$$

Our goal is to check that $S(m) \mod 2$ doesn't depend on m. Thus, we only need the part of $S(m)$ that depends on m.

$$S'(m) \stackrel{\text{def}}{=} \frac{-m^3}{6} + \frac{3m^2}{4} - \frac{5m}{6} - \frac{(-1)^m}{8}.$$

Since $S'(0) = -\frac{1}{8} \mod 2$, we want to check that for each $m \in \mathbb{N}$, $S'(m) \equiv -\frac{1}{8} \mod 2$. To do so we write $m = 12k + l$ with $k \in \mathbb{N}$ and $l \in [\![0, 11]\!]$. We obtain:

$$S'(12k+l) = -288k^3 - 72lk^2 + 108k^2 - 6kl^2 + 18kl - 10k - \frac{l^3}{6} + \frac{3l^2}{4} - \frac{5l}{6} - \frac{(-1)^l}{8}$$

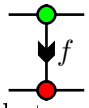
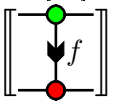
We see that $S'(12k+l) = -\frac{l^3}{6} + \frac{3l^2}{4} - \frac{5l}{6} - \frac{(-1)^l}{8} \mod 2$, this only depends on l. Thus we can just check that for each $l \in [\![0, 11]\!]$, $-\frac{l^3}{6} + \frac{3l^2}{4} - \frac{5l}{6} - \frac{(-1)^l}{8} = -\frac{1}{8} \mod 2$ (which is true). □

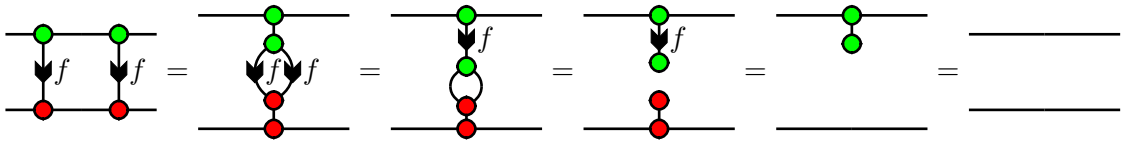
10.3 Algorithms

Many quantum algorithms are defined using oracles. An oracle can be viewed as a black box, it is not, however, an arbitrary map, a quantum oracle may have some structure: they are usually quantum encodings of classical functions, moreover, some promises can provide additional information about the behaviour of the oracle. In the spirit of the ZX-calculus, we decompose, in this section, classes of quantum oracles into smaller components with better understood algebraic properties.

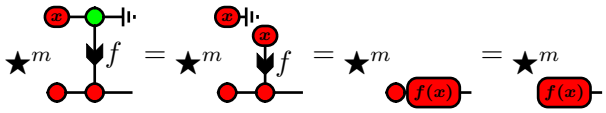
A section of the "Dodo book" [89] is dedicated to the description of quantum algorithms in ZX-calculus (in particular Deutsch-Jozsa and Grover), and a few articles [23, 69] address the diagrammatic description of quantum oracles.

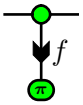
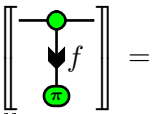
10.3.1 Oracles

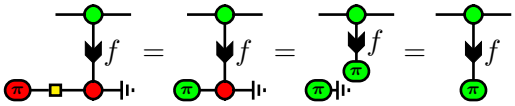
The function arrow of f is unitary if and only if f is a bijection. There is however a standard way to associate with any function $f : \mathbf{2}^n \to \mathbf{2}^m$ a unitary transformation defined as $U_f = |x\rangle |y\rangle \mapsto |x\rangle |f(x) \oplus y\rangle$, often call **quantum oracle**. As pointed out in [89], the quantum oracle can be constructed as follows:  $: [n] \otimes [m] \to [n] \otimes [m]$. Indeed,  $= |x\rangle |y\rangle \mapsto |x\rangle |f(x) \oplus y\rangle$. We can double check that quantum oracles are involutions:



From a quantum oracle, we can easily compute the original function using ancillas.



Note that the Toffoli gate is in fact the quantum oracle representing the AND gate. Often, we consider boolean functions, then, another kind of oracle is available. For any boolean function $f : \mathbf{2}^n \to \mathbf{2}$, the diagonal oracle of f is  $: [n] \to [n]$:  $= |x\rangle \mapsto (-1)^{f(x)} |x\rangle$. We can construct the diagonal oracle from the oracle using ancillas:



We have now enough graphical structures to tackle the most basic quantum algorithms.

10.3.2 Quantum algorithms relying on a single application of the oracle

In this section we provide a diagrammatic treatment of some quantum algorithms that frequently appear in quantum computing textbooks like [45]. They are oracle-based: given a function that satisfies some properties (the promise), we want to recover some information about the function using a minimal number of queries to the corresponding quantum oracle.

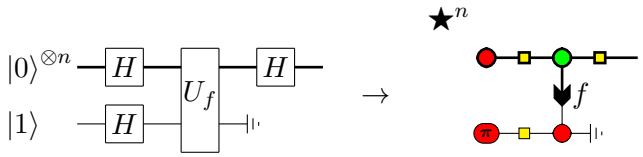
Bernstein-Vazirani

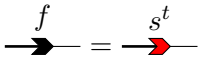
The Bernstein-Vazirani algorithm has been introduced in [109]. The goal is to recover a string of bits encoded into a function.

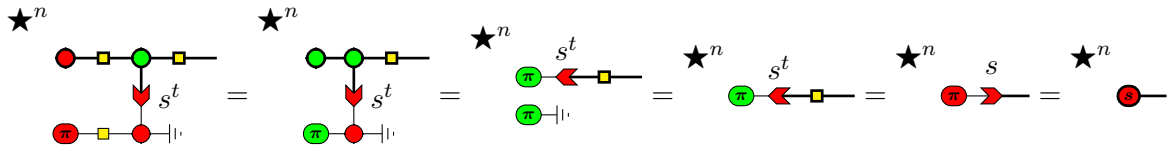
Input: A function $f : \{0,1\}^n \to \{0,1\}$ of the form $f(x) = s^t \cdot x$ with $s \in \{0,1\}^n$.

Problem: Find s.

Circuit:



Reformulating graphically the promise on f gives us:  and then:



We see the circuit directly outputs the state $|s\rangle$.

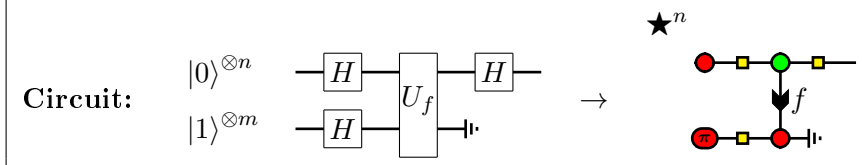
Deutsch-Jozsa

The Deutsch-Jozsa algorithm [1] is historically the first of all quantum algorithms. Given a function that is known to be either constant or balanced, the goal is to decide in which case we are using only one query to the oracle. The version we present here is a little bit more general than usual since we do not require f to output a single bit. The general principle is the same as Bernstein-Vazirani,the difference is that we are here interested in the probability of outputting $|0\rangle^{\otimes n}$.

