ZX on Ribbons / on Many-Worlds

Kostia Chardonnet
Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, 91190, Gif-sur-Yvette, France.
Université Paris Cité, CNRS, IRIF, F-75006, Paris, France

Marc de Visme
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria, LMF, 91190, Gif-sur-Yvette, France

Benoît Valiron
Université Paris-Saclay, CNRS, CentraleSupélec, ENS Paris-Saclay, LMF, 91190, Gif-sur-Yvette, France

Renaud Vilmart
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria, LMF, 91190, Gif-sur-Yvette, France

Abstract

We propose a new typed graphical language for quantum computation, based on compact categories with biproduct. Our language generalizes existing approaches such as ZX calculus and quantum circuits, while offering a natural framework to support quantum control: it natively supports "quantum tests".
The language comes equipped with a denotational semantics based on linear applications, and an equational theory. Through the use of normal forms for the diagrams, we prove the language to be universal, and the equational theory to be complete with respect to the semantics.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Linear logic; Theory of computation → Equational logic and rewriting

Keywords and phrases Quantum Computation, Linear Logic, Graphical Languages, Geometry of Interaction

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

Conventional wisdom has it that quantum computation is about *quantum data in superposition*. In the standard model, the memory holding quantum data is encapsulated inside a co-processor accessed through a simple interface: The co-processor holds individually addressable registers holding *quantum* bits, on which one can apply a fixed set of operations —*gates*— specified by the interface. If some of these gates can generate superposition of data, this is kept inside the co-processor and opaque to the programmer. A typical interaction with the co-processor is a purely classical sequence of elementary operations of the form "Apply gate X to register n ; apply gate Y to register m; *etc*". Such a sequence of instructions is usually represented as a *quantum circuit*. In this model, a quantum program is then a conventional program building a quantum circuit and sending it as a batch-job to the co-processor.

From a semantical perspective, the *state* of a quantum memory consisting of n quantum bits is a vector in a 2^n-dimensional Hilbert space. A quantum circuit is a linear, sequential description of elementary operations describing a *linear, unitary map* on the state space. The quantum co-processor should come with a *universal* set of such operations, that is, such that any unitary operation on the state space can be realized using the given elementary gates.

If unitary maps form the original representation for (pure) quantum operations, this extensional presentation makes a very rough semantics for the interaction with the quantum memory. It is akin to say that one could abstract a conventional program by the graph of the corresponding function: this hides many of the useful informations one might want to keep track of, such as topological constraints, composition of subroutines and execution flow, required resources or other costs, *etc* [29]. Most of the *structure* of the computation is lost in the matrix representation.

Coming all the way from Feyman's diagrams [19], graphical languages for representing quantum processes can be seen as an answer to the limitations of plain unitary matrices. In recent year, in the field of quantum computation has blossomed a wide variety of proposals to focus on specific aspects of the semantics of quantum programs [20, 12, 6].

Quantum circuits are an obvious candidate for a graphical language, and indeed, several lines of research took them as their main object of study [21, 13, 26, 5]. Quantum circuits in particular form a natural medium for describing the execution flow of a computation. Similar to the intuition one can build for classical boolean circuits, one can make an operational semantics for quantum circuits using a token-machine Geometry of Interaction. Each token in the circuit carries a quantum bit, and they flow in the circuit from inputs to outputs while getting modified as they pass through gates [13]. The main problem with the model of quantum circuits is the lack of a satisfactory equational presentation. If several attempts have been made for various subsets [10, 9, 22, 23], none of them provides a complete presentation.

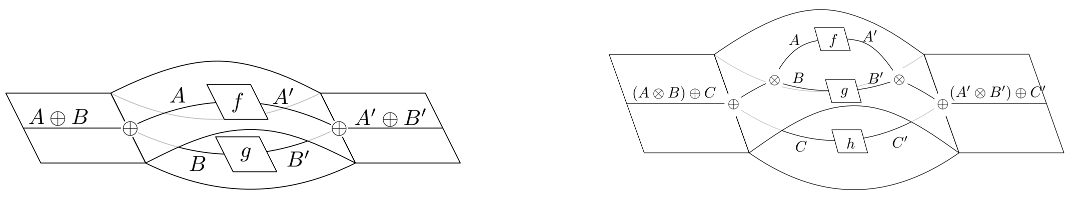
A recent proposal answering the defects of quantum circuits as a (formal) language is the ZX-calculus [12]. Rooted in category theory [1], it comes with a small set of generators and a sound and complete equational theory [32]. Conservative extension of quantum circuits, yet versatile, the ZX-calculus has shown its use in a wide variety of applications, ranging from optimization [17, 3], verification [18, 16], and error-correction [14].

Despite its success, the ZX calculus is nonetheless still tied to the quantum co-processor model. Indeed, it fails to account for one peculiar feature of quantum computation: non-causal execution paths. Indeed, the Janus-faced quantum computational paradigm features two seemingly distinct notions of control structure. One one hand, a quantum program follows *classical* control: it is hosted on the conventional computer governing the co-processor, and can therefore only enjoy loops, tests and other regular causally ordered sequences of operations. On the other hand, the lab bench turns out to be more flexible than the rigid co-processor model, permitting more elaborate *purely quantum* computational constructs than what quantum circuits or ZX calculus allow.

The archetypal example of an quantum computational behavior hardly attainable within ZX-calculus (or quantum circuits) is the *quantum SWITCH*. Consider two quantum bits x and y and two unitary operations U and V acting on y. The problem consists in generating the operation that performs UV on y if x is in state $|0\rangle$ and VU if it is in state $|1\rangle$. As x can be in superposition, in general the operation is then sending $(\alpha\,|0\rangle + \beta\,|1\rangle) \otimes |y\rangle$ to $\alpha\,|0\rangle \otimes (UV\,|y\rangle)_x + \beta\,|1\rangle \otimes (VU\,|y\rangle)$. It is a *purely quantum test*: not only can we have values in superpositions (here, x) but also *execution orders*. This is in sharp contrast with what happens within the standard quantum co-processor model.

Computational models supporting superpositions of execution orders have been studied in the literature. One trend of research consists in proposing a suitable extension of quantum circuits [7, 27, 31, 33]. These approaches typically aim at discussing the notion of quantum channel from a quantum information theoretical standpoint. Another research line is concerned with exhibiting the execution flow internal to ZX-terms in a distributed manner [4]. Somewhat similarly to what has been proposed in [13], tokens are let to flow in a ZX term, realizing the computation, the difference being that the tokens can themselves be in superposition. The limit is however that a ZX term is inherently *causal*, therefore restricting the possibilities for superposition of orders.

One limitation of the existing extensions of quantum circuits or of graphical languages



(a) Split over co-product **(b)** Splits over co-product and tensor

Figure 1 Examples of splits

based on ZX calculus is the limited support for *typed quantum data*. In the context of quantum circuits, the strategy have been to consider extended structures steming from proofs-nets and resource-sensitive logics such as linear logic. A proof-net is a structured graph describing a proof of a linear logic formula. Through the Curry-Howard isomorphism, a proof can be regarded as a program and a formula as a type specifying how the program handles resources. This turns out to be particularly well-suited for quantum computation [15, 25, 13].

In the approaches merging quantum circuits with types based on linear logic, the *additive types* such as the sum-type are *purely classical*: $1 \oplus 1$ is always a (conventional) boolean. In quantum computation, the sum-type $1 \oplus 1$ can can however be understood as a *sum of vector spaces*, giving an alternative interpretation to $1 \oplus 1$: it can be regarded as the type of a quantum bit, superposition of True and False. One should note that this appealing standpoint should be taken cautiously: (Pure) quantum information imposes strong constraints on the structure of the data in superposition: orthogonality and unit-norm have to be preserved [2, 28].

The standard categorical formalization underlying quantum computation, whether with the co-processor or with the purely quantum viewpoint, is symmetric monoidal categories, and compact closed categories in particular. The monoidal structure stands for the tensor of quantum data, while co-products are used to represent "tests", whether classical [13, 15] or quantum [28].

Graphical languages for symmetric monoidal structure with co-products usually rely on a notion of *sheet*, or *worlds*, to handle tests and co-products in general [15, 24]. Figure 1a shows for instance how to represent the construction of the morphism $f \oplus g : A \oplus A' \to B \oplus B'$ out of $f : A \to B$ and $g : A' \to B'$. The symbol "$\oplus$" stands for the "split" of worlds. Such a graphical language therefore comes with two distinct "splits": one for the monoidal structure —leaving inside one specific world—, and one for the co-product —splitting worlds—. They can be intertwined, as shown in Figure 1b. Another approach followed by [**?**] internalizes the two products (tensor product and co-product) into the structure of the diagrams themselves, at the price of a less intuitive tensor product and a form of synchronization constraint.

However, in the state of the art this "splitting-world" understanding has only been carried for *classical* tests [13] and probabilistic branching stemming from measurements [15, 30], and *not* for quantum superposition. The quantum SWITCH can for instance be naturally understood in this framework. Consider for instance Figure 2, read from left to right: as input, a pair of an element of type A and a quantum bits. Based on the value of the qubit (True of False), the wire A goes in the upper or the lower sheet, and is fed with U then V or V then U. Then everything is merged back together.

In this paper, we provide a rigorous interpretation for this type of (purely quantum) behavior.

Contributions In this paper, we introduce a new graphical language for quantum computation, based on compact category with biproduct. This language allows us to express

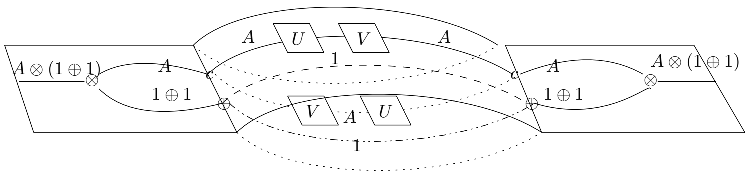


Figure 2 Quantum SWITCH with Worlds

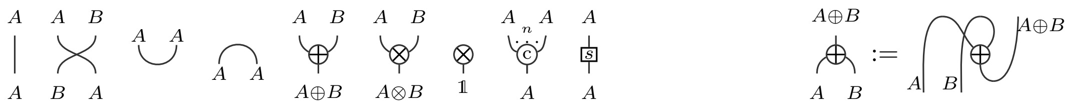


Figure 3 Generators of our First Graphical Language ($n \geq 0$, $s \in \mathbb{C}$)

any quantum process, as we can encode the ZX-Calculus within it. We develop a denotational semantic and an equational theory, and prove the soundness and completeness of the semantics. As a case-study, we show how the Quantum Switch can naturally be encoded in the language. In the paper, the missing proofs can be found in Appendix.

2 The Many-Worlds Calculus

While the goal is to define a graphical language in which each wire can be enabled or disabled depending on the world in which the computation takes place, we first define the category $\mathbf{C_D}$ of diagrams without any "worlds", and will then add the world annotations.

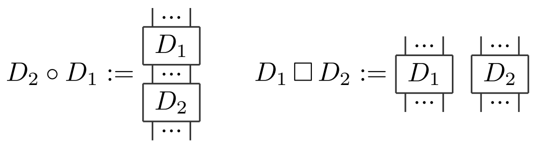
2.1 A First Graphical Language

We define our graphical language within the paradigm of colored PROP [TODO: CITE], meaning that a diagram will be composed of nodes, or *generators*, linked to each others through *typed wires*, wired that are allowed to cross each others. Additionally, we assume that our colored PROP is compact closed and auto-dual, meaning that we allow to curve wires to obtain a Cup or a Cap.

The generators of our language are described in Figure 3 and are respectively the Identity, the Swap, the Cup, the Cap, the Plus, the Tensor, the Unit, the n-ary Contraction, and the Scalar indexed with s ranging over $\mathbb{C}$. Mirrored versions of those generators are defined as syntactic sugar through the compact closure, as shown for the mirrored Plus on the right-hand-side of Figure 3. Diagrams are read top-to-bottom: the top-most wires are the *input* wires and the bottom-most wires are the *output* wires.

The types of our wires are built from the syntax $A, B ::= \mathbb{1} \mid A \oplus B \mid A \otimes B$. The objects of our category are the set of wires as generated by the grammar $\mathfrak{A}, \mathfrak{B} ::= \mathfrak{A} \square \mathfrak{B} \mid \boxslash \mid A$. The choice of the notation $\square$ for wires in parallel is uncommon, we use it to put an emphasis on the fact that contrary to languages like the ZX-calculus, wires that are in parallel are not necessarily "in tensor with one another". In fact, $A \square B$ can be understood semantically as "either $A \otimes B$ or $A \oplus B$".

Diagrams are obtained from generators by composing them in parallel (written $\square$), or sequentially (written $\circ$) as follows. Sequential composition requires the type (and number) of wires to match. We write $\mathbf{C_D}$ for the category of diagrams we defined as such.



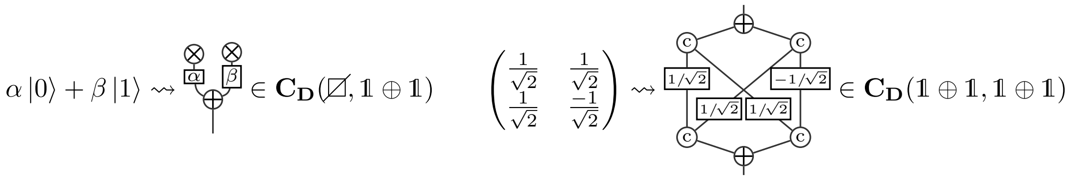


Figure 4 A quantum bit and the Hadamard unitary

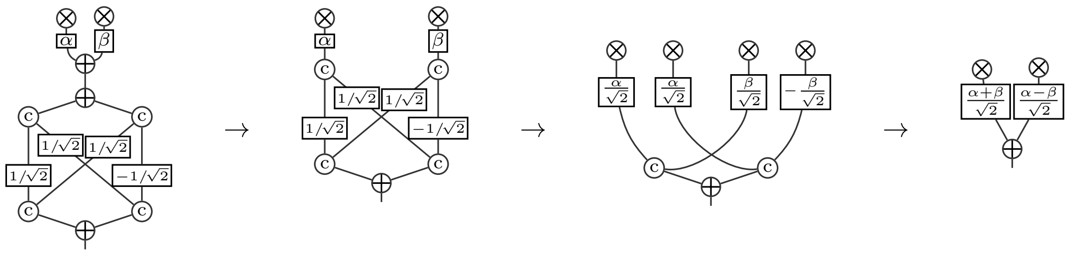
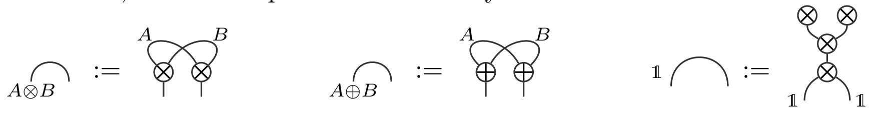


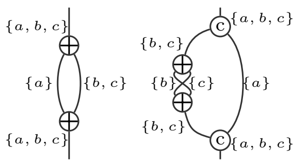
Figure 5 Applying the Hadamard unitary to a quantum bit

▶ **Example 1.** While our language lacks the worlds labeling, we can already illustrate it by encoding some basic quantum primitive in it and show how they operate. In Figure 4 we show the encoding of a quantum bit $\alpha\,|0\rangle + \beta\,|1\rangle$ and the Hadamard unitary. In particular, the Plus allows to "build" a new quantum bit from two scalars in parallel or to "open" a quantum bit to recover its corresponding scalars, the left branch corresponding to $|0\rangle$ and the right branch to $|1\rangle$. The meaning of the Contraction is better seen when applying Hadamard to a quantum bit as we show in Figure 5, it allows us to duplicate and sum scalars. The rewriting sequence of Figure 5 is made using the equational theory defined in Section 4, however to correctly define our equational theory properly, the worlds labeling are required. So while this specific worlds-free rewriting sequence is sound, many other similar worlds-free rewriting sequences are unsound.

