Logical Methods in Computer Science
Volume 21, Issue 2, 2025, pp. 13:1–13:44
https://lmcs.episciences.org/

Submitted Dec. 01, 2023
Published May 16, 2025

THE MANY-WORLDS CALCULUS

KOSTIA CHARDONNET [a,d], MARC DE VISME [b], BENOÎT VALIRON [c], AND RENAUD VILMART [b]

[a] Department of Computer Science and Engineering, University of Bologna, Italy.
e-mail address: kostia.chardonnet@pm.me

[b] Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France.
e-mail address: {mdevisme, rvilmart}@lmf.cnrs.fr

[c] Université Paris-Saclay, CNRS, CentraleSupélec, ENS Paris-Saclay, Inria, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France.
e-mail address: benoit.valiron@universite-paris-saclay.fr

[d] Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract. In this paper, we explore the interaction between two monoidal structures: a multiplicative one, for the encoding of pairing, and an additive one, for the encoding of choice. We propose a colored PROP to model computation in this framework, where the choice is parameterized by an algebraic side effect: the model can support regular tests, probabilistic and non-deterministic branching, as well as quantum branching, i.e. superposition.

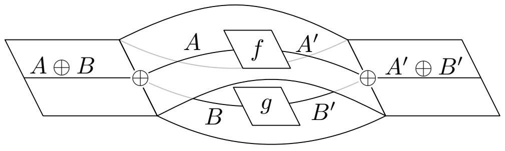
The graphical language comes equipped with a denotational semantics based on linear maps, and an equational theory. We prove the language to be universal, and the equational theory to be complete with respect to this semantics.

1. Introduction

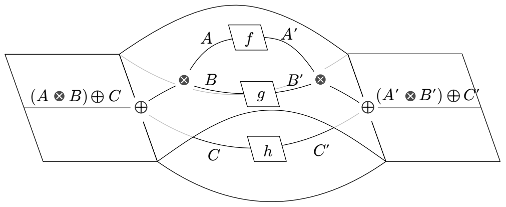
The basic execution flow of a computation is arguably based on three notions: sequences, tuples and branches. Sequences form the building block of compositionality, tuples are what makes it possible to consider multiple pieces of information together, while branches allow the behavior to change depending on the inputs or on the state of the system.

Ranging from (sometimes informal) flow-chart languages [Bar77] to sophisticated structures such as interaction or proof nets [Laf89, Gir96], graphical languages are commonly used to represent the possible control flow of a computation. On a formal level, a graphical language is a PROP [Lac04], that is, a symmetric, strict monoidal structure $(\mathcal{C}, \|, \varnothing)$ whose objects are of the form $W \| \cdots \| W$, with $\varnothing$ the unit for $\|$. The object W is a “wire”, and any object stands for a bunch of wires. The monoidal structure $\|$ formalizes how the bunching of wires behaves. We note that the usual symbol for a monoidal structure is $\otimes$, but for clarity we will avoid using it in situations where it is not the usual tensor product.

Key words and phrases: Quantum Computing, Graphical Languages.



(A) Split over Coproduct



(B) Splits over Coproduct and Tensor

Figure 1. Examples of Branchings

Two Canonical Monoidal Structures. The monoidal structure of a PROP is very versatile. On one hand, it can be considered in a *multiplicative* way, with $A \parallel B$ seen as the pairing of an element of type A and an element of type B. This approach is one followed in the design of MLL proof-nets for instance [Dal17, Gir96]. On the other hand, one can consider the monoidal structure in an *additive* way, with $\parallel$ for instance being a co- or a bi-product. Standard examples are the category **FinRel** of finite sets and relations, forming an additive PROP with $\parallel$ being the disjoint union, or the category of finite dimensional vector spaces (or semimodules[1]) and linear maps, with $\parallel$ being the cartesian product. From a computational perspective, an additive monoidal structure can be regarded as the possibility to *choose* a computational path upon the state of the input. Depending on the underlying system, this choice can be regarded as deterministic (if based on Set), non-deterministic (if based on Rel), probabilistic (if based on a suitable semimodule), *etc.*

To be able to handle both pairing and branching in a PROP, we cannot uniquely identify $\parallel$ as being multiplicative additive. We instead need to *extend* the PROP with two additional structures, one for pairing[2] ($\otimes$) and one for branching ($\oplus$).