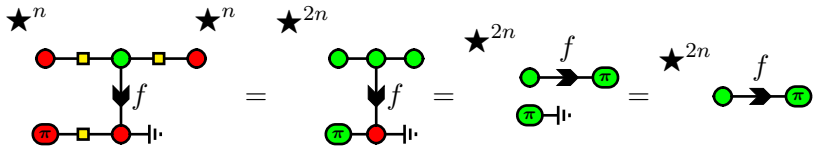
Input: A function $f : \{0,1\}^n \to \{0,1\}^m$ which is either constant or balanced.

Problem: Decide whether f is constant or balanced.

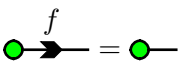
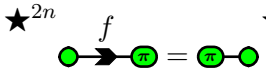
Circuit:

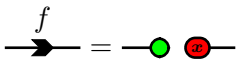


We compute the amplitude of the outcome $|0\rangle^{\otimes n}$:



We then have two cases:

- If f is balanced then  and 

$= 0$.
- If f is constant then there exists $x \in \{0,1\}^m$ such that  :

$= 1$.

So if the outcome is $|0\rangle^{\otimes n}$ then f is constant otherwise f is balanced.

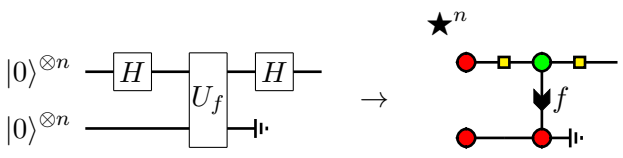
Simon

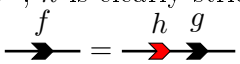
Simon's algorithm is more subtle than the algorithm we have seen so far. This algorithm is probabilistic, moreover, the quantum computation is combined with classical processing. We are given a strictly periodic function f and the goal is to find the period s. The quantum part of the algorithm is nothing but a random generator that outputs a string y such that $y \cdot s = 0$, in a uniform way. Repeating this quantum part several times, gives, with high probability, enough linearly independent equations to solve the linear system with a classical algorithm and find s.

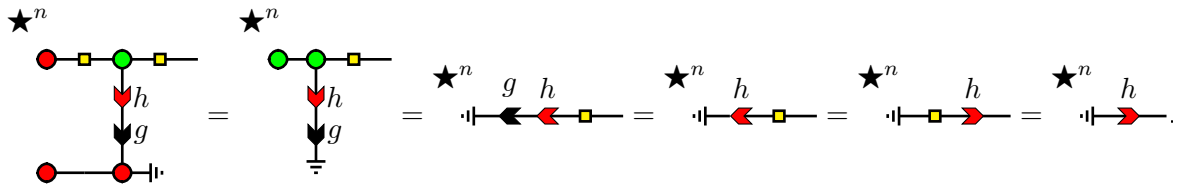
Input: A function $f : \{0,1\}^n \to \{0,1\}^n$ with an $s \in \{0,1\}^n$, $s \neq 0^n$, such that: $f(x) = f(y) \Leftrightarrow (x = y) \vee (x \oplus s = y)$.

Problem: Find s.

Circuit:



The translation of the promise into a graphical property is less straightforward than with the algorithms we have seen so far. Let h be an orthogonal projector on $s^\perp$, h is clearly strictly s periodic. So there is a bijective function $g : \{0,1\}^n \to \{0,1\}^n$ such that  . The circuit reduces to:



Since by definition $h^t = h$. We can directly see the resulting state: it is a uniform mixture of the elements in $s^\perp$. In other words, we can use this circuit to sample uniformly at random vectors y_i such that $y_i \cdot s = 0$.

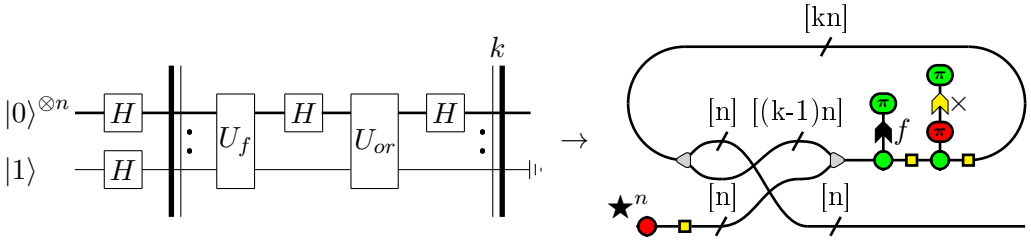
10.3.3 Iteration and Grover algorithm

The last and most famous algorithm we present is Grover's algorithm [110]. Given a boolean function $f : \mathbf{2}^n \to \mathbf{2}$ such that 1 has a unique preimage x_0. The objective is to find x_0. Roughly speaking, the algorithm consists in applying k times a combination of the quantum oracle and a diffusion operator on the superposition of all classical inputs. We then show that choosing k wisely, the output is $|x\rangle$ with a high probability.

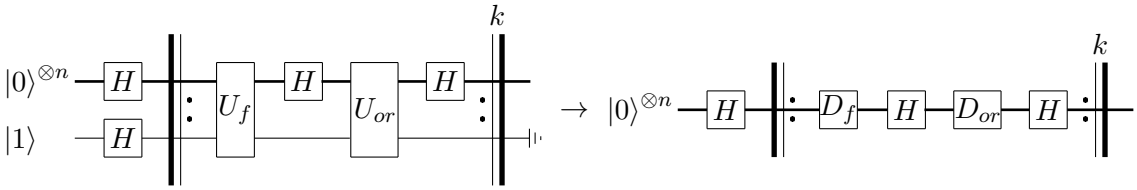
Input: A boolean function $f : \{0,1\}^n \to \{0,1\}$ such that $f^{-1}(\{1\}) = \{x\}$ with $x \in \{0,1\}^n$.

Problem: Find x.