▶ Remark 2. Instead of having the Cup and the Cap as generators and defining the mirrored version of each generator through them, one could proceed the other way around by defining the Cap as follows, and the Cup in a mirrored way:

176 

2.2 Adding Worlds Labeling

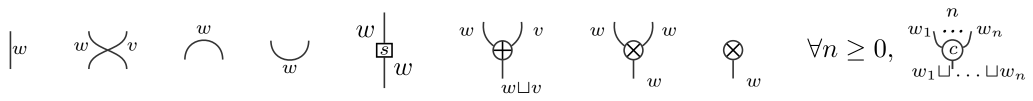


We now label wires of our diagram with worlds sets $w \subseteq W$ for a given a set of worlds W. For each world $a \in W$, wires labeled by a set containing a are said to be "enabled in a", and the others are said "disabled in a". This allows to correlate the enabling of wires. For example, the "controlled not" can be represented by the diagram on the right, with worlds $\{a, b, c, \star\}$. The left-hand-side of the diagram forces the world a to correspond to the case where the control qubit is equal to $|0\rangle$, and the worlds b and c when it is equal to $|1\rangle$. The right-hand-side of the diagram applies the identity in the world a, and a negation in the worlds b and c. Lastly, the world $\star$ appears nowhere in the labels, and corresponds to "we do not evaluate this circuit at all". While not strictly necessary, it is often practical to have a world absent from every wire.

▶ **Definition 3.** Given a set of worlds W, we define the auto-dual compact closed colored PROP $(\mathbf{MW}_W, \square, \varnothing)$ of many-worlds calculus over W as follows:

Its colors are the pairs $(A : w)$ of colors A of $\mathbf{C_D}$ and subsets $w \subseteq W$. We write $(\mathfrak{A}, \ell_{\mathfrak{A}})$ for the objects, where $\mathfrak{A}$ is an object of $\mathbf{C_D}$ and ℓ_A is a labeling function from the colors of $\mathfrak{A}$ to the subsets of W.

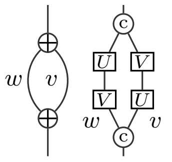
Its morphisms $f : (\mathfrak{A}, \ell_{\mathfrak{A}}) \to (\mathfrak{B}, \ell_{\mathfrak{B}}))$ are pairs $(\mathcal{D}_f, \ell_f)$ of a morphism $\mathcal{D}_f \in \mathbf{C_D}(\mathfrak{A}, \mathfrak{B})$ and a labeling function ℓ_f from the wires of $\mathcal{D}_f$ to the subsets of W, satisfying the following constraints: The label on an input or output wire of color $(A : w)$ must be equal to w, and



where $\sqcup$ denotes a set union which is disjoint. The constraints for the mirrored versions are similar. The sequential composition $\circ$ and the parallel composition $\square$ preserve the labels.

Because of the first restriction on labels, given a morphism f, one can infer the labels on its objects from the label ℓ_f, hence we can write $f : \mathfrak{A} \to \mathfrak{B}$ instead of $f : (\mathfrak{A}, \ell_{\mathfrak{A}}) \to (\mathfrak{B}, \ell_{\mathfrak{B}})$ unambiguously. It is discussed in Appendix B.1 how we can compose $f : \mathfrak{A} \to \mathfrak{B}$ and $g : \mathfrak{B} \to \mathfrak{C}$ even when their induced labels do not match.

▶ **Example 4** (The Quantum Switch). Given two subdiagrams U and V, the Quantum Switch of $U; V$ and $V; U$ presented in Figure 2 can be encoded as follows:



The figure is split into two parts : the control part on the left-hand-side, and the computational part on the right-hand-side. The idea is that the control part, that uses $\oplus$ will behave as an *if-then-else* and will bind the world w to the case where the quantum control bit is $|0\rangle$, and the world v to the case where the quantum control bit is $|1\rangle$. On the computational part, $U \circ V$ will be applied within the world v while $V \circ U$ will be applied within the world w.

The "sheets" presented in Figure 2 are here modeled with world labels.

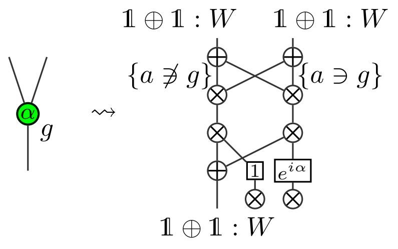
For the sake of simplicity, we represented the quantum switch here with two copies of U and V, one for each branch. It is actually possible within our language to share U and V between the two worlds in order to only have one copy of each (which is the while purpose of the quantum switch). We describe in Example 8 this "true" Quantum Switch, and one can show using the equational theory given in Section 4 that the two are equivalent.

2.3 Comparison with Other Graphical Languages

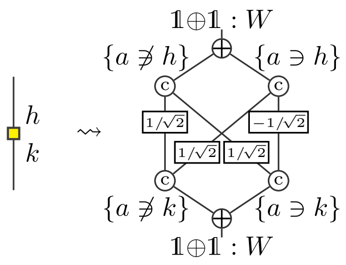
We draw some comparison between our language and two other graphical languages: the ZX-Calculus [11], and Duncan's Tensor-Sum Logic [15].

2.3.1 ZX-Calculus

The first difference is the restrictions of the ZX-calculus to computations between qubits, in other words linear map from $\mathbb{C}^{2^n} \mapsto \mathbb{C}^{2^m}$, while our language can encode any linear map from $\mathbb{C}^n \mapsto \mathbb{C}^m$. The Tensor generator allowing to decompose $\mathbb{C}^{2^n}$ into two instances of $\mathbb{C}^2$ was already present in the *scalable* extension of the ZX-calculus [?] , but the main differences come from the Plus (and the Contraction).



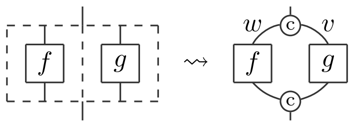
They allows us to reason about *quantum-control* in an direct way, as shown above for the Quantum Switch, which is not as easily doable in ZX-Calculus.



Additionally, every ZX diagram can be encoded in our graphical language. To encode a diagram of the ZX-calculus, we start by writing it with only Hadamard gates and green nodes, and then we encode each generator independently as described on the right. The worlds set W is equal to $\mathcal{P}(\text{Names})$ where each green node of the ZX-diagram is given a name, and each input and each output of an Hadamard gate is given a name too. For a name g, we write $\{a \ni g\} \subseteq W$ for the set $\{a \subseteq \text{Names} \mid g \in a\}$ and $\{a \not\ni g\}$ for its complementary.

2.3.2 Tensor-Sum Logic

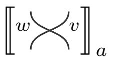
The core difference between their work and ours is the presence of the contraction in our graphical language. They instead rely on an enrichment of their category by a sum, which they represent graphically with boxes. We show on the right how the morphism $f + g$ would be encoded in both their and our language. More generally, their boxes correspond to uses of our contraction generator in a "well-bracketed" way. Another point of difference is their approach to quantum computation, as we do not assign the same semantics to those superposition of morphisms. In their approach, the superposition is a classical construction and corresponds to the measurement and the classical control flow, while in our approach the superposition is a quantum construction and corresponds to the quantum control.



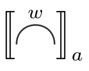
3 Semantics of the Many-Worlds Calculus

Our calculus aims at pure quantum computations. Following the ZX approach, we relax the condition on unitarity and define a semantics for our Many-World Calculus based on finite dimension Hilbert spaces and (general) linear operators. More precisely, we will define two semantics, a world-dependent semantics $[\![-]\!]_a$ for every world a, which will be a monoidal functor from $\mathbf{MW}_W$ to $\mathbf{FdHilb}$, and a worlds-agnostic semantics $[\![-]\!]$ which will not be functorial for the standard sequential composition and parallel composition, though we define in Appendix B.1 the worlds-agnostic sequential composition and parallel composition for which $[\![-]\!]$ will be functorial.

We start by defining those semantics on the objects. For every object $\mathfrak{A}$ of $\mathbf{C_D}$, we define its *enablings* $\mathfrak{A}^\bullet$ as "replacing any number of wire type by $\bullet$". For example $(A \,\square\, B)^\bullet = \{\bullet \,\square\, \bullet, \bullet \,\square\, B, A \,\square\, \bullet, A \,\square\, B\}$. To each enabling $\mathfrak{C} \in \mathfrak{A}^\bullet$ we associate a Hilbert space $\mathcal{H}_\mathfrak{C}$ as follows: $\mathcal{H}_{\mathfrak{C} \,\square\, \mathfrak{F}} := \mathcal{H}_\mathfrak{C} \otimes \mathcal{H}_\mathfrak{F}$, $\mathcal{H}_{\not\square} = \mathcal{H}_\bullet = \mathcal{H}_{1\!\!1} := \mathbb{C}$, $\mathcal{H}_{A \otimes B} := \mathcal{H}_A \otimes \mathcal{H}_B$, $\mathcal{H}_{A \oplus B} := \mathcal{H}_A \oplus \mathcal{H}_B$. Then, for any object $(\mathfrak{A}, \ell_\mathfrak{A})$ of $\mathbf{MW}_W$ and any world $a \in W$, we define $(\mathfrak{A}, \ell_\mathfrak{A}) \Downarrow a$ to be the enabling of $\mathfrak{A}$ in which every $(A : w)$ with $a \in w$ is preserved and every $(A : w)$ with $a \notin w$ is replaced by $\bullet$. For example $(A : \{a\} \,\square\, B : \{b\}) \Downarrow a = A \,\square\, \bullet$. We can then define the semantics $[\![-]\!]_a : \mathbf{MW}_W \to \mathbf{FdHilb}$ and $[\![-]\!] : \mathbf{MW}_W \to \mathbf{FdHilb}$ on objects as $[\![(\mathfrak{A}, \ell_\mathfrak{A}))]\!]_a := \mathcal{H}_{(\mathfrak{A}, \ell_\mathfrak{A}) \Downarrow a}$ and $[\![(\mathfrak{A}, \ell_\mathfrak{A})]\!] := \bigoplus_{\mathfrak{C} \in \mathfrak{A}^\bullet} \mathcal{H}_\mathfrak{C}$. Then, for the morphisms, we proceed by compositionality for $[\![-]\!]_a$, meaning that we define $[\![-]\!]_a$ on every generator and compute the semantics of a diagram by decomposing it with $[\![g \circ f]\!]_a := [\![g]\!]_a \circ [\![f]\!]_a$ and $[\![f \,\square\, g]\!]_a := [\![f]\!]_a \otimes [\![g]\!]_a$. When looking at a generator $gen \in \mathbf{MW}_W((\mathfrak{A}, \ell_\mathfrak{A}), (\mathfrak{B}, \ell_\mathfrak{B}))$, in most

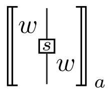


$$[\![\;]\!]_a = \begin{cases} \mathrm{Id} & \in \mathbf{FdHilb}(\mathcal{H}_A, \mathcal{H}_A) & \text{if } a \in w \backslash v \\ \mathrm{Id} & \in \mathbf{FdHilb}(\mathcal{H}_B, \mathcal{H}_B) & \text{if } a \in v \backslash w \\ h \otimes h' \mapsto h' \otimes h & \in \mathbf{FdHilb}(\mathcal{H}_{A\otimes B}, \mathcal{H}_{B\otimes A}) & \text{if } a \in w \cap v \\ (1) & \in \mathbf{FdHilb}(\mathbb{C}, \mathbb{C}) & \text{otherwise} \end{cases}$$

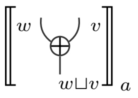


$$[\![\;]\!]_a = \begin{cases} h \otimes h' \mapsto \langle h | h' \rangle & \in \mathbf{FdHilb}(\mathcal{H}_{A\times A}, \mathbb{C}) & \text{if } a \in w \\ (1) & \in \mathbf{FdHilb}(\mathbb{C}, \mathbb{C}) & \text{otherwise} \end{cases}$$

and the (conjugate) transposed operator for the Cup



$$[\![\;]\!]_a = \begin{cases} (s) & \in \mathbf{FdHilb}(\mathbb{C}, \mathbb{C}) & \text{if } a \in w \\ (1) & \in \mathbf{FdHilb}(\mathbb{C}, \mathbb{C}) & \text{otherwise} \end{cases}$$



$$[\![\;]\!]_a = \begin{cases} \begin{pmatrix} \mathrm{Id} \\ 0 \end{pmatrix} & \in \mathbf{FdHilb}(\mathcal{H}_{A\oplus B}, \mathcal{H}_A) & \text{if } a \in w \\ \begin{pmatrix} 0 \\ \mathrm{Id} \end{pmatrix} & \in \mathbf{FdHilb}(\mathcal{H}_{A\oplus B}, \mathcal{H}_B) & \text{if } a \in v \\ (1) & \in \mathbf{FdHilb}(\mathbb{C}, \mathbb{C}) & \text{otherwise} \end{cases}$$

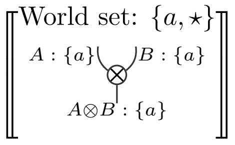
and the (conjugate) transposed operator for the mirrored Plus

Figure 6 Semantics of the generators of $\mathbf{MW}_W$ in a world $a \in W$.

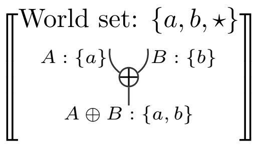
cases $[\![(\mathfrak{A}, \ell_{\mathfrak{A}})]\!]_a = [\![(\mathfrak{B}, \ell_{\mathfrak{B}})]\!]_a = H$: we then define $[\![gen]\!]_a$ as the identity of $\mathbf{Hilb}(H, H)$. We list in Figure 6 all the cases where the semantics of a generator is not the identity.

The worlds-agnostic semantics is defined from the world-dependent semantics, as follows. Consider $f \in \mathbf{MW}_W((\mathfrak{A}, \ell_{\mathfrak{A}}), (\mathfrak{B}, \ell_{\mathfrak{B}}))$. Then $[\![f]\!] \in \mathbf{FdHilb}\left(\bigoplus_{\mathfrak{C}\in\mathfrak{A}^\bullet} \mathcal{H}_{\mathfrak{C}}, \bigoplus_{\mathfrak{F}\in\mathfrak{B}^\bullet} \mathcal{H}_{\mathfrak{B}}\right)$ is defined as $[\![f]\!] := \left\{\sum_{a\in W_{\mathfrak{A},\mathfrak{B},\mathfrak{C},\mathfrak{F}}} [\![f]\!]_a\right\}_{\mathfrak{C}\in\mathfrak{A}^\bullet, \mathfrak{F}\in\mathfrak{B}^\bullet}$, where $W_{\mathfrak{A},\mathfrak{B},\mathfrak{C},\mathfrak{F}}$ is a shortcut notation for the set $\left\{a \in W \mid (\mathfrak{A}, \ell_{\mathfrak{A}}) \Downarrow a = \mathfrak{C} \text{ and } (\mathfrak{B}, \ell_{\mathfrak{B}}) \Downarrow a = \mathfrak{F}\right\}$.