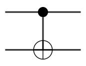
In this paper, we focus on a framework where these two monoidal structures are available. Graphical languages for such a setting usually rely on a notion of *sheet*, or *worlds*, to handle general branching [Dun09, Mel14]. Figure 1a shows for instance how to represent the construction of the morphism $f \oplus g : A \oplus B \to A' \oplus B'$ out of $f : A \to A'$ and $g : B \to B'$. The symbol "$\oplus$" stands for the "split" of worlds. Such a graphical language therefore comes with two distinct "splits": one for the monoidal structure —living inside one specific world—, and one for the coproduct —splitting worlds—. They can be intertwined, as shown in Figure 1b. Another approach followed by [CDH20] externalizes the two products (tensor product and coproduct) 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 deterministic or probabilistic branching [Dal17, Dun09, Sta15]. These existing approaches do not support more exotic branchings, such as *quantum superposition.*

Quantum Computation. 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 coprocessor accessed through a simple interface: The coprocessor holds individually addressable registers consisting of *quantum* bits, on which one can apply a fixed

[1]We consider *free* finite dimensional semimodules, that is semimodules that have a finite basis, which allow a matricial representation of morphisms, and not just a finite generating set.

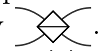
[2]Once again we avoid the usual symbol $\otimes$ and reserve it for situation in which it is the usual tensor product.

set of operations —*gates*, e.g. the *CNot* gate  that can be applied on pairs of qubits— specified by the interface. If some of these gates can generate superposition of data, this is kept inside the coprocessor and opaque to the programmer. A typical interaction with the coprocessor 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 coprocessor.

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 (pure) quantum circuit is a linear, sequential description of elementary operations describing a *linear, unitary map* on the state space.

Coming all the way from Feyman's diagrams [FH65], graphical languages are commonly used for representing quantum processes. Whether directly based on quantum circuits [GLR$^+$13, Dal17, PRZ17, CBB$^+$21] or stemming from categorical analysis such as the ZX-calculus [CK17], these formal languages are still tied to the quantum coprocessor model in the sense that the only monoidal structure that can be applied to quantum information is the (multiplicative) Kronecker product. The only possible branching is based on the (probabilistic) measurement.

Quantum Control Flow. However, quantum computation is not always reducible to the quantum coprocessor model, and superposition can be generalized to computation, yielding *non-causal execution paths.* Indeed, in general, the quantum computational paradigm features two seemingly distinct notions of control structure. On the one hand, a quantum program follows *classical* control: it is hosted on the conventional computer governing the coprocessor, 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 coprocessor model, permitting more elaborate *purely quantum* computational constructs than what quantum circuits or ZX-calculus allow.

The archetypal example of a quantum computational behavior hardly attainable within quantum circuits or ZX-calculus is the *Quantum Switch* [CDPV13]. 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 $|1\rangle$, all while making a single use of U and a single use of V. As x can be in superposition, in general the operation is then sending $(\alpha\,|0\rangle + \beta\,|1\rangle) \otimes |y\rangle$ to $\alpha\,|0\rangle \otimes (UV\,|y\rangle) + \beta\,|1\rangle \otimes (VU\,|y\rangle)$. While such an operation is physically realizable [PMA$^+$15, TCM$^+$21], it requires to move away from the standard quantum circuit-model [CDPV13]. It is a *purely quantum test*: not only can we have values in superpositions (here, x) but also *execution orders.* This is allowed by physical components that have no counterpart in the quantum coprocessor model, such as the *polarizing beam splitter*, represented by .

Computational models supporting superpositions of execution orders form an active subject of research in the literature. One strand of research consists in proposing a suitable extension of quantum circuits [CDP08, PMM$^+$17, VKB21, WDAB21]. These approaches typically aim at discussing the notion of quantum channel from a quantum information theoretical standpoint. Another strand is to focus on the additive aspect of quantum computation as in [CP20] which focusses on the polarizing beam splitter. However, here,

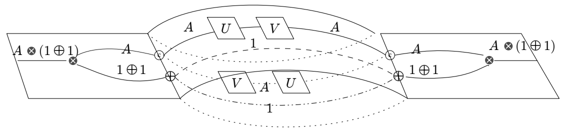
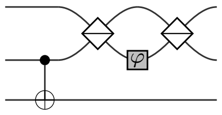


Figure 2. Quantum Switch with Worlds

similarly to the coprocessor model but with reversed roles, it is the multiplicative part that is hidden inside the primitives.