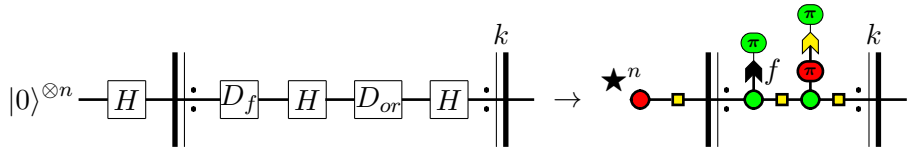
Circuit:



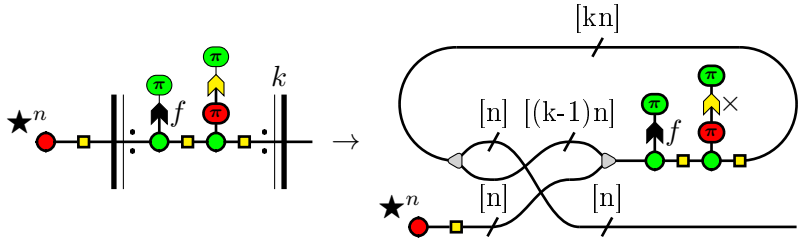
Here we need to explain the translation into diagrams. First the ancillas is only here to form the diagonal oracles we then have:

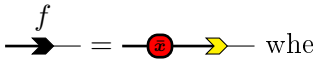


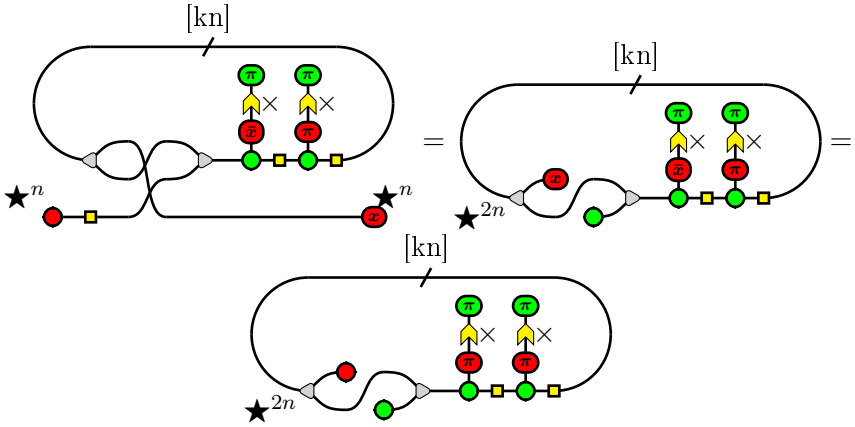
translating into diagrams:

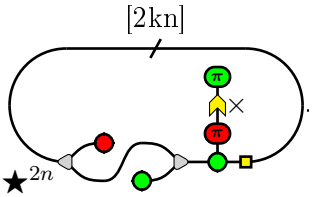


Now, using the iteration mechanism of Lemma 3 to represent the k queries gives:

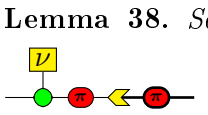


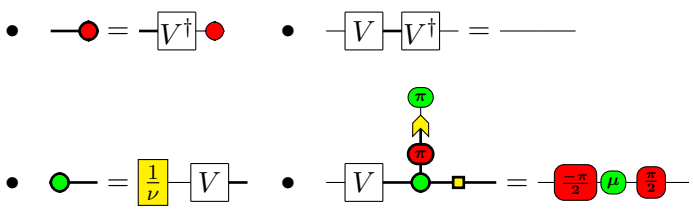
The $\times$ stands for the matrix resulting of the thickening of the AND gate. The promise translates to:  where $\bar{x}$ is the bit-wise negation of x. Our goal is to compute the probability of the outcome $|x\rangle$. Making the x red phase slides gives:



Using the iteration mechanism we get:  .

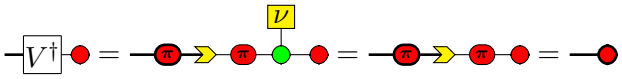
Here we will translate a geometric approach into diagrams.

Lemma 38. *Setting* $\nu \overset{\text{def}}{=} \frac{1}{\sqrt{2^n-1}}$, $\cos(\frac{\mu}{2}) \overset{\text{def}}{=} \frac{-1}{\sqrt{2^n}}$ *and* $\sin(\frac{\mu}{2}) \overset{\text{def}}{=} \frac{\sqrt{2^n-1}}{\sqrt{2^n}}$, *the map*  *satisfies:*

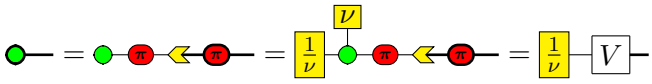


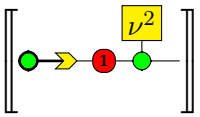
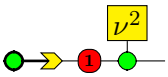
Proof. We proceed point by point:

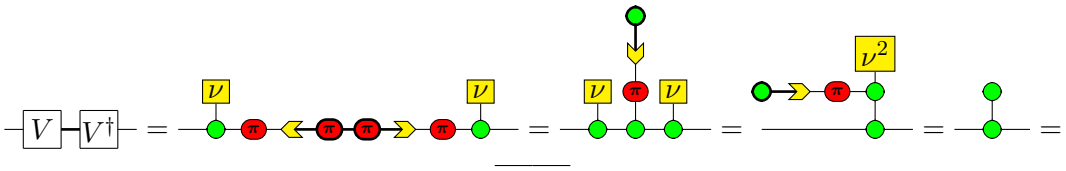
- By rewriting:

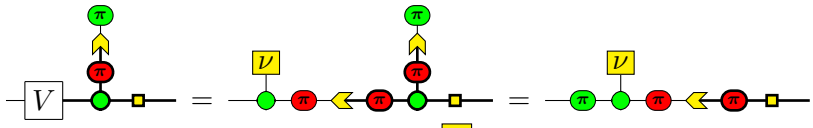
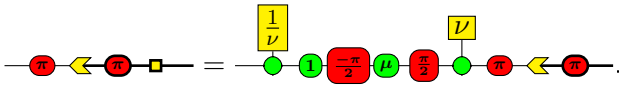


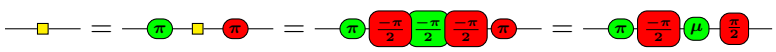
- By rewriting:



- First we can check:  $= [\![\; — \;]\!]$ and then  $=$ —. So:



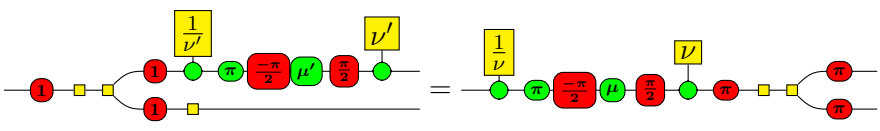
- First we have :  . So the question reduces to show that:  .
  We proceed by induction on n. If $n = 1$ then $\nu = 1$ and $\mu = \frac{-\pi}{2}$:



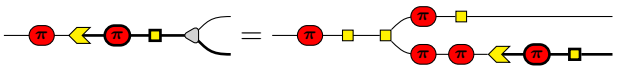
  For $n > 1$ we need the following observation:

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \otimes \begin{pmatrix}\frac{1}{\sqrt{2^{n-1}}} & \frac{2^{n-1}-1}{\sqrt{2^{n-1}}}\\ \frac{1}{\sqrt{2^{n-1}}} & \frac{-1}{\sqrt{2^{n-1}}}\end{pmatrix} \begin{pmatrix}1 & 0\\ 0 & 1\\ 0 & 1\\ 0 & 1\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & 1\\ 0 & 1\\ 0 & 1\end{pmatrix} \begin{pmatrix}\frac{1}{\sqrt{2^n}} & \frac{2^n-1}{\sqrt{2^n}}\\ \frac{1}{\sqrt{2^n}} & \frac{-1}{\sqrt{2^n}}\end{pmatrix}.$$

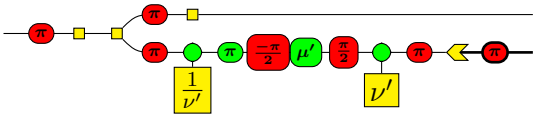
  Graphically:



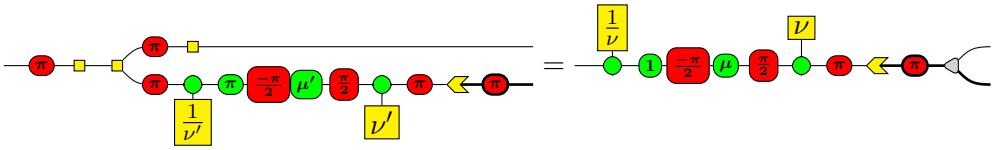
  with $\nu' \overset{\text{def}}{=} \frac{1}{\sqrt{2^{n-1}-1}}$, $\cos(\frac{\mu'\pi}{2}) \overset{\text{def}}{=} \frac{-1}{\sqrt{2^{n-1}}}$ and $\sin(\frac{\mu'\pi}{2}) \overset{\text{def}}{=} \frac{\sqrt{2^{n-1}-1}}{\sqrt{2^{n-1}}}$.
  We have:



The induction hypothesis gives:

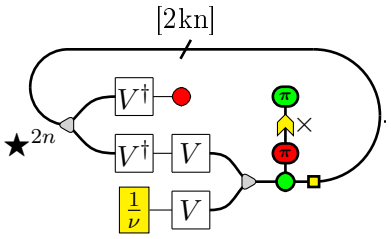


And finally, using the previous observation:

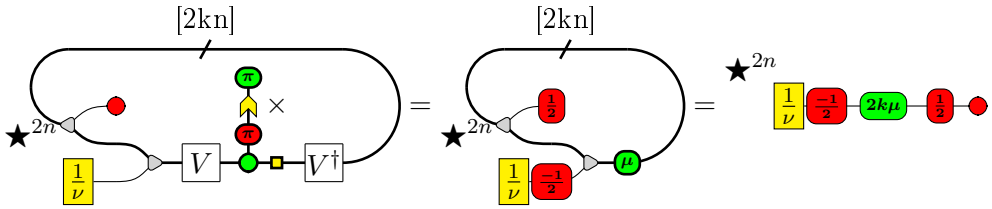


□

This lemma allows us to rewrite the diagram as follows:



Making the isometries slide gives:



The last step being the iteration mechanism. We can now compute the interpretation:

$$\left[\!\!\left[\star^{2n} \; \frac{1}{\nu} \; \frac{-1}{2} \; 2k\mu \; \frac{1}{2} \right]\!\!\right] = \left| (1 \quad 0) \begin{pmatrix} \cos(k\mu) & \sin(k\mu) \\ -\sin(k\mu) & \cos(k\mu) \end{pmatrix} \begin{pmatrix} -\cos(\frac{\mu}{2}) \\ \sin(\frac{\mu}{2}) \end{pmatrix} \right|^2 = \cos^2(\tfrac{2k+1}{2}\mu)$$

So the probability is maximal when $\frac{2k+1}{2}\mu \simeq 0 \bmod \pi$. Moreover $\mu = \pi \pm \frac{2}{\sqrt{2^n}} + o(\frac{1}{\sqrt{2^n}})$, thus $k \simeq \frac{\pi}{4}\sqrt{2^n}$.

If the graphical verification of most of the other algorithms we presented are neat and straightforward. This approach of Grover's algorithm is still not rigorous enough to be implemented in a future proof assistant. More works need to be done on the higher-level structure like the iteration mechanism we have sketched out.

Conclusion

There are different schools when it comes to conclusions. Either it is a cardinal sin to omit it or if you have nothing more to say then say nothing. For some reason, I am more inclined toward the second one. However, in the present case, I still have some things to say.

The PhD thesis tells (and I think has to tell) a very different story than what really happened during three years of research.

When I started my PhD I had a precise plan in mind: designing a graphical language mixing the ZX-calculus and the proof nets of linear logic to work on higher-order quantum transformations. I never was really really satisfied by the result and nothing came out of it but at least those works already had some scalable notations hidden in them. Very quickly I started to work on the scalable ZX-calculus with Simon Perdrix and Dominic Horsman. If the main idea were there very quickly it took a very long time to write them down [2]. Meanwhile, I was invited by Simon Perdrix, Emmanuel Jeandel, and Renaud Vilmart to work on the completeness of the ZX calculus with grounds. Following discussions with Mathieu Huot this paper ended far more categorical than expected, but in a good way though.

At some point, I became interested in the links between group algebras and ZX-calculus. I obtained rules very similar to the ZW-calculus which lead me to consider flexsymmetry and the softening trick of Chapter 5 for the first time. However, it took at least a year to come up with the paradigms of Chapter 4 to clarify the idea. Looking at so many spiders finally lead me and Emmanuel Jeandel to try to find all of them which gave the notion of Z^*-algebra presented in Chapter 6.

In parallel, we had discussions with Robert Booth and Damian Markham on the possibility to draw the group algebra of infinite groups. This led me and Robert Booth to work on the endless sea of dead ends that is the design of graphical languages for continuous-variable quantum computing, but I still have some hope. The desperate search for valid semantics for such infinite-dimensional graphical languages led to discussions with Marc De Visme which ended up with the extension of SZX calculus to streams of qubits, an old idea that came back to the beginning of scalable ZX-calculus. Sadly this line of research is not presented in this thesis.

This brings us to another purpose of conclusions, advertising further research and prospects.

- I really think the paradigms of Chapter 3 could be useful to clarify graphical language design. In my mind, there really is a correspondence between paradigm and implementation of the paradigmatic graphical languages into proof assistants. I hope I will have the opportunity someday to provide more support to this claim by developing the paradigm formalism further. Maybe by extending the notion to a framework where paradigmatic generators and equations can depend on generators and equations in a natural way?

[2] Mainly because I was drawing everything in Tikz directly, before what I call the Tikzit miracle. I am very grateful to the developers of this software that was used for almost all the drawings in this thesis.