For example, the worlds-agnostic semantics of the Tensor and the Plus are simply the collection of all their world-dependent semantics assembled into a single linear operator:



$$[\![\;]\!] = \begin{array}{c} \\ A\otimes B \\ \bullet \end{array} \begin{array}{cccc} A\square B & A\square\bullet & \bullet\square B & \bullet\square\bullet \\ \end{array} \begin{pmatrix} \mathrm{Id} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



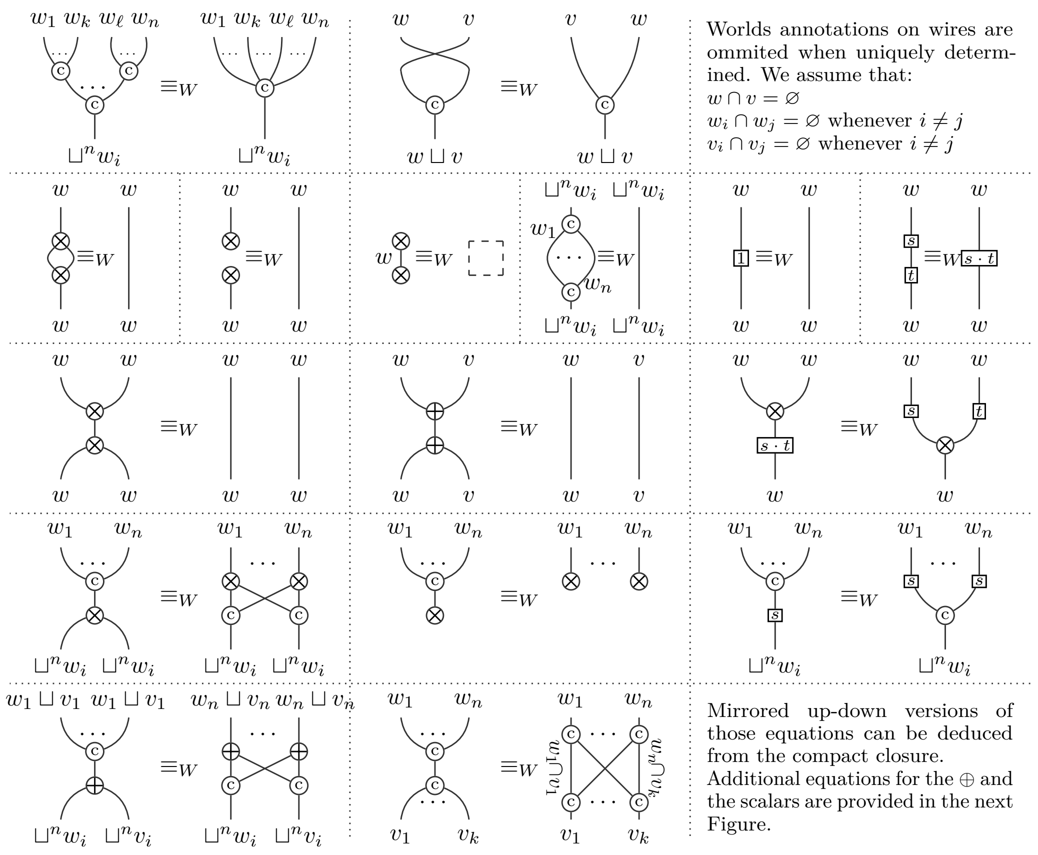
$$[\![\;]\!] = \begin{array}{c|cccc} & A\square B & A\square\bullet & \bullet\square B & \bullet\square\bullet \\ \hline A\oplus B & \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} & \begin{smallmatrix} \mathrm{Id} \\ 0 \end{smallmatrix} & \begin{smallmatrix} 0 \\ \mathrm{Id} \end{smallmatrix} & \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \\ \bullet & 0 & 0 & 0 & 1 \end{array}$$

The worlds-agnostic semantics is universal in the following sense:

▶ **Theorem 5** (Universality). *For every $\mathfrak{A}, \mathfrak{B}$ objects of $\mathbf{C_D}$, and for every linear operator in $U \in \mathbf{FdHilb}\left(\bigoplus_{\mathfrak{C}\in\mathfrak{A}^\bullet} \mathcal{H}_{\mathfrak{C}}, \bigoplus_{\mathfrak{F}\in\mathfrak{B}^\bullet} \mathcal{H}_{\mathfrak{B}}\right)$, there exists a worlds set W, some labelings $\ell_{\mathfrak{A}}, \ell_{\mathfrak{B}}$, and a morphism $f \in \mathbf{MW}_W((\mathfrak{A}, \ell_{\mathfrak{A}}), (\mathfrak{B}, \ell_{\mathfrak{B}}))$ such that $[\![f]\!] = U$.* ◀

4 The Equational Theory

Similarly to how our semantics is defined in two steps, the equational theory is also defined in two steps:


Worlds annotations on wires are ommited when uniquely determined. We assume that:
$w \cap v = \varnothing$
$w_i \cap w_j = \varnothing$ whenever $i \neq j$
$v_i \cap v_j = \varnothing$ whenever $i \neq j$

Mirrored up-down versions of those equations can be deduced from the compact closure. Additional equations for the $\oplus$ and the scalars are provided in the next Figure.

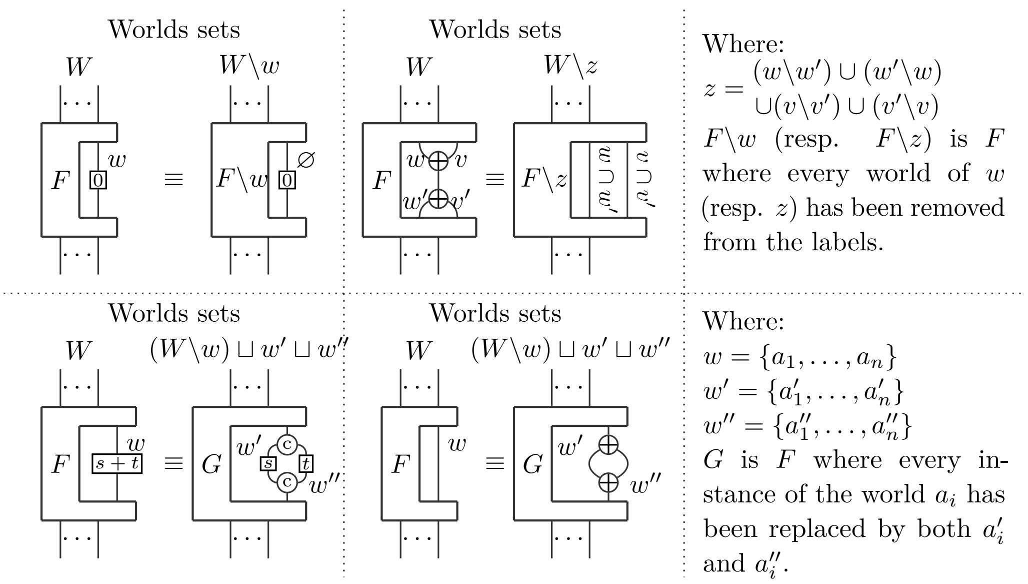
Figure 7 Equations with a Fixed Worlds Set W

(1) A set of equations within $\mathbf{MW}_W$ for a fixed set of worlds W, which will not be complete, but will be sound for $[\![-]\!]_a$ for every $a \in W$, hence sound for $[\![-]\!]$ too. We write $\equiv_W$ for the induced congruence[1] over $\mathbf{MW}_W$. We list those equations in Figure 7. Quite notably, the last two rows describes that, up to the worlds annotations, the contraction is a natural transformation.

(2) Five additional equations with side effects on the set of worlds, which will be sound are complete for $[\![-]\!]$, but not for the $[\![-]\!]_a$. We write $\equiv$ for the induced equivalence relation, spanning over the morphisms of all the categories $\mathbf{MW}_W$ for W any set of worlds. We have: One equation that allows us to rename the worlds: for every morphism $(\mathcal{D}_f, \ell_f)$ of $\mathbf{MW}_W$, and for every bijection $i : W \to V$, we have $(\mathcal{D}_f, \ell_f) \equiv (\mathcal{D}_f, i \circ \ell_f) \in \mathbf{MW}_V$; Two equations allowing the annihilation (or creation, when looking at them from right to left) of worlds due to coproducts or scalars (first row of Figure 8); Two equations allowing the splitting (or merging, when looking at them from right to left) of worlds due to coproducts or scalars (second row of Figure 8).

▶ **Proposition 6** (Soundness). *For f a morphism of $\mathbf{MW}_W$ and g of $\mathbf{MW}_W$, whenever $f \equiv g$ we have $[\![f]\!] = [\![g]\!]$. Additionally if $W = V$, whenever $f \equiv_W g$ we have $\forall a \in W, [\![f]\!]_a = [\![g]\!]_a$.*

[1] In other words the smallest equivalence relation satisfying those equations and such that $f \equiv_W f' \implies \forall g, h, l, g \circ (f \otimes h) \circ k \equiv_W g \circ (f' \otimes h) \circ k$.



Where:
$z = \begin{matrix}(w\backslash w') \cup (w'\backslash w)\\ \cup(v\backslash v') \cup (v'\backslash v)\end{matrix}$
$F\backslash w$ (resp. $F\backslash z$) is F where every world of w (resp. z) has been removed from the labels.

Where:
$w = \{a_1, \ldots, a_n\}$
$w' = \{a'_1, \ldots, a'_n\}$
$w'' = \{a''_1, \ldots, a''_n\}$
G is F where every instance of the world a_i has been replaced by both a'_i and a''_i.

Figure 8 Equations with Side-Effects on Worlds Sets

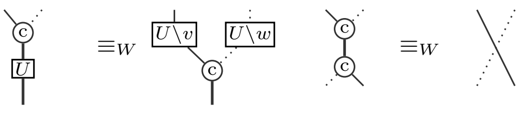
▶ **Theorem 7** (Completeness). *For every $f : \mathfrak{A} \to \mathfrak{B}$ morphism of* $\mathbf{MW}_W$ *and $g : \mathfrak{A} \to \mathfrak{B}$ one of* $\mathbf{MW}_V$*, $[\![f]\!] = [\![g]\!] \iff f \equiv g$* ◀

The soundness can be proved by a case-by-case analysis on every equation. The completeness theorem follows from the existence of a normal form for $\equiv$, as described in the following subsection (Section 5.1).

▶ **Example 8** (The Quantum Switch). As claimed in Example 4, it is possible to represent the Quantum Switch with only one copy of U and V, and one can rewrite it to the version with two copies of each using the equational theory as shown in Figure 9.

In those diagrams, the worlds set is $W = w \sqcup v$ and we rely on thick, thin and doted wires to indicate respectively worlds label $w \sqcup v$, w and v. Each figure has a control side which operates on a quantum bit (type $\mathbb{1} \oplus \mathbb{1}$) and bind the world w to $|0\rangle$ and the world v to $|1\rangle$, and an computational side which operates on a data of an arbitrary type A, on which could be applied U and/or V which stand for two morphisms of $\mathbf{MW}_W(A : W, A : W)$.

The first rewriting steps relies on the two lemmas on the right, both of which being deducible from the equational theory (see ??). The second rewriting step is simply using the properties of a compact closed category.



5 Normal Form and Completeness

We can prove that the previous equational theory is complete, by defining a normal form on the morphisms, and by showing that all diagrams can be put in this normal form, which is unique.

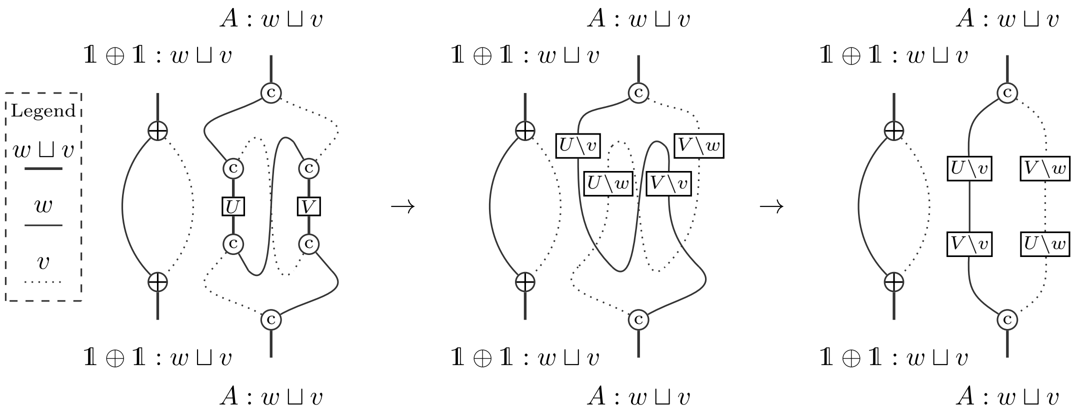
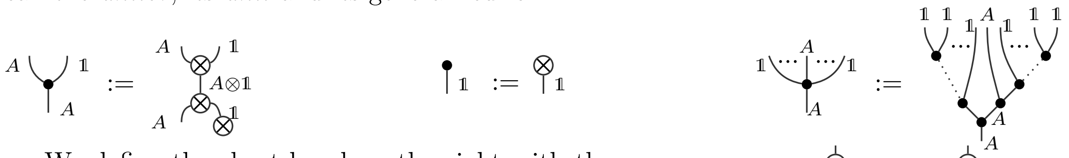


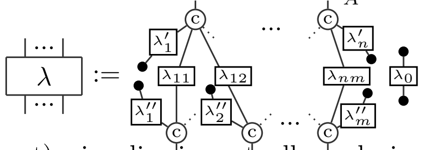
Figure 9 Rewriting the Quantum Switch

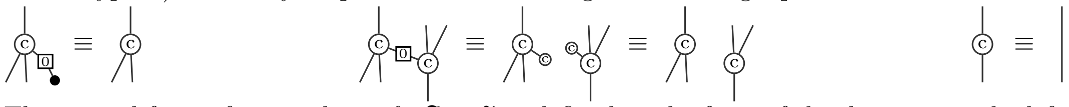
5.1 Normal Form

It will be practical for our normal forms to define the following syntactic sugar, which we call the *unitor*, its *unit* and its generalized form:



We define the short-hand on the right with the assumption that the worlds set is in bijection with the set of scalars, in other words the scalars $\lambda_{ij}, \lambda'_i, \lambda''_j$ and λ_0 have for worlds label $\{a_{ij}\}, \{a'_i\}, \{a''_j\}$ and $\{a_0\}$ respectively. In particular, all the input (resp. output) wires live in mutually exclusive worlds. An important observation is that any permutation of wires (all mutually exclusive, and of type $\mathbb{1}$) can easily be put in this form using the following equations:





The normal form of a morphism $f : \mathfrak{A} \to \mathfrak{B}$ is defined as the form of the diagram on the left of Figure 10, where the morphism $\text{iso}_{\mathfrak{A}}$ is defined inductively right.