Limitation of Current Approaches and Objective of the Paper. Although there is a finer and finer understanding of superposition of causal orders in the literature, none of the existing PROPs can support the quantum switch on complex data built from tensors and coproducts at a fine-grain level, which becomes necessary when trying to integrate indefinite causal orders to usual tensor-based models of quantum computation. Our aim in this paper is to give an adequate framework for understanding non-ambiguously diagrams such as:



By doing so, we shall be able to express operators composed of polarising beam splitters, but where (i) the controlled system is explicit, and there are no black boxes as primitives, and (ii) the controlled system can become the control system later on, and vice-versa; harnessing the power of both indefinite causal order-enabling primitives and superposition-inducing quantum gates. Having fine-grained primitives both quantum data and quantum control flow moreover shall permit better implementation capabilities and automated reasoning on programs making use of them.

We claim in this paper that the same intuition underlying probabilistic branching can be followed for quantum (and more general) branching. In the conventional case, $1 \oplus 1$ is a regular boolean: either "left" (standing e.g. for True) *or* "right" (standing e.g. for False). In quantum computation, the sum-type $1 \oplus 1$ 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 [AG05, SVV18].

The Quantum Switch can then 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 bit. Based on the value of the qubit (True or 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. However, while in Figure 2 two copies of U and V are required, we will see that in our system only one copy of each is needed.

Contributions. In this paper, we introduce a new graphical language for quantum computation, based on compact category with biproduct [HV19]. This language allows us to express any process with both pairing and a general notion of algebraic branching, encompassing deterministic, non-deterministic, probabilistic and quantum branching. We develop a denotational semantic and an equational theory, and prove its soundness and completeness with respect to the semantics. As a case-study, we show how the Quantum Switch can naturally be encoded in the language as well as the quantum programming language for quantum control presented in [SVV18]. Additionally, while in Figure 2, we represent the quantum switch with two syntactic copies of U and V, this is merely a pedagogical choice. As shown in Figure 22, we are able to represent the quantum switch with only one syntactic instance of U and V, allowing us to write an higher-order version of this Quantum Switch in Figure 23.

2. The Many-Worlds Calculus

Our calculus is parameterized by a commutative semiring $(R, +, 0, \times, 1)$. It can be instantiated by the complex numbers $(\mathbb{C}, +, 0, \times, 1)$ to represent pure quantum computations, the non-negative real numbers $(\mathbb{R}_{\geq 0}, +, 0, \times, 1)$ for probabilistic computations, or the booleans $(\{\text{False}, \text{True}\}, \text{OR}, \text{False}, \text{AND}, \text{True})$ for non-deterministic computations.

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.To do so, we introduce three categories (actually colored PROPs), each of which being a refinement on the previous one and addressing its main caveat. These three categories are recalled in the following table, for reference, but are developed in details in the rest of the section.

$\mathbf{C_D}$	$\mathbf{MW}_W$ for W a set of worlds	$\mathbf{MW}_\forall$
Colors: $A, B \ ::= \ \mathbb{0} \mid A \oplus B$ $\mid \ \mathbb{1} \mid A \otimes B$ *Objects:* $\mathcal{A} \ ::= \ \varnothing$ $\mid \ A_1 \parallel \cdots \parallel A_n$ *Morphisms:* sequential and parallel composition of elements Figure 3. *Compositions:* usual sequential and parallel composition of colored PROP.	*Colors:* $(A : w)$ with $A \in \text{Colors}(\mathbf{C_D})$, $w \subseteq W$ *Objects:* $(\mathcal{A}, \ell_{\mathcal{A}})$ with $\mathcal{A} \in \text{Objects}(\mathbf{C_D})$, $\ell_{\mathcal{A}} : \{A_i \in \mathcal{A}\} \to \mathcal{P}(W)$ *Morphisms:* (f, ℓ_f) with $f \in \mathbf{C_D}$, $\ell_f : \text{Wires}(f) \to \mathcal{P}(W)$ *Compositions:* usual sequential and parallel composition of colored PROP.	*Colors:* same as $\mathbf{C_D}$ *Objects:* same as $\mathbf{C_D}$ *Morphisms:* $f_W \in \text{Morphisms}(\mathbf{MW}_W)$ *Compositions:* $f_W \parallel g_V :=$ $(f^{-\times V} \parallel g^{W\times -})_{W\times V}$ $g_V \circ f_W :=$ $(g^{(W\times -)\cap Z} \circ f^{(-\times V)\cap Z})_Z$

Table 1. Overview of the introduced colored PROPs.