- Flexsymmetry, introduced in Chapter 5, is, I think, a nice notion that allows in a concise and formal way to say what we want to say when we speak of graphical languages. Where graphical here really means related to graphs. I have only developed it in the monochromatic case the coloured version might also be of interest, in connection with graphs with coloured edges. I also did not mention a weaker notion, flexcyclicity, that allows to keep a cyclic ordering of the edges connected to one vertex. This notion is connected to port-graphs and would allow to tackle the case of non-commutative symmetric Frobenius algebras.

- The classification of Z^*-algebra in Chapter 6 could be done in higher dimensions than two. I personally think that dimensions 3 and 4 might be of interest, mainly because numerous notions in physics are defined in those dimensions. In even higher dimensions, if we can look at generalizations of ZX, ZH, and ZW to qudits, I can't think of really good motivation to do that for now. Furthermore, there are good reasons to expect an infinite number of Z^*-algebras up to isomorphism. One could also look at other monoidal categories like we did for **LinRel** in Chapter 7.

- The discard construction presented in Chapter 8 only is equivalent to the CPM construction if we have enough isometries. This leaves open the axiomatization of the Clifford+T mixed state fragment. I personally thought for some time that the discard construction was more fundamental than the CPM construction which I saw as a direct mimicking of the density matrix construction. I have since changed my mind seeing the beautiful application of the CPM construction to other situations [111]. I feel there is still some unifying to do in the world of mixed-state categorical quantum mechanics. A lot of constructions have been proposed and the precise links between all of them are obscured by the fact that they all coincide in the case of Hilbert spaces.

- I now think that the scalable notations of Chapter 9 are more fundamental than I first thought. They were first designed with a very pragmatic goal in mind but the more I worked with them the more I started to remark how natural they are. The distribution equation seems to hide a natural transformation and the scalable construction seems related to the work on concategories and strictification [39]. In a sense, the dividers and gatherer are natural isomorphisms between objects that have been artificially differentiated to work into a prop instead of a strict symmetric monoidal category. This might be the subject of future work.

- In Chapter 10, we have used scalable notations to verify some standard quantum algorithms. For the moment, this work is merely exploratory. Trying to tackle graphically many algorithms and protocols is the only way to evaluate our graphical methods. The ultimate goal is to be able to compile high-level quantum programming languages directly into diagrams. Then a graphical proof assistant could be used to provide proofs of correctness and optimizations. Case studies like this are steps toward a double understanding. First, how graphical languages must be designed to fit this purpose. Second, what should be the specifications of future graphical proof assistants. Those two perspectives are clearly entangled.

I expect those research projects to follow the usual ratio of survival of research ideas to more serious investigations. Which is approximatively one over ten.

My goal in this thesis was to make the diagrammatic methods as operational as possible. The flexsymmetry paradigm gives a very compact presentation by hierarchizing the different

properties of ZX-calculus. The concept of Z^*-algebra makes clear the links between the three quantum graphical calculi. I think this allows for a pragmatic conjoint use of the three languages together as I kind of did in Chapter 10.

The extension to mixed states allows us to express measurements, randomness, and classical control in the language. It even simplifies some computations by getting rid of the scalars implementing global phases. I personally think the discard construction has huge advantages over the doubling in practice but only usage can tell.

Finally, the scalable notations allow compact presentations of quantum protocols. Of course, there are a lot of other works going in the same direction and I do not claim to have reached an ultimate presentation of quantum graphical languages, but this was my guiding goal.

As I started my PhD I saw the completeness era come to an end. I believe that we enter into a period of experimentation. We now see different people from various fields showing interest in the ZX-calculus. It seems that it is now time for the graphical languages to be confronted with all quantum phenomenon one can think of and evolve in consequence. At least, this is what I see happening on the ZX-discord.

If the work presented here can help anyone have a better grasp of the nature of diagrammatic methods, or motivate someone to use those notations for anything they like, then those three years might not have been in vain. Anyway, at least, personally, in hindsight, I overall very enjoyed it.

Bibliography

[1] David Deutsch and Richard Jozsa. "Rapid solution of problems by quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439.1907 (1992), pp. 553–558.

[2] Richard Cleve et al. "Quantum algorithms revisited". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 339–354.

[3] Peter W. Shor. "Polynominal time algorithms for discrete logarithms and factoring on a quantum computer". In: *Algorithmic Number Theory, First International Symposium, ANTS-I, Ithaca, NY, USA, May 6-9, 1994, Proceedings*. Ed. by Leonard M. Adleman and Ming-Deh A. Huang. Vol. 877. Lecture Notes in Computer Science. Springer, 1994, p. 289. DOI: 10.1007/3-540-58691-1_68. URL: https://doi.org/10.1007/3-540-58691-1_68.

[4] Titouan Carette. "When Only Topology Matters". In: *arXiv preprint arXiv:2102.03178* (2021).

[5] Titouan Carette and Emmanuel Jeandel. "A recipe for quantum graphical languages". In: *47th International Colloquium on Automata, Languages and Programming* (2020).

[6] Titouan Carette et al. "Completeness of Graphical Languages for Mixed States Quantum Mechanics". In: *International Colloquium on Automata, Languages, and Programming (ICALP'19)*. 2019.

[7] Titouan Carette, Dominic Horsman, and Simon Perdrix. "SZX-Calculus: Scalable Graphical Quantum Reasoning". In: *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*. Ed. by Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen. Vol. 138. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 55:1–55:15. DOI: 10.4230/LIPIcs.MFCS.2019.55. URL: https://doi.org/10.4230/LIPIcs.MFCS.2019.55.

[8] Titouan Carette and Simon Perdrix. "Colored props for large scale graphical reasoning". In: *arXiv preprint arXiv:2007.03564* (2020).

[9] Titouan Carette, Yohann D'Anello, and Simon Perdrix. "Quantum Algorithms and Oracles with the Scalable ZX-calculus". In: *arXiv preprint arXiv:2104.01043* (2021).

[10] Titouan Carette. "A note on diagonal gates in SZX-calculus". In: *arXiv preprint arXiv:2012.09540* (2020).

[11] Bob Coecke and Ross Duncan. "Interacting Quantum Observables: Categorical Algebra and Diagrammatics". In: *New Journal of Physics* 13.4 (2011), p. 043016. DOI: 10.1088/1367-2630/13/4/043016. URL: https://doi.org/10.1088%2F1367-2630%2F13%2F4%2F043016.

[12] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. "A complete axiomatisation of the ZX-calculus for Clifford+ T quantum mechanics". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. ACM. 2018, pp. 559–568.

[13] Kang Feng Ng and Quanlong Wang. "Completeness of the ZX-calculus for pure qubit Clifford+ T quantum mechanics". In: *arXiv preprint arXiv:1801.07993* (2018).

[14] Amar Hadzihasanovic. "A Diagrammatic Axiomatisation for Qubit Entanglement". In: *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*. 2015, pp. 573–584. DOI: 10.1109/LICS.2015.59.