The output wires of $\text{iso}_{\mathfrak{A}}$ for any $\mathfrak{A}$ live in mutually exclusive worlds, but once again, we don't overload the diagrams with unitors or world names encoding this information, although it will be used in the following.

Notice that graphically, there is no difference between $\mathfrak{A} \,\square (\mathfrak{B} \,\square\, \mathfrak{C})$ and $(\mathfrak{A} \,\square\, \mathfrak{B}) \,\square\, \mathfrak{C}$, in other words $\square$ is strictly associative. However $\text{iso}_{\mathfrak{A}\,\square(\mathfrak{B}\,\square\,\mathfrak{C})}$ and $\text{iso}_{(\mathfrak{A}\,\square\,\mathfrak{B})\,\square\,\mathfrak{C}}$ are different, but they are equivalent up to a rearranging of the output wires:

▶ **Lemma 9.** *There exists a wire permutation σ such that* $\text{iso}_{\mathfrak{A}\,\square(\mathfrak{B}\,\square\,\mathfrak{C})} = \sigma \circ \text{iso}_{(\mathfrak{A}\,\square\,\mathfrak{B})\,\square\,\mathfrak{C}}$. ◀

We hence have a choice to make here for canonicity, and choose $\text{iso}_{A_0\,\square\,A_1\,\square\,A_2\,\square\,\dots} :=$ $\text{iso}_{(\dots((A_0\,\square\,A_1)\,\square\,A_2)\,\square\,\dots)}$.

We define $\text{iso}^{-1}_{\mathfrak{A}}$ inductively in the same way, but upside-down. We note that $\text{iso}^{-1}_{\mathfrak{A}} \circ \text{iso}_{\mathfrak{A}}$ is the normal form of $\text{id}_{\mathfrak{A}}$.

▶ **Proposition 10.** *The normal form is unique.*

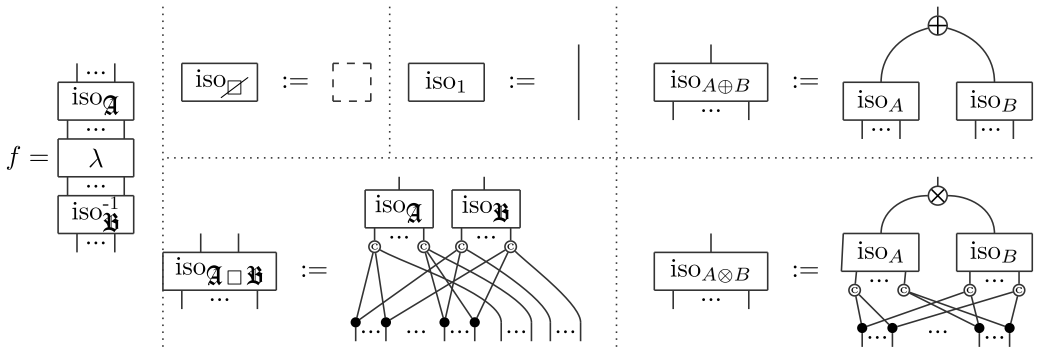


Figure 10 The Normal Form (left) and the Inductive Definition of $\text{iso}_{\mathfrak{A}}$ (right)

5.2 Completeness

We can now use this normal form to show that our equational theory is complete for arbitrary morphisms. To do so, we need to show that all the generators can be put in normal form, and then that any composition of morphisms in normal form can be put in normal form. We do exactly so, and leave the details to the appendix.

▶ **Proposition 11.** *The generators can be put in normal form.* ◀

▶ **Proposition 12.** *Compositions of diagrams in normal form can be put in normal form.*

This allows us to prove the completeness theorem claimed above:

Proof of Theorem 7. The right-to-left direction of the equivalence can be directly checked by verifying that all the axioms preserve the semantics.

Let D_1 and D_2 be two diagrams such that $[\![D_1]\!] = [\![D_2]\!]$. Both diagrams can be put in normal form, resp. D_1^{NF} and D_2^{NF}, with $D_i \equiv D_i^{NF}$ and thus $[\![D_i^{NF}]\!] = [\![D_i]\!]$. By uniqueness of the normal form, and since $[\![D_1^{NF}]\!] = [\![D_2^{NF}]\!]$, we get $D_1^{NF} \equiv D_2^{NF}$, which ends the proof that $D_1 \equiv D_2$. ◀

6 Conclusion

We introduced a new sound and complete graphical language based on compact categories with biproducts, along with a equational theory and a words system, helping us build a denotational semantics of our language.

This language allows us to generalize already existing quantum graphical languages such as the ZX-Calculus with the additions of richer types than just the usual qubits and tensors of qubits, in particular the biproduct allows us to reason about superposition of executions, as shown by the encoding of the Quantum Switch in Example 8.

While our language allows for quantum control, it does not capture another language that aim at formalizing quantum control, namely the PBS-Calculus [8]. How and in which context could we capture the PBS-Calculus is left for future work.

References

1 Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols. In *Proceedings of the 19th Symposium on Logic in Computer Science, LICS'04*, pages 415–425. IEEE, IEEE Computer Society Press, July 2004. doi:10.1109/LICS.2004.1319636.

2 Thorsten Altenkirch and Jonathan Grattage. A functional quantum programming language. In Prakash Panangaden, editor, *Proceedings of the 20th Symposium on Logic in Computer Science, LICS'05*, pages 249–258. IEEE, IEEE Computer Society Press, 2005. doi:10.1109/LICS.2005.1.

3 Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski, and John van de Wetering. There and back again: A circuit extraction tale, 2020. arXiv:2003.01664.

4 Kostia Chardonnet, Benoît Valiron, and Renaud Vilmart. Geometry of interaction for ZX-diagrams. In Filippo Bonchi and Simon J. Puglisi, editors, *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021*, volume 202 of *LIPIcs*, pages 30:1–30:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2021. doi:10.4230/LIPIcs.MFCS.2021.30.

5 Christophe Chareton, Sébastien Bardin, François Bobot, Valentin Perrelle, and Benoît Valiron. A Deductive Verification Framework for Circuit-building Quantum Programs. arXiv:2003.05841. To appear in *Proceedings of ESOP'21*.

6 Christophe Chareton, Sébastien Bardin, Dongho Lee, Benoît Valiron, Renaud Vilmart, and Zhaowei Xu. Formal methods for quantum programs: A survey. arXiv:2109.06493v1, 2021.

7 G. Chiribella, G. M. D'Ariano, and P. Perinotti. Transforming quantum operations: Quantum supermaps. *EPL (Europhysics Letters)*, 83(3):30004, 2008. doi:10.1209/0295-5075/83/30004.

8 Alexandre Clément and Simon Perdrix. Pbs-calculus: A graphical language for coherent control of quantum computations. *arXiv preprint arXiv:2002.09387*, 2020.

9 Robin Cockett and Cole Comfort. The category TOF. In Peter Selinger and Giulio Chiribella, editors, *Proceedings 15th International Conference on Quantum Physics and Logic, QPL 2018*, volume 287 of *EPTCS*, pages 67–84, 2019.

10 Robin Cockett, Cole Comfort, and Priyaa Srinivasan. The category CNOT. In Peter Selinger and Giulio Chiribella, editors, *Proceedings 15th International Conference on Quantum Physics and Logic, QPL 2018*, volume 287 of *EPTCS*, pages 258–293, 2019. doi:10.4204/EPTCS.266.18.

11 Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, 2011.

12 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.

13 Dal Lago, Ugo and Faggian, Claudia and Valiron, Benoît and Yoshimizu, Akira. The geometry of parallelism: Classical, probabilistic, and quantum effects. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, POPL 2017, page 833–845, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3009837.3009859.

14 Niel de Beaudrap and Dominic Horsman. The ZX calculus is a language for surface code lattice surgery. *Quantum*, 4:218, January 2020. doi:10.22331/q-2020-01-09-218.

15 Ross Duncan. Generalized proof-nets for compact categories with biproducts. In Gay and Mackie [20], chapter 3, pages 70–134.

16 Ross Duncan and Liam Garvie. Verifying the smallest interesting colour code with quantomatic. In Bob Coecke and Aleks Kissinger, editors, *Proceedings 14th International Conference on Quantum Physics and Logic, Nijmegen, The Netherlands, 3-7 July 2017*, volume 266 of *Electronic Proceedings in Theoretical Computer Science*, pages 147–163, 2018. doi:10.4204/EPTCS.266.10.

17 Ross Duncan, Aleks Kissinger, Simon Perdrix, and John Van De Wetering. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *Quantum*, 4:279, 2020.

18 Ross Duncan and Maxime Lucas. Verifying the Steane code with Quantomatic. In Bob Coecke and Matty Hoban, editors, *Proceedings of the 10th International Workshop on Quantum Physics and Logic, Castelldefels (Barcelona), Spain, 17th to 19th July 2013*, volume 171 of *Electronic Proceedings in Theoretical Computer Science*, pages 33–49. Open Publishing Association, 2014. doi:10.4204/EPTCS.171.4.

19 Richard P. Feynman and A. R. Hibbs. *Quantum Mechanics and Path Integrals.* McGraw-Hill Publishing Company, 1965.

20 Simon Gay and Ian Mackie, editors. *Semantic Techniques in Quantum Computation.* Cambridge University Press, 2009.

21 Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. Quipper: A scalable quantum programming language. In Hans-Juergen Boehm and Cormac Flanagan, editors, *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI'13*, pages 333–342. ACM, 2013. doi:10.1145/2491956.2462177.

22 Christian Hutslar, Jacques Carette, and Amr Sabry. A library of reversible circuit transformations (work in progress). In Jarkko Kari and Irek Ulidowski, editors, *Proceedings of the 10th International Conference on Reversible Computation, RC 2018*, volume 11106 of *Lecture Notes in Computer Science*, pages 339–345. Springer, 2018. doi:10.1007/978-3-319-99498-7_24.

23 Justin Makary, Neil J. Ross, and Peter Selinger. Generators and relations for real stabilizer operators. In Chris Heunen and Miriam Backens, editors, *Proceedings of hte 18th International Conference on Quantum Physics and Logic, QPL 2021*, volume 343 of *EPTCS*, pages 14–36, 2021. doi:10.4204/EPTCS.343.2.

24 Paul-André Mellies. Local states in string diagrams. In *Rewriting and Typed Lambda Calculi*, pages 334–348. Springer, 2014.

25 Michele Pagani, Peter Selinger, and Benoît Valiron. Applying quantitative semantics to higher-order quantum computing. In Suresh Jagannathan and Peter Sewell, editors, *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL'14*, pages 647–658. ACM, 2014. doi:10.1145/2535838.2535879.

26 Jennifer Paykin, Robert Rand, and Steve Zdancewic. QWIRE: a core language for quantum circuits. In Giuseppe Castagna and Andrew D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL'17*, pages 846–858. ACM, 2017. doi:10.1145/3009837.3009894.

27 Christopher Portmann, Christian Matt, Ueli Maurer, Renato Renner, and Björn Tackmann. Causal boxes: Quantum information-processing systems closed under composition. *IEEE Transactions on Information Theory*, 63(5):3277–3305, 2017. doi:10.1109/TIT.2017.2676805.

28 Amr Sabry, Benoît Valiron, and Juliana Kaizer Vizzotto. From symmetric pattern-matching to quantum control. In Christel Baier and Ugo Dal Lago, editors, *Proceedings of the 21st International Conference on Foundations of Software Science and Computation Structures, FoSSaCS 2018*, volume 10803 of *Lecture Notes in Computer Science*, pages 348–364. Springer, 2018. doi:10.1007/978-3-319-89366-2_19.

29 Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. t|ket⟩: a retargetable compiler for NISQ devices. *Quantum Science and Technology*, 6(1):014003, 2020. doi:10.1088/2058-9565/ab8e92.

30 Sam Staton. Algebraic effects, linearity, and quantum programming languages. In Sriram K. Rajamani and David Walker, editors, *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL'15*, pages 395–406. ACM, 2015. doi:10.1145/2676726.2676999.

31 Augustin Vanrietvelde, Hlér Kristjánsson, and Jonathan Barrett. Routed quantum circuits. *Quantum*, 5:503, 2021. doi:10.22331/q-2021-07-13-503.

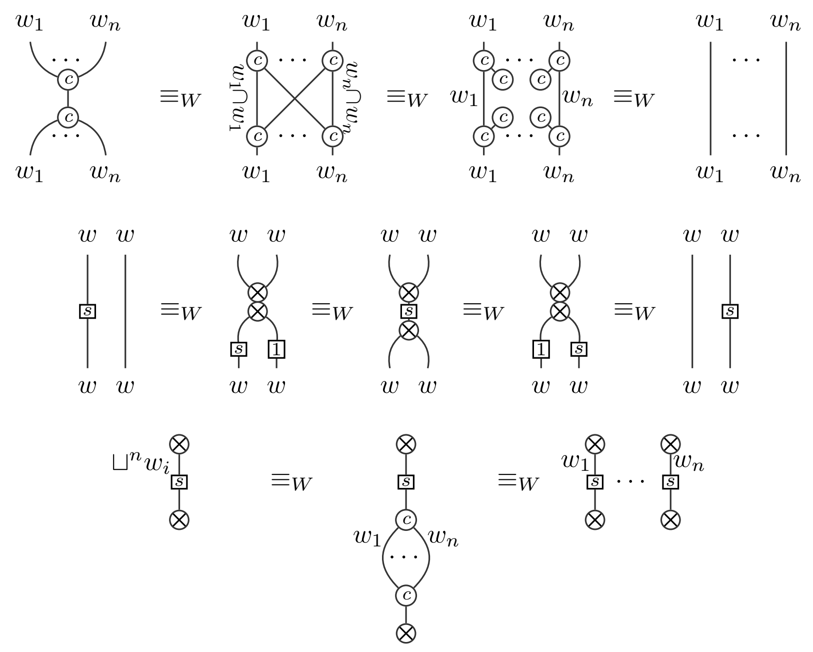
32 Renaud Vilmart. *ZX-Calculi for Quantum Computing and their Completeness.* Theses, Université de Lorraine, September 2019. URL: `https://hal.archives-ouvertes.fr/tel-02395443`.

33 Julian Wechs, Hippolyte Dourdent, Alastair A. Abbott, and Cyril Branciard. Quantum circuits with classical versus quantum control of causal order. *PRX Quantum*, 2:030335, 2021. `doi:10.1103/PRXQuantum.2.030335`.

A Induced Equations

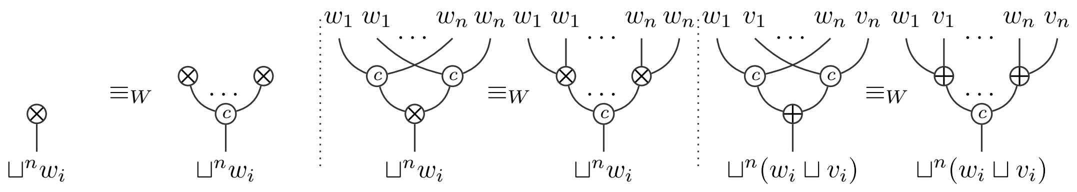
We include in this section a few practical lemmas that are provable from the equational theory $\equiv_W$.