[15] Miriam Backens and Aleks Kissinger. "ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity". In: *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018*. Ed. by Peter Selinger and Giulio Chiribella. Vol. 287. Electronic Proceedings in Theoretical Computer Science. 2019, pp. 23–42. DOI: 10.4204/EPTCS.287.2.

[16] Peter Selinger. "Dagger Compact Closed Categories and Completely Positive Maps". In: *Electronic Notes in Theoretical Computer Science* 170 (2007), pp. 139–163. DOI: 10.1016/j.entcs.2006.12.018. URL: https://doi.org/10.1016%2Fj.entcs.2006.12.018.

[17] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017. ISBN: 9781316219317. DOI: 10.1017/9781316219317. URL: https://doi.org/10.1017/9781316219317.

[18] Nicholas Chancellor et al. "Graphical structures for design and verification of quantum error correction". In: *arXiv preprint arXiv:1611.08012* (2016).

[19] Fabio Zanasi. "Interacting Hopf Algebras: the theory of linear systems". In: *PhD thesis, arXiv preprint arXiv:1805.03032* (2018).

[20] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. "A categorical semantics of signal flow graphs". In: *International Conference on Concurrency Theory*. Springer. 2014, pp. 435–450.

[21] Ross Duncan and Kevin Dunne. "Interacting Frobenius Algebras are Hopf". In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. ACM. 2016, pp. 535–544.

[22] Pawel Sobocinski. *Graphical Linear Algebra*. URL: https://graphicallinearalgebra.net/.

[23] William Zeng and Jamie Vicary. "Abstract structure of unitary oracles for quantum algorithms". In: *arXiv preprint arXiv:1406.1278* (2014). DOI: 10.4204/EPTCS.172.19.

[24] Ross Duncan and Simon Perdrix. "Graph states and the necessity of Euler decomposition". In: *Conference on Computability in Europe (CiE)*. Springer. 2009, pp. 167–177.

[25] Stach Kuijpers, John van de Wetering, and Aleks Kissinger. "Graphical fourier theory and the cost of quantum addition". In: *arXiv preprint arXiv:1904.07551* (2019).

[26] Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. "Fast and effective techniques for T-count reduction via spider nest identities". In: *arXiv preprint arXiv:2004.05164* (2020).

[27] Anthony Munson, Bob Coecke, and Quanlong Wang. "AND-gates in ZX-calculus: spider nest identities and QBC-completeness". In: ().

[28] Saunders Mac Lane. "The development and prospects for category theory". In: *Applied Categorical Structures* 4.2 (1996), pp. 129–136.

[29] Pierre-Louis Curien. "The joy of string diagrams". In: *International Workshop on Computer Science Logic*. Springer. 2008, pp. 15–22.

[30] Daniel Marsden. "Category theory using string diagrams". In: *arXiv preprint arXiv:1401.7220* (2014).

[31] Jiří Adámek, Horst Herrlich, and George E Strecker. "Abstract and concrete categories. The joy of cats". In: (2004).

[32] Saunders Mac Lane. *Categories for the working mathematician*. Vol. 5. Springer Science & Business Media, 2013.

[33] Steve Awodey. *Category theory*. Oxford university press, 2010.

[34] Michael Barr and Charles Wells. "Toposes, theories, and triples". In: *Reprints in Theory and Applications of Categories* 12 (2005), pp. 1–287.

[35] Hugh Tebby. *Impro etc.* URL: `https://improetc.wordpress.com/2018/11/20/interview-with-keith-johnstone/#more-3436`.

[36] John C Baez, Brandon Coya, and Franciscus Rebro. "PROPS IN NETWORK THEORY". In: *Theory and Applications of Categories* 33.25 (2018), pp. 727–783.

[37] Filippo Bonchi, Pawel Sobocinski, and Fabio Zanasi. "Full abstraction for signal flow graphs". In: *ACM SIGPLAN Notices* 50.1 (2015), pp. 515–526.

[38] Saunders MacLane. "Categorical algebra". In: *Bulletin of the American Mathematical Society* 71.1 (1965), pp. 40–106.

[39] Paul Blain Levy, Sergey Goncharov, and Lutz Schröder. *Traced Concategories*. 2018. URL: `http://events.cs.bham.ac.uk/syco/2/slides/levy.pdf`.

[40] Philip Hackney and Marcy Robertson. "On the category of props". In: *Applied Categorical Structures* 23.4 (2015), pp. 543–573.

[41] Peter Selinger. "A survey of graphical languages for monoidal categories". In: *New structures for physics*. Springer, 2010, pp. 289–355.

[42] André Joyal and Ross Street. "The geometry of tensor calculus, I". In: *Advances in mathematics* 88.1 (1991), pp. 55–112.

[43] Aleks Kissinger. "Graph rewrite systems for classical structures in†-symmetric monoidal categories". PhD thesis. Citeseer, 2008.

[44] Werner Heisenberg. *Der Teil und das Ganze: Gespräche im Umkreis der Atomphysik*. Piper verlag, 1969.

[45] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[46] Noson S Yanofsky and Mirco A Mannucci. *Quantum computing for computer scientists*. Cambridge University Press, 2008.

[47] John van de Wetering. "ZX-calculus for the working quantum computer scientist". In: *arXiv preprint arXiv:2012.13966* (2020).

[48] Aleks Kissinger and John van de Wetering. "Reducing T-count with the ZX-calculus". In: *arXiv preprint arXiv:1903.10477* (2019).

[49] Aleks Kissinger and Vladimir Zamdzhiev. "Quantomatic: A proof assistant for diagrammatic reasoning". In: *International Conference on Automated Deduction*. Springer. 2015, pp. 326–336.

[50] Niel de Beaudrap and Dominic Horsman. "The ZX calculus is a language for surface code lattice surgery". In: *https://quantum-journal.org/papers/q-2020-01-09-218/* (2017).

[51] Aleks Kissinger and John van de Wetering. "Universal MBQC with generalised parity-phase interactions and Pauli measurements". In: *arXiv:1704.06504* (2017).

[52] Niel de Beaudrap. "Well-tempered ZX and ZH calculi". In: *arXiv preprint arXiv:2006.02557* (2020).

[53] Stephen Lack. "Composing props". In: *Theory and Applications of Categories* 13.9 (2004), pp. 147–163.

[54] Renaud Vilmart. "A Near-Optimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics". In: *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2019. eprint: arXiv:1812.09114. URL: https://arxiv.org/abs/1812.09114.

[55] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. "A Generic Normal Form for ZX-Diagrams and Application to the Rational Angle Completeness". In: *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 2019. eprint: 1805.05296.

[56] Renaud Vilmart. "A ZX-Calculus with Triangles for Toffoli-Hadamard, Clifford+ T, and Beyond". In: *QPL 2018*. Vol. 287. 2018, pp. 313–344.