▶ **Lemma 13.** *Whenever w_i are disjoints sets of worlds, we have the following:*

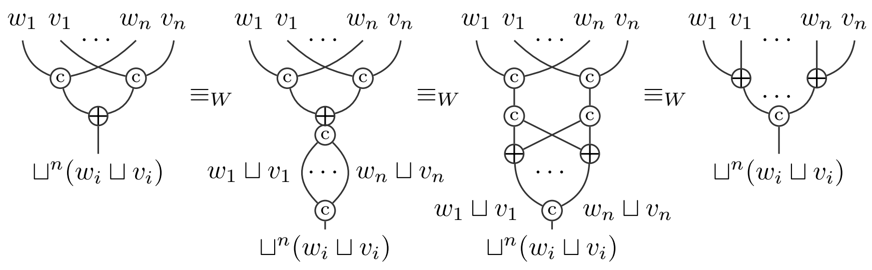


◀

▶ **Lemma 14.** *Whenever w_i are disjoints sets of worlds, and that the v_i are disjoints set of worlds too, we have the following:*

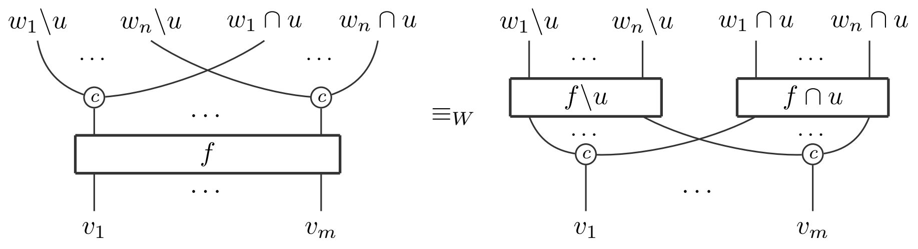


Proof. We provide a proof for the third equation, the first two are proven similarly.



◀

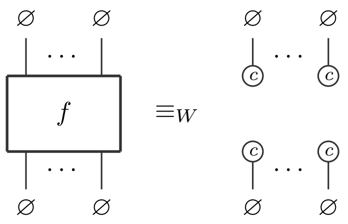
▶ **Corollary 15.** *For every $f : \Box^n(A_i : w_i) \to \Box^m(B_j : v_j)$ with worlds set W and every $u \subseteq W$, we have*



where $f\backslash u : \Box^n(A_i : w_i\backslash u) \to \Box^m(B_j : v_j\backslash u)$ is equal to f where every worlds label w has been replaced by $w\backslash u$, and similarly for $f \cap w$.

This is simply proven by induction over f. All the generator cases (including Cup and Cap) follow directly from the equations given in Figure 7 an **??** together with the properties of a compact close category.

▶ **Lemma 16.** *For every $f : \Box^n(A_i : \varnothing) \to \Box^m(B_j : \varnothing)$ with worlds set W but such that every worlds label of f is $\varnothing$, we have*



This is simply proven by replacing every wire by two contractions of arity zero (sixth axiom of Figure 7 with $n = 0$), and then using the naturality of the contraction of arity zero (last two lines of Figure 7 with $n = 0$) to consume every generator.

B Dealing with Worlds Labeling

The worlds labeling can appear to be a very strict structure making it unpractical to manipulate diagrams. In this appendix, we show a number of tools that allow to manipulate worlds labeling in a more flexible way.

B.1 Worlds-Agnostic Category

In this section, we define the category $\mathbf{MW}_\forall$ which has the same objects as $\mathbf{C_D}$, so no worlds labeling on the objects, and has for mosphisms all the morphisms of the categories $\mathbf{MW}_W$ for every worlds set W. In other words, we explain how to make sequential composition and parallel composition of morphisms that have different worlds set.

We write $[f]_W : \mathfrak{A} \to \mathfrak{B}$ for a morphism $(\mathcal{D}_f, \ell_f) \in \mathbf{MW}_W((\mathfrak{A}, \ell_f^{\mathfrak{A}}), (\mathfrak{B}, \ell_f^{\mathfrak{B}}))$, where $\ell_f^{\mathfrak{A}}$ and $\ell_f^{\mathfrak{B}}$ are the labeling induced on $\mathfrak{A}$ and $\mathfrak{B}$ by ℓ_f.

The goal of this section is to allow to compose $[f]_W : A \to B$ and $[g]_V : B \to C$ even when the inferred labels on B do not match and when W and V are different.

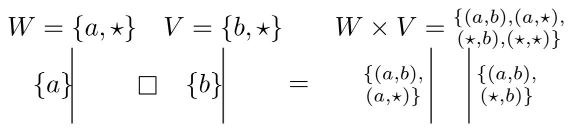


Figure 11 Example of Worlds-Agnostic Parallel Composition

B.1.1 Extending the Worlds Set

Before tackling the sequential composition, we start by defining the parallel composition $\square$ between morphisms with different world sets. For W, V two sets of worlds, and for $\ell : S \to W$, we define $\ell^{-\times V} : S \to W \times V$ as $\ell^{-\times V}(s) = \{(a,b) \mid a \in \ell(s), b \in V\}$. Similarly, for $[f]_W : \mathfrak{A} \to \mathfrak{B}$, we define $[f^{-\times V}]_{W\times W}$ as the morphism $(\mathcal{D}_f, \ell_f^{-\times V})$. We define $\ell^{V\times -}$ and $f^{V\times -}$ symmetrically.

▶ **Definition 17** (Worlds-Agnostic Parallel Composition). *For $[f]_W : \mathfrak{A} \to \mathfrak{B}$ and $[g]_V : \mathfrak{C} \to \mathfrak{D}$ two morphisms, we define*

$$[f]_W \square [g]_V := [f^{-\times V} \square g^{W\times -}]_{W\times V} : (\mathfrak{A} \square \mathfrak{B}) \to (\mathfrak{C} \square \mathfrak{D})$$

In Figure 11, we show the result of this parallel composition when $[f]_{\{a,\star\}} = \mathbf{id}_{A:\{a\}}$ and $[g]_{\{b,\star\}} = \mathbf{id}_{A:\{b\}}$ for a color A.

B.1.2 Restricting the Worlds Set

We now tackle the sequential composition. For W a set of worlds, $w \subseteq W$, we define the restriction of $\ell : S \to W$ to $W \backslash w$ as $\ell^{\backslash w} : S \to W \backslash w$ as $\ell^{\backslash w}(s) = \{a \in \ell(s) \mid a \notin w\}$. For $[f]_W : \mathfrak{A} \to \mathfrak{B}$ we define $[f^{\backslash w}]_{W\backslash w} : \mathfrak{A} \to \mathfrak{B}$ as the morphism $(\mathcal{D}_f, \ell_f^{\backslash w})$.

▶ **Definition 18.** *Given two morphisms $[f]_W : (\mathfrak{A}, \ell_{\mathfrak{A}}) \to (\mathfrak{B}, \ell_{\mathfrak{B}})$ and $[g]_V : (\mathfrak{B}, k_{\mathfrak{B}}) \to (\mathfrak{C}, k_{\mathfrak{C}})$, we write w for the smallest subset of $W \times V$ such that $(f^{-\times V})^{\backslash w}$ induces the same worlds labeling on $\mathfrak{B}$ as $(g^{W\times -})^{\backslash w}$. We then define*

$$[g]_V \circ [f]_W := [(g^{W\times -})^{\backslash w} \circ (f^{-\times V})^{\backslash w}]_{(W\times V)\backslash w}$$

We continue the previous example in Figure 12 by composing the result with the Cup over $A : \{c, \star\}$. We proceed in two steps: first we handle the situation as if it was a parallel composition, leading to a diagram labeled over $W \times V \times U$, but with multiple contradictory labels on the wire. Then, we eliminate as few worlds as possible to make those labels compatible:

- We eliminate $(a, b, \star)$ and $(a, \star, \star)$ which are on the *left* label but not on the *bottom* one.
- We eliminate $(\star, b, c)$ and $(\star, \star, c)$ which are on the *bottom* label but not on the *left* one.
- We eliminate $(\star, b, \star)$ which is on the *right* label but not on the *bottom* one. We would also eliminate $(a, b, \star)$ if we had not done so already.
- We eliminate $(a, \star, c)$ which is on the *bottom* label but not on the *right* one. We would also eliminate $(\star, \star, c)$ if we had not done so already.

The eliminated worlds are $w = \{(a, b, \star), (a, \star, c), (\star, b, c), (a, \star, \star), (\star, b, \star), (\star, \star, c)\}$, and what remains is $\{(a, b, c), (\star, \star, \star)\}$.

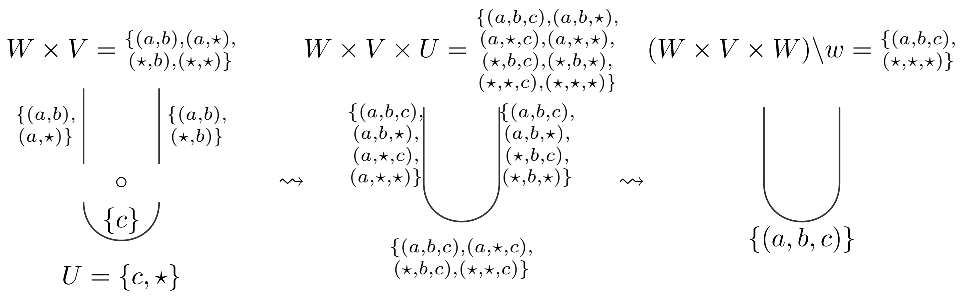

$W \times V = \{(a,b),(a,\star),(\star,b),(\star,\star)\}$
$\{(a,b),(a,\star)\}$
$\{(a,b),(\star,b)\}$
$\circ$
$\{c\}$
$U = \{c,\star\}$
$\rightsquigarrow$
$W \times V \times U = \{(a,b,c),(a,b,\star),(a,\star,c),(a,\star,\star),(\star,b,c),(\star,b,\star),(\star,\star,c),(\star,\star,\star)\}$
$\{(a,b,c),(a,b,\star),(a,\star,c),(a,\star,\star)\}$
$\{(a,b,c),(a,b,\star),(\star,b,c),(\star,b,\star)\}$
$\{(a,b,c),(a,\star,c),(\star,b,c),(\star,\star,c)\}$
$\rightsquigarrow$
$(W \times V \times W)\backslash w = \{(a,b,c),(\star,\star,\star)\}$
$\{(a,b,c)\}$

Figure 12 Example of Worlds-Agnostic Sequential Composition

B.1.3 The Worlds-Agnostic Category and its Semantics

$\mathbf{MW}_\forall$ with the sequential and parallel composition as described above forms a category (up to renaming of worlds), and is in fact an auto-dual compact closed colored PROP (up to renaming of worlds).

▶ **Proposition 19.** *The worlds-agnostic semantics* $[\![-]\!]$ *defined in Section 3 is a monoidal functor from* $\mathbf{MW}_\forall$ *to* ***FdHilb****.*

Proof. We recall here the definition of the worlds-agnostic semantics of $[f]_W : \mathfrak{A} \to \mathfrak{B}$:

$$[\![[f]_W]\!] := \left\{ \sum_{\substack{a\in W \\ (\mathfrak{A},\ell^f_{\mathfrak{A}})\Downarrow a=\mathfrak{A}' \\ (\mathfrak{B},\ell^f_{\mathfrak{B}})\Downarrow a=\mathfrak{B}'}} [\![f]\!]_a \right\}_{\mathfrak{A}'\in\mathfrak{A}^\bullet,\mathfrak{B}'\in\mathfrak{B}^\bullet}$$

From the definition of the worlds-agnostic compositions, we directly have:

$$[\![[f]_W \,\square\, [g]_V]\!]_{(a,b)} = [\![[f]_W]\!]_a \otimes [\![[g]_V]\!]_b \qquad\qquad [\![[g]_V \circ [f]_W]\!]_{(a,b)} = [\![[g]_V]\!]_b \circ [\![[f]_W]\!]_a$$

The functoriality with respect to the parallel composition is then immediate:

$$[\![[f]_W \,\square\, [g]_V]\!] = \left\{\sum [\![[f]_W \,\square\, [g]_V]\!]_{(a,b)}\right\} = \left\{\sum [\![[f]_W]\!]_a\right\}\otimes\left\{\sum [\![[g]_V]\!]_b\right\} = [\![[f]_W]\!]\otimes[\![[g]_V]\!]$$

The functoriality with respect to the sequential composition is more subtle, as one must carefully manipulate the indices of the sum and remark that the set of worlds w eliminated by the worlds-agnostic composition satisfies the following:

$$(a,b) \notin w \iff (\mathfrak{B},\ell^f_{\mathfrak{B}}) \Downarrow a = (\mathfrak{B},\ell^g_{\mathfrak{B}}) \Downarrow b \text{ where } \begin{array}{l} f : (\mathfrak{A},\ell^f_{\mathfrak{A}}) \to (\mathfrak{B},\ell^f_{\mathfrak{B}}) \\ g : (\mathfrak{B},\ell^g_{\mathfrak{C}}) \to (\mathfrak{C},\ell^g_{\mathfrak{C}}) \end{array}$$

Then, we have

$$[\![[g]_V \circ [f]_W]\!] = \left\{\sum [\![[g]_V \circ [f]_W]\!]_{(a,b)}\right\} = \left\{\sum [\![[g]_V]\!]_b\right\}\circ\left\{\sum [\![[f]_W]\!]_a\right\} = [\![[g]_V]\!]\circ[\![[f]_W]\!]$$

◀

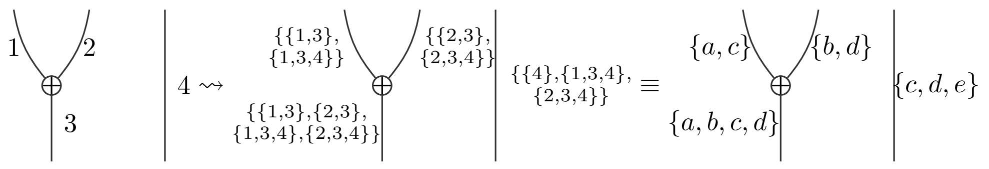


Figure 13 A diagram, its canonical labeling, and its equivalent with worlds set $\{a, b, c, d, e, \star\}$

B.2 Keeping the Worlds Labeling Implicit

In Figure 5, we provide an example where one can use the equational theory without the worlds. In the previous section, we provided a worlds-agnostic semantics to our diagrams. It is reasonable to wonder how much can be done within $\mathbf{C_D}$, in other words without using the worlds labeling. In the following, we show how to provide a canonical labeling for diagrams of $\mathbf{C_D}$ and deduce a denotational semantics for $\mathbf{C_D}$.