[57] Quanlong Wang. "Completeness of algebraic ZX-calculus over arbitrary commutative rings and semirings". In: *arXiv preprint arXiv:1912.01003* (2019).

[58] Hector Miller-Bakewell. "Entanglement and Quaternions: The graphical calculus ZQ". In: *arXiv preprint arXiv:2003.09999* (2020).

[59] Miriam Backens. "The ZX-Calculus is Complete for Stabilizer Quantum Mechanics". In: *New Journal of Physics* 16.9 (2014), p. 093021. DOI: 10.1088/1367-2630/16/9/093021. URL: https://doi.org/10.1088%2F1367-2630%2F16%2F9%2F093021.

[60] Miyamoto Musashi. *The Complete Musashi: The Book of Five Rings and Other Works: The Definitive Translations of the Complete Writings of Miyamoto Musashi–Japan's Greatest Samurai*. Tuttle Publishing, 2018.

[61] A. Kissinger and John van de Wetering. *PyZX*. 2018. URL: https://github.com/Quantomatic/pyzx.

[62] Brendan Fong and David I Spivak. "Hypergraph categories". In: *Journal of Pure and Applied Algebra* 223.11 (2019), pp. 4746–4777.

[63] *Sagesses Vosgiennes*. Presses Universitaires Spinalliennes, 2021.

[64] Bob Coecke and Ross Duncan. "Interacting quantum observables: categorical algebra and diagrammatics". In: *New Journal of Physics* 13.4 (2011), p. 043016.

[65] Amar Hadzihasanovic. "ZW calculi: diagrammatic languages for pure-state quantum computing". In: *Logic and Applications LAP 2018* (2018), p. 13.

[66] Miriam Backens and Aleks Kissinger. "ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity". In: Proceedings of the 15th International Conference on *Quantum Physics and Logic,* Halifax, Canada, 3-7th June 2018. Ed. by Peter Selinger and Giulio Chiribella. Vol. 287. Electronic Proceedings in Theoretical Computer Science. Open Publishing Association, 2019, pp. 23–42. DOI: 10.4204/EPTCS.287.2.

[67] Quanlong Wang, Simon Perdrix, and Miriam Backens. "Towards a Minimal Stabilizer ZX-calculus". In: *Logical Methods in Computer Science* 16 (2020).

[68] Quanlong Wang. "Qutrit ZX-calculus is complete for Stabilizer Quantum Mechanics". In: *arXiv preprint arXiv:1803.00696* (2018).

[69] Jamie Vicary. "Topological structure of quantum algorithms". In: *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*. IEEE Computer Society. 2013, pp. 93–102.

[70] André Ranchin. "Depicting qudit quantum mechanics and mutually unbiased qudit theories". In: *arXiv preprint arXiv:1404.1288* (2014).

[71] Stefano Gogioso and Aleks Kissinger. "Fully graphical treatment of the quantum algorithm for the Hidden Subgroup Problem". In: *arXiv preprint arXiv:1701.08669* (2017).

[72] Quanlong Wang and Xiaoning Bian. "Qutrit dichromatic calculus and its universality". In: *arXiv preprint arXiv:1406.3056* (2014).

[73] Aleks Kissinger. "Pictures of processes: automated graph rewriting for monoidal categories and applications to quantum computing". In: *arXiv preprint arXiv:1203.0202* (2012).

[74] Amar Hadzihasanovic. "The algebra of entanglement and the geometry of composition". In: *arXiv preprint arXiv:1709.08086* (2017).

[75] Joseph Collins and Ross Duncan. "Hopf-Frobenius algebras and a simpler Drinfeld double". In: *Electronic Proceedings in Theoretical Computer Science* (2019).

[76] J.S. Ponizovskii. "Semigroup Rings". In: *Semigroup Forum* 36 (1987), pp. 1–46.

[77] M Koppinen. "On algebras with two multiplications, including Hopf algebras and Bose–Mesner algebras". In: *Journal of Algebra* 182.1 (1996), pp. 256–273.

[78] Yukio Doi and Mitsuhiro Takeuchi. "BiFrobenius algebras". In: *Contemporary Mathematics* 267 (2000), pp. 67–98.

[79] E Study. "Über Systeme complexer Zahlen und ihre Anwendung in der Theorie der Transformationsgruppen". In: *Monatshefte für Mathematik und Physik* 1 (1890), pp. 283–354.

[80] Khadra Dekkar and Abdenacer Makhlouf. "Bialgebra structures of 2-associative algebras". In: *arXiv preprint arXiv:0809.1144* (2008).

[81] Amar Hadzihasanovic, Kang Feng Ng, and Quanlong Wang. "Two Complete Axiomatisations of Pure-state Qubit Quantum Computing". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. Oxford, United Kingdom: ACM, 2018, pp. 502–511. ISBN: 978-1-4503-5583-4. DOI: 10.1145/3209108.3209128. URL: http://doi.acm.org/10.1145/3209108.3209128.

[82] Teimuraz Pirashvili. "On the *PROP* corresponding to bialgebras". In: *Cahiers de topologie et géométrie différentielle catégoriques* 43.3 (2002), pp. 221–239.

[83] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. "Interacting Hopf algebras". In: *Journal of Pure and Applied Algebra* 221.1 (2017), pp. 144–184.

[84] Directed by someone. *The wonderful adventures of Simon Perdrix IV: Simon Perdrix and the wild diagrams*. ZX Studio, 2021.

[85] Peter Selinger. "Towards a Quantum Programming Language". In: *Mathematical. Structures in Comp. Sci.* 14.4 (2004), pp. 527–586. ISSN: 0960-1295. DOI: 10.1017/S0960129504004256. URL: https://doi.org/10.1017/S0960129504004256.

[86] Bob Coecke. "Axiomatic Description of Mixed States from Selinger's CPM-Construction". In: *Electronic Notes in Theoretical Computer Science* 210 (2008). Proceedings of the 4th International Workshop on Quantum Programming Languages (QPL 2006), pp. 3 –13. ISSN: 1571-0661. DOI: https://doi.org/10.1016/j.entcs.2008.04.014. URL: http://www.sciencedirect.com/science/article/pii/S1571066108002296.

[87] Bob Coecke and Chris Heunen. "Pictures of complete positivity in arbitrary dimension". In: *Information and Computation* 250 (2016), pp. 50–58.

[88] Bob Coecke and Simon Perdrix. "Environment and Classical Channels in Categorical Quantum Mechanics". In: *Logical Methods in Computer Science* Volume 8, Issue 4 (2012). DOI: 10.2168/LMCS-8(4:14)2012. URL: https://lmcs.episciences.org/719.

[89] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017. DOI: 10.1017/9781316219317.

[90] Mathieu Huot and Sam Staton. "Universal Properties in Quantum Theory". In: *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018*. Ed. by Peter Selinger and Giulio Chiribella. Vol. 287. Electronic Proceedings in Theoretical Computer Science. 2019, pp. 213–223. DOI: 10.4204/EPTCS.287.12.

[91] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[92] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. "A Complete Axiomatisation of the ZX-Calculus for Clifford+T Quantum Mechanics". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. Oxford, United Kingdom: ACM, 2018, pp. 559–568. ISBN: 978-1-4503-5583-4. DOI: 10.1145/3209108.3209131. URL: http://doi.acm.org/10.1145/3209108.3209131.

[93] Aleks Kissinger and Sander Uijlen. "A categorical semantics for causal structure". In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. 2017, pp. 1–12.

[94] Michael Barr and Charles Wells. *Toposes, Triples and Theories*. Springer-Verlag, New York, 1985. URL: http://www.tac.mta.ca/tac/reprints/articles/12/tr12abs.html.

[95] Fabio Zanasi. "Interacting Hopf Algebras – the theory of linear systems". PhD thesis. Université de Lyon, 2015. URL: http://www.zanasi.com/fabio/#/publications.html.

[96] Daniel Marsden. "A graph theoretic perspective on CPM (Rel)". In: *arXiv preprint arXiv:1504.07003* (2015).

[97] Stefano Gogioso. "A Bestiary of Sets and Relations". In: *arXiv preprint arXiv:1506.05025* (2015).

[98] Koenraad M. R. Audenaert and Martin B. Plenio. "Entanglement on Mixed Stabilizer States: Normal Forms and Reduction Procedures". In: *New Journal of Physics* 7 (2005), pp. 170–170. DOI: 10.1088/1367-2630/7/1/170. URL: https://doi.org/10.1088%2F1367-2630%2F7%2F1%2F170.

[99] Brett Giles and Peter Selinger. "Exact Synthesis of Multiqubit Clifford+T Circuits". In: *Phys. Rev. A* 87 (3 2013), p. 032332. DOI: 10.1103/PhysRevA.87.032332. URL: https://link.aps.org/doi/10.1103/PhysRevA.87.032332.

[100] Martti Karvonen. "The Way of the Dagger". In: *arXiv e-prints* (2019), arXiv–1904.

[101] Aleks Kissinger and David Quick. "A first-order logic for string diagrams". In: *arXiv preprint arXiv:1505.00343* (2015).

[102] Vladimir Zamdzhiev. "A Framework for Rewriting Families of String Diagrams". In: *Proceedings Tenth International Workshop on Computing with Terms and Graphs, TERM-GRAPH@FSCD 2018, Oxford, UK, 7th July 2018*. Ed. by Maribel Fernández and Ian Mackie. Vol. 288. EPTCS. 2018, pp. 63–76. DOI: 10.4204/EPTCS.288.6. URL: https://doi.org/10.4204/EPTCS.288.6.

[103] Apiwat Chantawibul and Paweł Sobociński. "Monoidal multiplexing". In: *International Colloquium on Theoretical Aspects of Computing*. Springer. 2018, pp. 116–131.

[104] Titouan Carette, Dominic Horsman, and Simon Perdrix. "SZX-Calculus: Scalable Graphical Quantum Reasoning". In: *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2019.

[105] Scott Aaronson. *Shtetl-Optimized*. URL: https://www.scottaaronson.com/blog/.

[106] Marc Hein et al. "Entanglement in graph states and its applications". In: *Proceedings of the International School of Physics "Enrico Fermi" on "Quantum Computers, Algorithms and Chaos", arXiv:quantph/0602096;* (2006).

[107] Michel Grabisch. "Bases and transforms of set functions". In: *On Logical, Algebraic, and Probabilistic Aspects of Fuzzy Set Theory*. Springer, 2016, pp. 215–231.

[108] Anne Canteaut and Marion Videau. "Symmetric boolean functions". In: *IEEE Transactions on information theory* 51.8 (2005), pp. 2791–2811.

[109] Ethan Bernstein and Umesh Vazirani. "Quantum complexity theory". In: *SIAM Journal on computing* 26.5 (1997), pp. 1411–1473.

[110] L. K. Grover. "Quantum Mechanics Helps in Searching for a Needle in a Haystack". In: *Phys. Rev. Lett.* 79 (1997), p. 325. DOI: 10.1103/PhysRevLett.79.325. eprint: quant-ph/9706033.

[111] Cole Comfort and Aleks Kissinger. "A Graphical Calculus for Lagrangian Relations". In: *CoRR* abs/2105.06244 (2021). arXiv: 2105.06244. URL: https://arxiv.org/abs/2105.06244.

Résumé

Cette thèse concerne l'application de langages graphiques à l'informatique quantique. Par langages graphiques on entend l'usage de diagrammes, très similaires aux circuits, représentant des évolutions de systèmes quantiques. La these introduit ces langages dans le formalisme de la théorie des catégories et s'intéresse en particulier à un langage: le ZX-calcul, ainsi qu'à ses proches parents le ZW-calcul et le ZH-calcul. La notion de flexsymétrie est introduite, décrivant des diagrammes dont les entrés et sorties sont toutes interchangeables entre elles. La notion est ensuite utilisée pour classifier tous les langages similaires au ZX-calcul. Il est montré que les seuls langages admissibles sont le ZX-calcul, le ZW-calcul et le ZH-calcul. Ensuite est abordée la question de l'extension de ces langages au cas de systèmes mixtes classiques-quantiques. Une construction catégorique générale est proposée et est utilisée pour étendre les différents langages. Enfin la thèse introduit des notations permettant de représenter de manière compacte des algorithmes quantiques mettant en jeux des diagrammes arbitrairement grands. Afin d'en éprouver l'efficacité, ces notations sont utilisées pour montrer graphiquement la correction de différents algorithmes quantiques.

Mots-clés: Informatique quantique, Théorie des catégories, Langages graphiques, ZX-calcul.

Abstract

This thesis is about the application of graphical languages to quantum computing. By graphical language, we mean the use of diagrams, similar to circuits, representing the evolution of quantum systems. The thesis introduces those languages in the formalism of category theory and focuses mainly on one language: the ZX-calculus, and its close relatives, the ZW-calculus and ZH-calculus. The notion of flexsymmetry is introduced, describing diagrams whose inputs and outputs are all interchangeable. This notion is used to classify all languages similar to the ZX-calculus. It is shown that the only admissible languages are the ZX-calculus, the ZH-calculus, and the ZW-calculus. Then is tackled the question of extending those languages to mixed-state quantum mechanics. A general categorical construction is proposed and is applied to provide extensions of the different languages. Finally, the thesis introduces notations allowing to handle in a compact way quantum algorithms relying on arbitrary large diagrams. To challenge their efficiency, those notations are used to show the correction of various quantum algorithms.

Keywords: Quantum computing, Category theory, Graphical languages, ZX-calculus.