We consider a diagram $\mathcal{D}$ of $\mathbf{C_D}$ and we want to define a canonical worlds labeling over its set of wires $\mathbb{W}$. The core idea is that a world will be a set of wires that can cohabit with each others while satisfying the constraints of the nodes Plus, Tensor, etc. For example in Figure 13, there are four wires with wire 1 and 2 being incompatible due to the Plus, but also inseparable from wire 3, hence the canonical worlds set would be:

$$\{\varnothing, \{1,3\}, \{2,3\}, \{4\}, \{1,3,4\}, \{2,3,4\}\}$$

More formally, a subset of wires $S \subseteq \mathbb{W}$ is said *valid* if the labeling $\ell_S : x \mapsto \{\star\}$ if $x \in S$ and $\varnothing$ otherwise is a valid labeling for $\mathcal{D}$, in other words if $(\mathcal{D}, \ell_S)$ is a morphism of $\mathbf{MW}_{\{\star\}}$. We will take for worlds set $W_{\mathcal{D}} \subseteq \mathcal{P}(\mathbb{W})$ the sets of valid subsets of wires. The labeling will simply be: $\ell_{\mathcal{D}} : x \mapsto \{S \in \mathbb{V} \mid x \in S\}$.

▶ **Proposition 20.** *For every diagram $\mathcal{D}$ of $\mathbf{C_D}$, $(\mathcal{D}, \ell_{\mathcal{D}})$ is a morphism of $\mathbf{MW}_{W_{\mathcal{D}}}$. Up to renaming of the worlds, this construction is a monoidal functor from $\mathbf{C_D}$ to $\mathbf{MW}_{\forall}$.*

It follows that if we define $[\![\mathcal{D}]\!]$ as $[\![(\mathcal{D}, \ell_{\mathcal{D}})]\!]$, this semantics is a monoidal functor from $\mathbf{C_D}$ to $\mathbf{FdHilb}$. We note that it is enough to compute the semantics of the generators of $\mathbf{C_D}$ (see FIGURE) to obtain the semantics for every diagram using $[\![\mathcal{D} \circ \mathcal{D}']\!] = [\![\mathcal{D}]\!] \circ [\![\mathcal{D}']\!]$ and $[\![\mathcal{D} \,\square\, \mathcal{D}']\!] = [\![\mathcal{D}]\!] \otimes [\![\mathcal{D}']\!]$.

C Proofs of Universality and Uniqueness of Normal Form

Universality

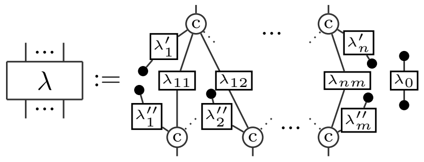
In this section, we show that for every $\mathfrak{A}, \mathfrak{B}$ objects of $\mathbf{C_D}$, and for every linear operator

$$\Lambda = \begin{pmatrix} \lambda_{1\,1} & \cdots & \lambda_{n\,1} & \lambda''_1 \\ \vdots & & \vdots & \vdots \\ \lambda_{1\,m} & \cdots & \lambda_{n\,m} & \lambda''_m \\ \lambda'_1 & \cdots & \lambda'_n & \lambda_0 \end{pmatrix} \in \mathbf{FdHilb}\left(\bigoplus_{\mathfrak{C}\in\mathfrak{A}^\bullet} \mathcal{H}_{\mathfrak{C}}, \bigoplus_{\mathfrak{F}\in\mathfrak{B}^\bullet} \mathcal{H}_{\mathfrak{B}}\right)$$

then the morphism $f = \mathrm{iso}_{\mathfrak{B}}^{-1} \circ \lambda \circ \mathrm{iso}_{\mathfrak{A}}$ defined in Figure 10 satisfies $[\![f]\!] = \Lambda$.

As stated in the corresponding section, there is one world for each scalar in λ, so we write $a_{i\,j}$ the world associated to $\lambda_{i\,j}$ and a'_i, a''_j, a_0 similarly, and W the set of all those worlds. Writing $\bullet^k$ for $\bullet\square\ldots\square\bullet$ (with k elements) and $\mathbb{1}_i^k$ for $\bullet^k$ where the i-th $\bullet$ has been replaced by $\mathbb{1}$, we have the following:

$$\begin{cases} [\![\lambda]\!]_{a_{i\,j}} : \mathbb{1}_i^n \to \mathbb{1}_j^m : & x \mapsto \lambda_{i\,j}\cdot x \\ [\![\lambda]\!]_{a'_i} : \mathbb{1}_i^n \to \bullet^m : & x \mapsto \lambda'_i \cdot x \\ [\![\lambda]\!]_{a''_j} : \bullet^n \to \mathbb{1}_j^m : & x \mapsto \lambda''_j \cdot x \\ [\![\lambda]\!]_{a_0} : \bullet^n \to \bullet^m : & x \mapsto \lambda_0 \cdot x \end{cases}$$

where 

Additionally, one can show by induction that $[\![\mathrm{iso}_{\mathfrak{A}}]\!]_{a_{i\,j}}$ and $[\![\mathrm{iso}_{\mathfrak{A}}]\!]_{a'_i}$ (resp. $[\![\mathrm{iso}_{\mathfrak{A}}]\!]_{a''_j}$ and $[\![\mathrm{iso}_{\mathfrak{A}}]\!]_{a_0}$) are simply the projection on the i-th (resp. $(n+1)$-th) element of the canonical basis, and $\left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_{a_{i\,j}}$ and $\left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_{a''_j}$ (resp. $\left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_{a'_i}$ and $\left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_{a_0}$) are simply the injection on the j-th (resp. $(m+1)$-th) element of the canonical basis. Since we have $[\![f]\!]_a = \left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_a \circ [\![\lambda]\!]_a \circ [\![\mathrm{iso}_{\mathfrak{A}}]\!]_a$ for every $a \in W$, and $[\![f]\!]$ being the collection of all the $[\![f]\!]$, we obtain that $[\![f]\!] = \Lambda$.

Uniqueness of Normal Form

Let f and g be two diagrams in normal form (with respectively λ and μ as inner block), such that $[\![f]\!] = [\![g]\!]$ (the naming of the worlds is taken to be the same in both diagrams, and is the same as in the previous proof). By the definition of $[\![.]\!]$, we have $[\![f]\!]_a = [\![g]\!]_a$ for every $a \in W$. We hence have $\left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_a \circ [\![\lambda]\!]_a \circ [\![\mathrm{iso}_{\mathfrak{A}}]\!]_a = \left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_a \circ [\![\mu]\!]_a \circ [\![\mathrm{iso}_{\mathfrak{A}}]\!]_a$.

Denoting $e_i^{\mathfrak{A}}$ (resp. $e_i^{\mathfrak{B}}$) the i-th element of the basis of $\mathfrak{A}$ (resp. $\mathfrak{B}$), we have:

$$\begin{cases} \lambda_{ij} = e_j^{\mathfrak{B}\dagger} \left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_{a_{ij}} \circ [\![\lambda]\!]_{a_{ij}} \circ [\![\mathrm{iso}_{\mathfrak{A}}]\!]_{a_{ij}} e_i^{\mathfrak{A}} = e_j^{\mathfrak{B}\dagger} \left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_{a_{ij}} \circ [\![\mu]\!]_{a_{ij}} \circ [\![\mathrm{iso}_{\mathfrak{A}}]\!]_{a_{ij}} e_i^{\mathfrak{A}} = \mu_{ij} \\ \lambda'_i = \bullet^{|\mathfrak{B}|\dagger} \left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_{a'_i} \circ [\![\lambda]\!]_{a'_i} \circ [\![\mathrm{iso}_{\mathfrak{A}}]\!]_{a'_i} e_i^{\mathfrak{A}} = \bullet^{|\mathfrak{B}|\dagger} \left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_{a'_i} \circ [\![\mu]\!]_{a'_i} \circ [\![\mathrm{iso}_{\mathfrak{A}}]\!]_{a'_i} e_i^{\mathfrak{A}} = \mu'_i \\ \lambda''_j = e_j^{\mathfrak{B}\dagger} \left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_{a''_j} \circ [\![\lambda]\!]_{a''_j} \circ [\![\mathrm{iso}_{\mathfrak{A}}]\!]_{a''_j} \bullet^{|\mathfrak{A}|} = e_j^{\mathfrak{B}\dagger} \left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_{a''_j} \circ [\![\mu]\!]_{a''_j} \circ [\![\mathrm{iso}_{\mathfrak{A}}]\!]_{a''_j} \bullet^{|\mathfrak{A}|} = \mu''_j \\ \lambda_0 = \bullet^{|\mathfrak{B}|\dagger} \left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_{a_0} \circ [\![\lambda]\!]_{a_0} \circ [\![\mathrm{iso}_{\mathfrak{A}}]\!]_{a_0} \bullet^{|\mathfrak{A}|} = \bullet^{|\mathfrak{B}|\dagger} \left[\!\!\left[\mathrm{iso}_{\mathfrak{B}}^{-1}\right]\!\!\right]_{a_0} \circ [\![\mu]\!]_{a_0} \circ [\![\mathrm{iso}_{\mathfrak{A}}]\!]_{a_0} \bullet^{|\mathfrak{A}|} = \mu_0 \end{cases}$$

where $|\mathfrak{B}|$ denotes the number of wires in $\mathfrak{B}$.
Hence, all coefficients in the scalars of f and g are the same. Since the structure is otherwise the same for f and g, they are the same diagram.

D Proof of Soundness

Given that most of the time, $[\![\mathrm{gen}]\!]_a$ is the identity, the equations defining $\equiv_W$ are quite straightforward to verify. We immediately have that $\equiv_W$ is sound with respect to $[\![-]\!]_a$ for every $a \in W$. Since $[\![-]\!]$ is defined from $[\![-]\!]_a$, soundness with respect to $[\![-]\!]$ is also correct. We then handle the five additional equations of $\equiv$.

Renaming Applying a bijection to the worlds set W does not change the result computed by $\sum_{a\in W}\dots$, hence this equation is sound with respect to $[\![-]\!]$.

Annihilation due to Scalars This equation simply removes elements equal to zero from the sum $\sum_{a\in W}\dots$, hence it is sound with respect to $[\![-]\!]$.

Annihilation due to Plus Since $\oplus$ is a biproduct in FdHilb, we have $\mathrm{proj}_H^{H\oplus K}\circ\mathrm{inj}_H^{H\oplus K} = \mathbf{id}_H$, $\mathrm{proj}_K^{H\oplus K}\circ\mathrm{inj}_K^{H\oplus K} = \mathbf{id}_K$, $\mathrm{proj}_K^{H\oplus K}\circ\mathrm{inj}_H^{H\oplus K} = 0$ and $\mathrm{proj}_H^{H\oplus K}\circ\mathrm{inj}_K^{H\oplus K} = 0$. One can then simply remove from the $\sum_{a\in W}\dots$ the elements equal to zero, which proves that *Annihilation due to Plus* is sound with respect to $[\![-]\!]$.

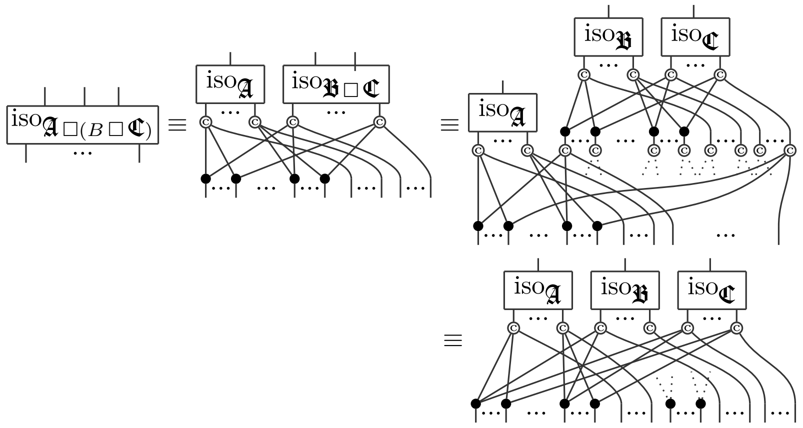
Splitting due to Scalars Since **FdHilb** is a vector space, we have $(s+t)\cdot f = s\cdot f + t\cdot f$, which is exactly the property required for this equation to be sound for $[\![-]\!]$.

Splitting due to Plus Similarly, we have in **FdHilb** the property that $\mathbf{id}_{H\oplus K} = \mathrm{inj}_H^{H\oplus K}\circ\mathrm{proj}_H^{H\oplus K} + \mathrm{inj}_K^{H\oplus K}\circ\mathrm{proj}_K^{H\oplus K}$, which is the property required for this equation to be sound for $[\![-]\!]$.

E Proofs for Completeness

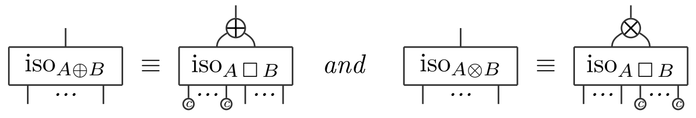
E.1 The Normal Form

Proof of Lemma 9. First notice that in both $\mathrm{iso}_{\mathfrak{A}\,\square(\mathfrak{B}\,\square\,\mathfrak{C})}$ and $\mathrm{iso}_{(\mathfrak{A}\,\square\,\mathfrak{B})\,\square\,\mathfrak{C}}$, we can use the bialgebra between contractions and unitors, followed by their respective fusions in the following way:

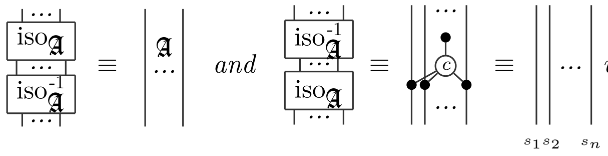


and similarly for $\mathrm{iso}_{(A\,\square\,B)\,\square\,C}$. It then suffices to check which contractions the bottom unitors are linked to. Naming the ith contraction existing iso_A as a_i, and similarly for B and C, we can see that for each triple (a_i, b_j, c_k) there is exactly one unitor connected to precisely contraction a_i, b_j and c_k, in both diagrams. The same is true for every pair (a_i, b_j), (a_i, c_k) and (b_j, c_k), as well as for every 1-tuple $(a_i,)$, $(b_j,)$ and $(c_k,)$. This shows that both diagrams are equal up to rearranging of the outputs. ◀

▶ **Lemma 21.** *Notice that*

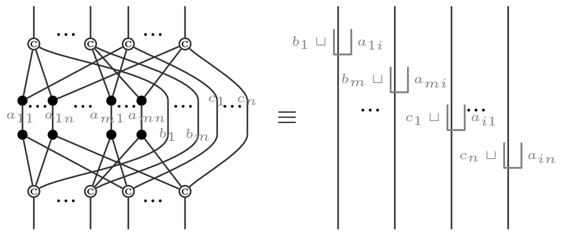


▶ **Lemma 22.** *We have the following identities:*



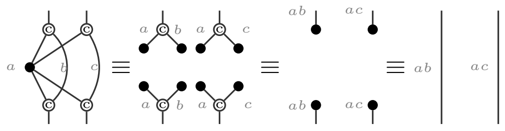
where $\ell(s_i) \cap \ell(s_j) = \varnothing$ when $i \neq j$.

Proof of Lemma 18. We will use the following identities:

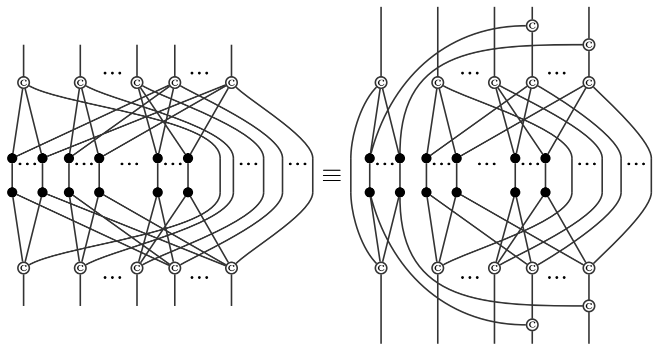


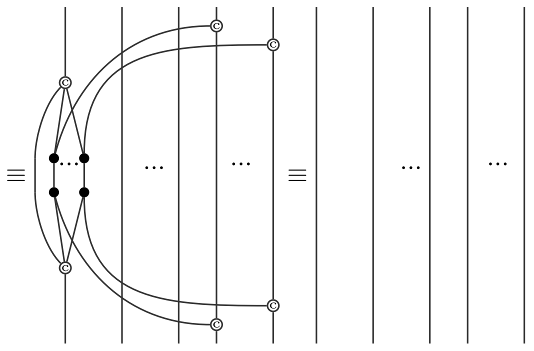
when all the wires a_{ij}, b_i, c_j are mutually exclusive (1)

We can show this result by induction on n and m. Case $(0, m)$ is obvious. Case $(1, 1)$ can be proven easily using worlds sets:

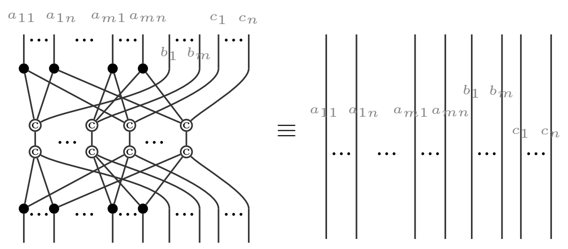


For any n and m, we can then prove the case $(n+1, m)$ using the cases (n, m) and $(1, m)$ (the case $(n, m+1)$ is completely symmetric):

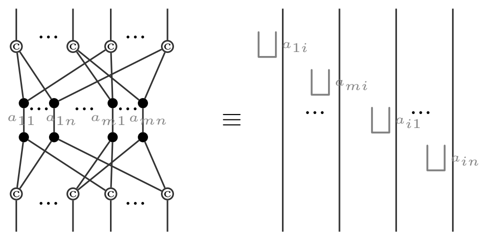




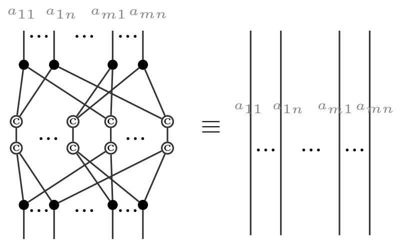
In a similar way, it is possible to show the following three identities:



when all the wires a_{ij}, b_i, c_j are mutually exclusive

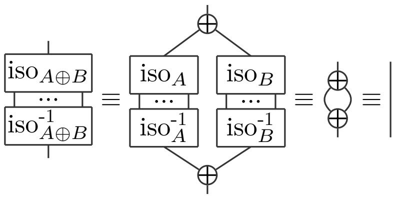


when all the wires a_{ij} are mutually exclusive



when all the wires a_{ij} are mutually exclusive

- $[\mathrm{iso}^{-1}_{\mathfrak{A}} \circ \mathrm{iso}_{\mathfrak{A}}]$: The result is obvious in cases $\mathbb{1}$ and $\not\varnothing$. For $A \oplus B$:



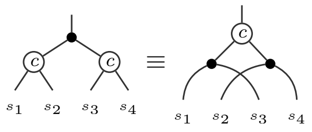
The proof is similar for $\otimes$ and $\square$ using the previous identities.

- $[\mathrm{iso}_{\mathfrak{A}} \circ \mathrm{iso}^{-1}_{\mathfrak{A}}]$: The result is again obvious for $\mathbb{1}$ and $\not\varnothing$. The general result is easy to prove by induction using the above identities.

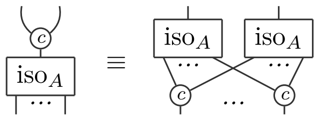
◀

E.2 The Completeness

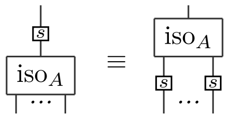
We want to show here the results of Section 5.2. To do so we will first derive a few lemmas:

▶ **Corollary 23** (of ??).  *when* $s_1 \cap s_4 = s_2 \cap s_3 = \varnothing$

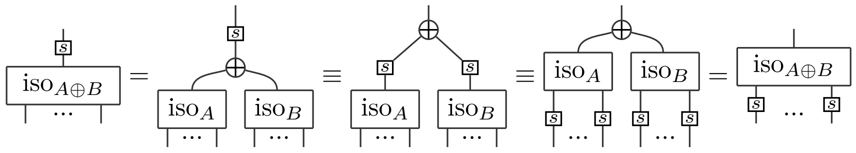
▶ **Corollary 24** (of ??). *Single-colored isos distribute over the contraction:*



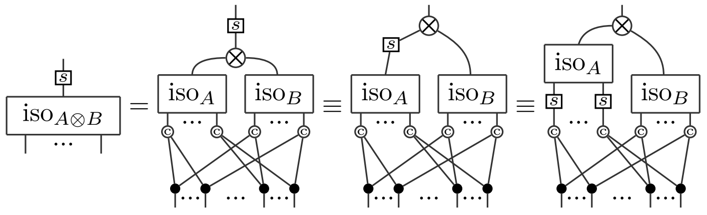
▶ **Lemma 25.** *Scalars distribute over single-colored isos:*

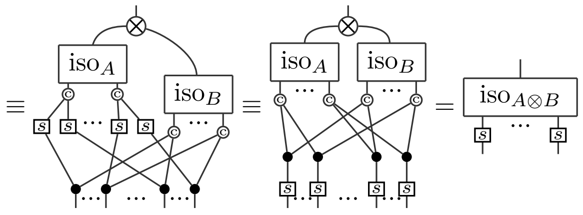


Proof. The result is obvious for $\mathbb{1}$ and $\not\varnothing$. For $A \oplus B$:



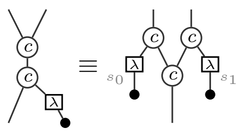
For $A \otimes B$:





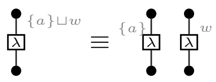
◀

▶ **Corollary 26** (of ??).

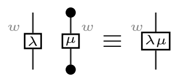


with $s_0 \cap s_1 = \varnothing$.

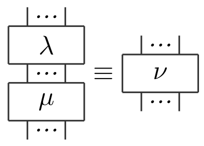
▶ **Corollary 27** (of ??).



▶ **Corollary 28** (of ??).

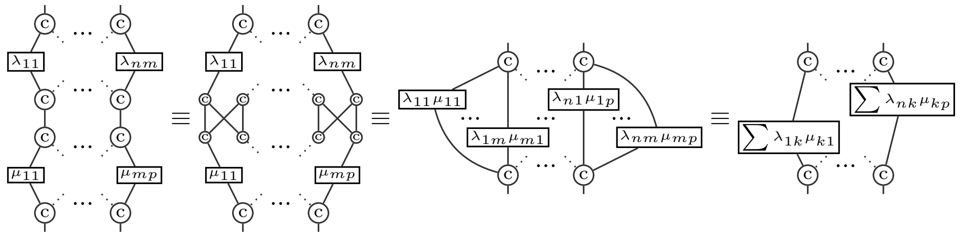


▶ **Lemma 29.**



with $\nu_{ij} = \sum_k \lambda_{ik}\mu_{k_j} + \lambda'_i\mu''_j$, $\nu'_i = \sum_k \lambda_{ik}\mu''_k + \mu_0\lambda'_i$, $\nu''_i = \sum_k \lambda''_k\mu_{ki} + \lambda_0\mu''_i$ *and* $\nu_0 = \lambda_0\mu_0 + \sum_k \lambda''_k\mu'_k$.

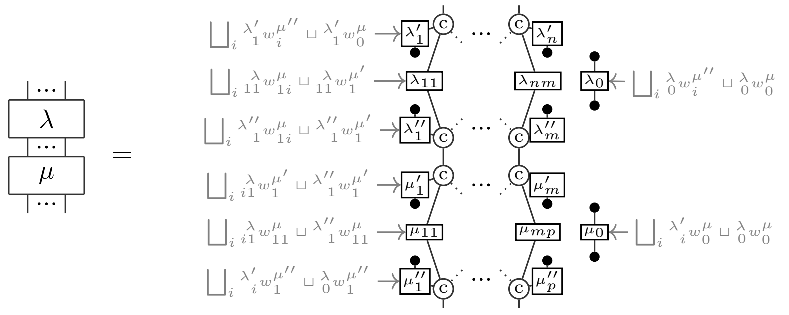
Proof. Consider first the following simpler case:



(2)

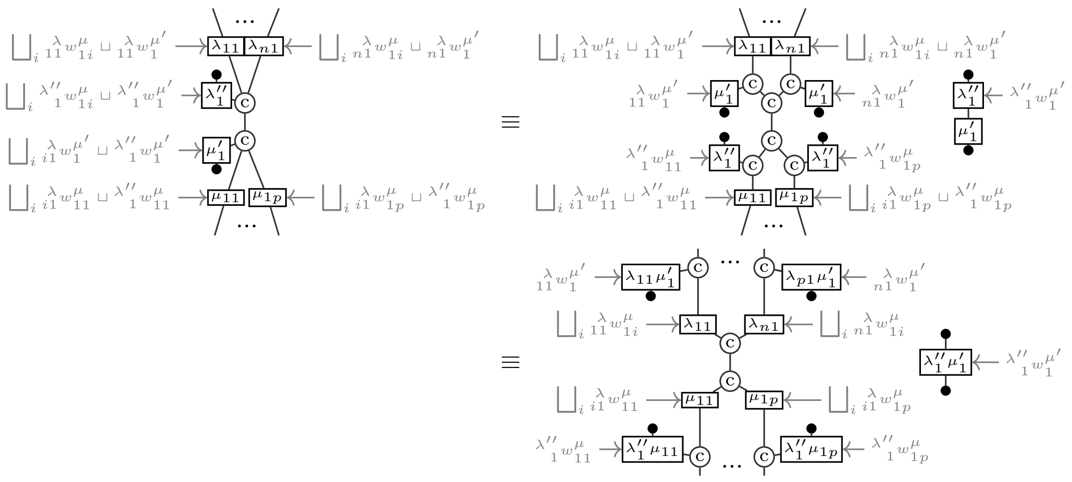
Importantly, in the last diagram, we have that each internal wire lives in a set of worlds that has empty intersection with any of the others. It is hence possible to rename this set of worlds to a singleton, and to do it for each internal wire.

For the general case, it is important to properly look at the worlds. Denoting w^{λ}_{ij} the (singleton) world that bears λ_{ij}, $w^{\lambda'}_i$ the one that bears λ'_i, $w^{\lambda''}_i$ the one that bears λ''_i and w^{λ}_0 the one that bears λ_0 in matrix block λ, and similarly in matrix block μ, we get:

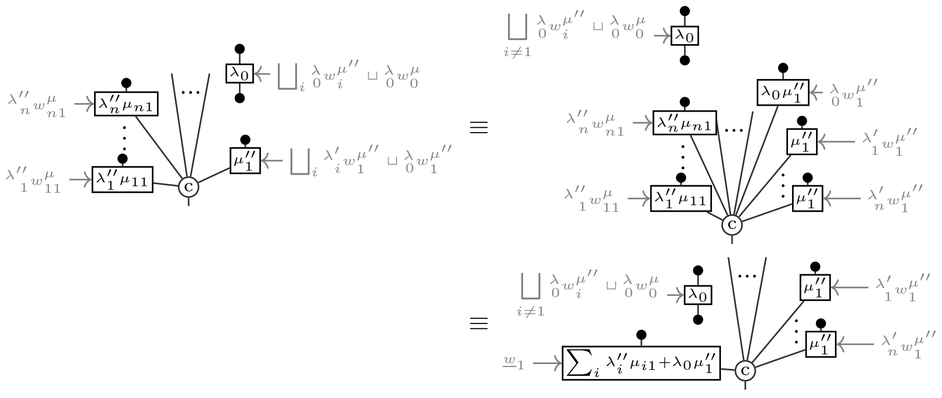


λ
μ
$=$
$\bigsqcup_i {}^{\lambda'}_{1} w^{\mu''}_{i} \sqcup {}^{\lambda'}_{1} w^{\mu}_{0}$
$\bigsqcup_i {}^{\lambda}_{11} w^{\mu}_{1i} \sqcup {}^{\lambda}_{11} w^{\mu'}_{1}$
$\bigsqcup_i {}^{\lambda''}_{1} w^{\mu}_{1i} \sqcup {}^{\lambda''}_{1} w^{\mu'}_{1}$
$\bigsqcup_i {}^{\lambda}_{i1} w^{\mu'}_{1} \sqcup {}^{\lambda''}_{1} w^{\mu'}_{1}$
$\bigsqcup_i {}^{\lambda}_{i1} w^{\mu}_{11} \sqcup {}^{\lambda''}_{1} w^{\mu}_{11}$
$\bigsqcup_i {}^{\lambda'}_{i} w^{\mu''}_{1} \sqcup {}^{\lambda}_{0} w^{\mu''}_{1}$
λ'_1
λ'_n
λ_{11}
λ_{nm}
λ''_1
λ''_m
μ'_1
μ'_m
μ_{11}
μ_{mp}
μ''_1
μ''_p
λ_0
μ_0
$\bigsqcup_i {}^{\lambda}_{0} w^{\mu''}_{i} \sqcup {}^{\lambda}_{0} w^{\mu}_{0}$
$\bigsqcup_i {}^{\lambda'}_{i} w^{\mu}_{0} \sqcup {}^{\lambda}_{0} w^{\mu}_{0}$

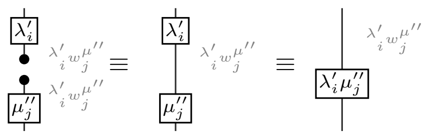
where ${}^{a}_{x}w^{b}_{y} = \{(z_1, z_2) \mid z_1 \in w^{a}_{x}, z_2 \in w^{b}_{y}\}$. Let us now look at how to push the middle dangling wires through. At the first junction, we have:



Doing this at every junction wire, we are able to push all the middle dangling wires to the boundaries, and where the middle is in the shape of the previous simple case (2).
Let us now look at the boundaries, for instance the first output. After pushing all dangling wires, and taking into account the scalar diagram λ_0, we have:

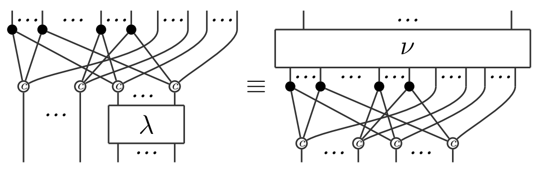


where on the right hand side of the diagram, we decomposed the dangling wire by its different elementary worlds. One of them was shared with the λ_0 scalar, and using Lemmas 22 and 23, we could bring them together. Then on the left hand side, we used the fact that none of the ${}^{\lambda''}_{i}w^{\mu}_{i1}$ nor the ${}^{\lambda}_{0}w^{\mu''}_{1}$ could be found elsewhere in the diagram, so we could sum all the scalars without altering the rest of the diagram.
Doing so for all boundaries, scalars λ_0 and μ_0 are only left on world ${}^{\lambda}_{0}w^{\mu}_{0}$. We can then use Lemma 23 to bring them together. All the worlds on right hand side of the last diagram above ${}^{\lambda'}_{i}w^{\mu''}_{1}$ can be found on the boundaries at the top (i.e. for each input i and output j there is world ${}^{\lambda'}_{i}w^{\mu''}_{j}$ shared between the two). Using the small following derivation, we can bring them together:

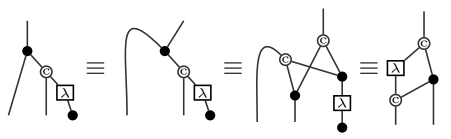


Now this world can be found nowhere else in the diagram, it is hence possible to sum the scalar with the other that has the same neighbourhood, obtained from (2).
It remains to deal with the scalars. We have $\lambda_0 mu_0$ on world ${}^{\lambda}_{0}w^{\mu}_{0}$ from what we just discussed. We also have scalar $\lambda''_i\mu'_i$ on world ${}^{\lambda''}_{i}w^{\mu'}_{i}$ from the dangling wires at the junction. All these worlds are again mutually exclusive and found nowhere else in the diagram. We can hence sum them and get $\lambda_0\mu_0 + \sum_i \lambda''_i\mu'_i$ as the scalar on the side. After all this, the diagram is in the appropriate form (up to renaming of worlds), with all worlds in the internal wires being mutually exclusive. ◀

▶ **Lemma 30.** *For any "matrix block" λ, there exists a "matrix block" ν such that:*

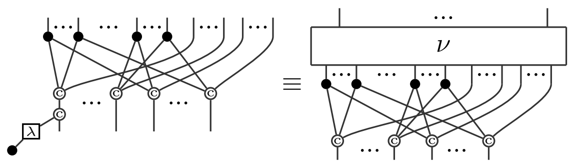


Proof. We show the result through several steps. First we show the following:



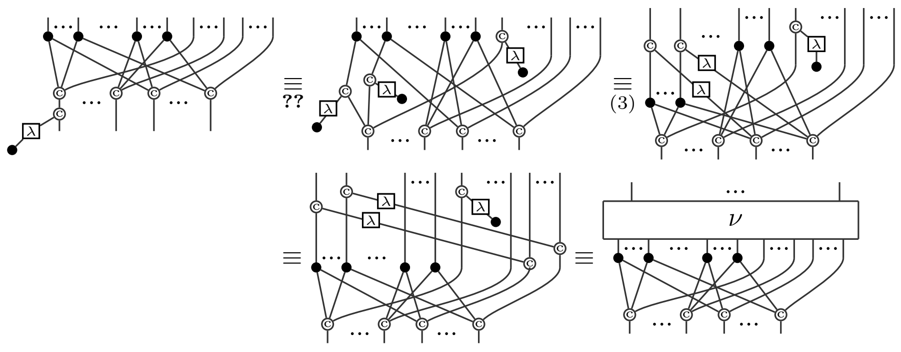
(3)

Then, we deal with the top dangling scalars and prove that:

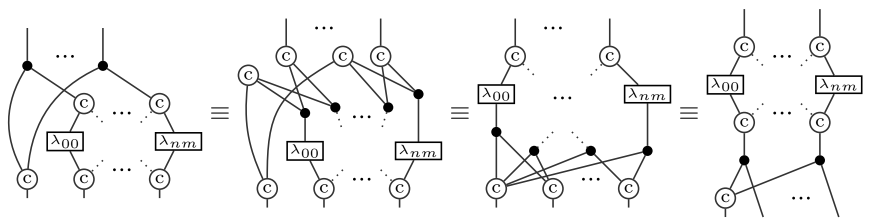


(4)

Indeed:

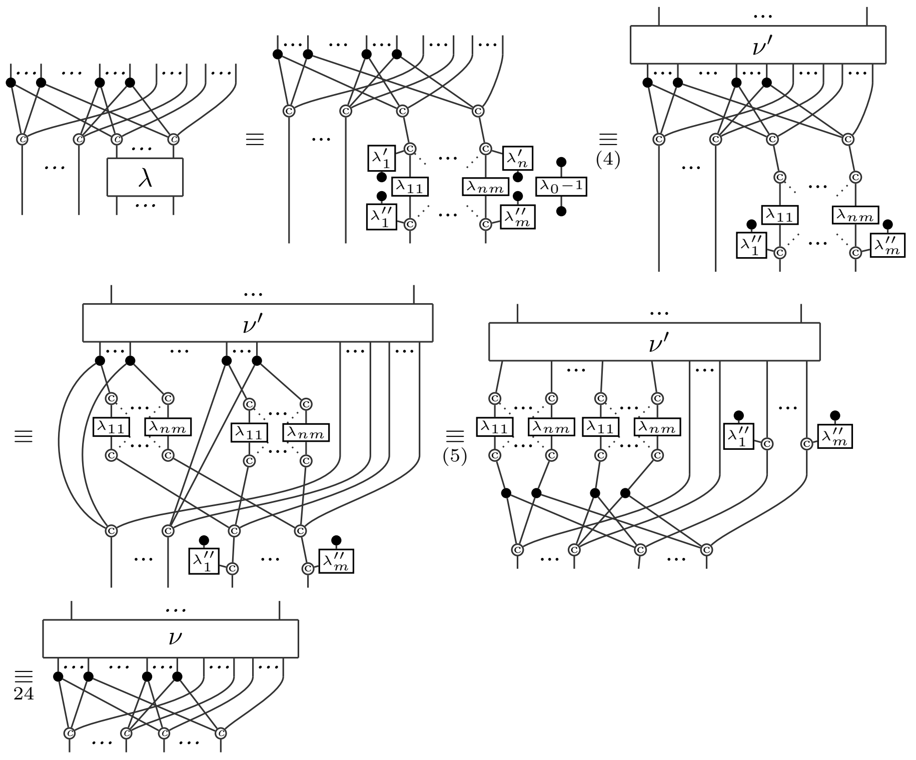


This case generalizes to any bottom wire the dangling scalar is applied on. When we have several of them, we can make them go through the top part in turns, then aggregate them under a single "matrix block" using Lemma 24.
We then derive the following equation:



(5)

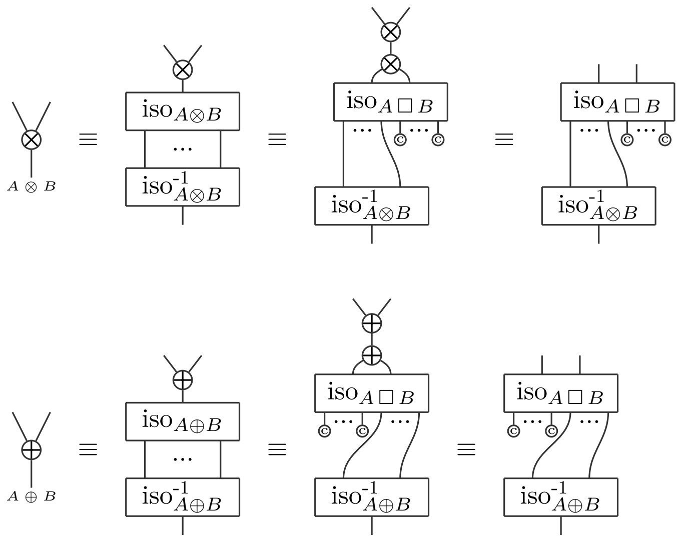
We can finally prove the lemma:



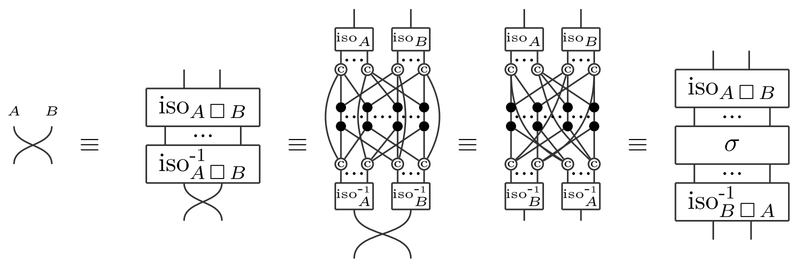
◀

We can now move on to show that generators can be put in normal form:

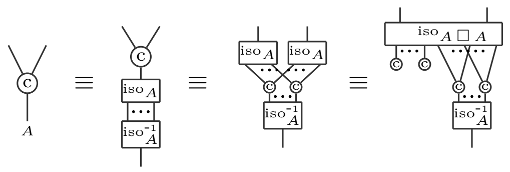
Proof of Proposition 11.

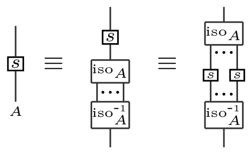


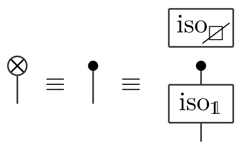
$A \otimes B$
≡
$\mathrm{iso}_{A\otimes B}$
$\mathrm{iso}^{-1}_{A\otimes B}$
$\mathrm{iso}_{A\,\square\,B}$
$A \oplus B$
$\mathrm{iso}_{A\oplus B}$
$\mathrm{iso}^{-1}_{A\oplus B}$

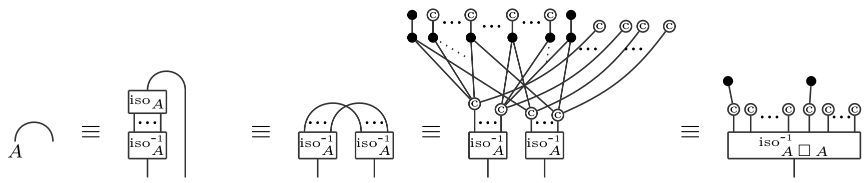


with σ a simple permutation of wires.





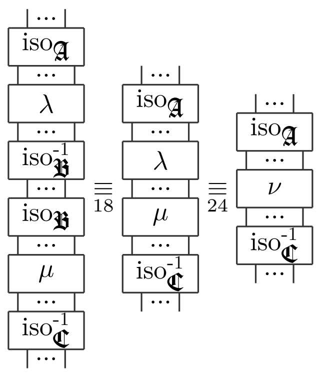




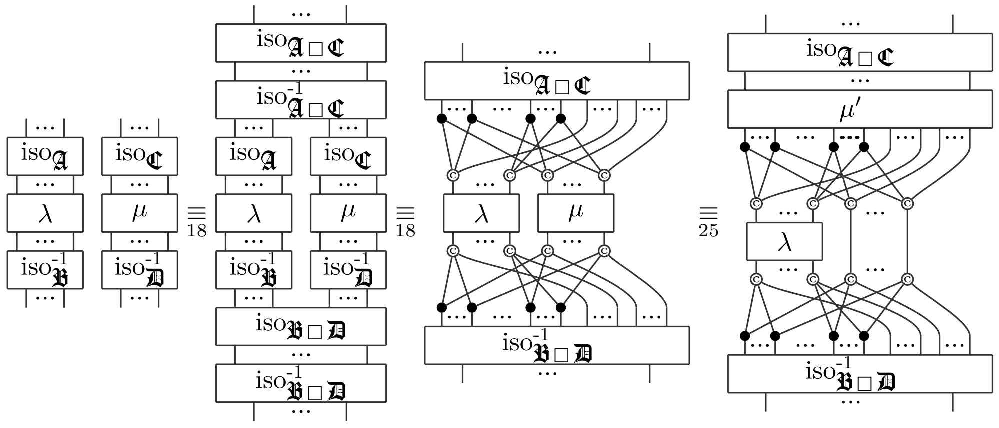
The upside-down versions of the generators are provided in exactly the same way (but upside-down). ◀

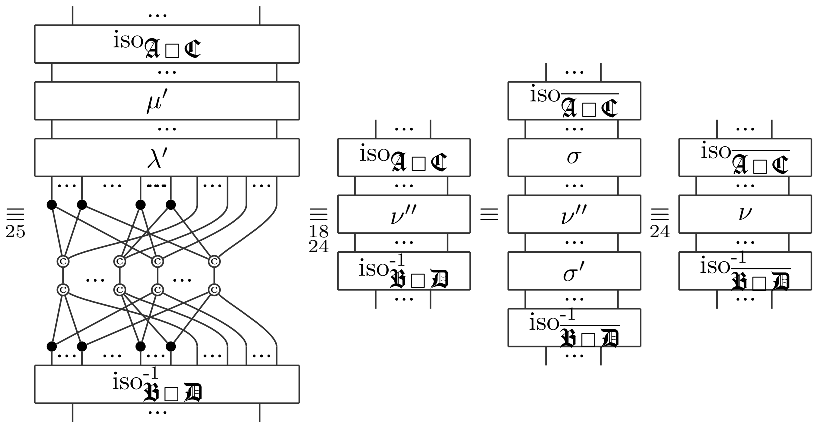
And then that compositions of diagrams in normal form can be put in normal form:

Proof of Proposition 12. In the case of sequential composition:


$\mathrm{iso}_{\mathfrak{A}}$
λ
$\mathrm{iso}^{-1}_{\mathfrak{B}}$
$\mathrm{iso}_{\mathfrak{B}}$
μ
$\mathrm{iso}^{-1}_{\mathfrak{C}}$
$\equiv_{18}$
$\mathrm{iso}_{\mathfrak{A}}$
λ
μ
$\mathrm{iso}^{-1}_{\mathfrak{C}}$
$\equiv_{24}$
$\mathrm{iso}_{\mathfrak{A}}$
ν
$\mathrm{iso}^{-1}_{\mathfrak{C}}$

In the case of parallel composition:


iso$_{\mathfrak{A}}$
λ
iso$^{-1}_{\mathfrak{B}}$
iso$_{\mathfrak{C}}$
μ
iso$^{-1}_{\mathfrak{D}}$
≡ 18
iso$_{\mathfrak{A}\square\mathfrak{C}}$
iso$^{-1}_{\mathfrak{A}\square\mathfrak{C}}$
iso$_{\mathfrak{B}\square\mathfrak{D}}$
iso$^{-1}_{\mathfrak{B}\square\mathfrak{D}}$
≡ 18
≡ 25
μ′


≡ 25
iso$_{\mathfrak{A}\square\mathfrak{C}}$
μ′
λ′
iso$^{-1}_{\mathfrak{B}\square\mathfrak{D}}$
≡ 18 24
ν′′
≡
iso$_{\overline{\mathfrak{A}\square\mathfrak{C}}}$
σ
σ′
iso$^{-1}_{\overline{\mathfrak{B}\square\mathfrak{D}}}$
≡ 24
ν

where $\overline{\mathfrak{A}\square\mathfrak{C}}$ represents the canonical choice of composition with $\square$ (and similarly for $\overline{\mathfrak{B}\square\mathfrak{D}}$).

◀