2.1. The Unlabeled Language.

We define our graphical language within the paradigm of colored PROP [Car21, HR15], meaning that a morphism is a diagram composed of nodes, or *generators*, linked to each other through *colored wires*, that are allowed to cross each others. Additionally, we assume that our colored PROP is compact closed and auto-dual, i.e. we allow to bend wires to obtain a Cup ($\overset{A\ \ A}{\cup}$) or a Cap ($\underset{A\ \ A}{\cap}$).

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 for $n \geq 0$, and the Scalar for s ranging over the commutative semiring R. Mirrored versions of those generators are defined as syntactic sugar through the compact closure, as shown for the

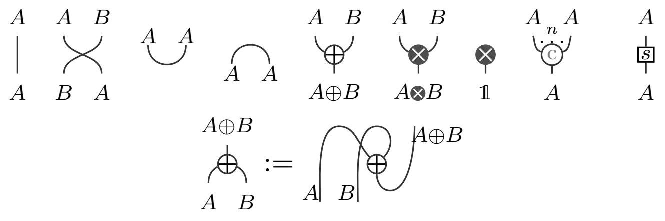
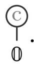
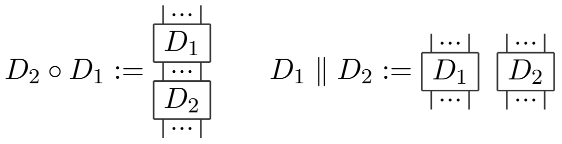


Figure 3. Generators of our Unlabeled Language ($n \geq 0$, $s \in R$)

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 labels A, B, etc correspond to the colors of our PROP. There is one color for every type generated by the syntax[3] $A, B ::= \mathbb{1} \mid \mathbb{0} \mid A \oplus B \mid A \otimes B$. As such, the Unit starts a wire of type $\mathbb{1}$, the Plus combines two wires of type A and B into a wire of type $A \oplus B$, and similarly the Tensor combines two wires of type A and B into a wire of type $A \otimes B$.

In a colored PROP, an object $\mathcal{A}$ is simply a list of colors $\mathcal{A} = A_1 \parallel \cdots \parallel A_n$ (or $\mathcal{A} = \varnothing$ for the empty collection). For example, the Plus is a morphism from $A \parallel B$ to $A \oplus B$. The choice of the notation $\parallel$ 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, $A \parallel B$ can be understood semantically as "either $A \otimes B$ or $A \oplus B$".

Diagrams are obtained from generators given in Figure 3 and by composing them in parallel (written $\parallel$), or sequentially (written $\circ$). Sequential composition requires the type (and number) of wires to match. Notice that there is no generator for the type $\mathbb{0}$, as it is a special case of the contraction with $n = 0$ and on type $\mathbb{0}$, i.e. . We write $\mathbf{C_D}$ for the category of diagrams we defined as such, and $\mathbf{C_D}(\mathcal{A}, \mathcal{B})$ for the set of diagrams that are morphisms from $\mathcal{A}$ to $\mathcal{B}$.

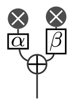


Example 2.1. We consider $R = \mathbb{C}$. While $\mathbf{C_D}$ lacks the worlds labeling[4], we can already illustrate our language by encoding some basic quantum primitive in it and show how they operate. In Figure 4 we show the encoding of a quantum bit $\alpha\left|0\right\rangle + \beta\left|1\right\rangle$ 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 $\left|0\right\rangle$ and the right branch to $\left|1\right\rangle$. 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 5, however to correctly define our equational theory, the worlds

[3]Those types are purely syntactic. We do not assume any associativity/distributivity/etc at the syntactic level. However, we have morphisms representing those properties (e.g. a morphism from $(A \oplus B) \oplus C$ to $A \oplus (B \oplus C)$), and up to the equational theory $\equiv$ those morphisms will be isomorphisms.

[4]We call worlds labeling the attribution of *multiple* worlds to a single wire, as defined in Section 2.3.

labeling are required. So while this specific worlds-free rewriting sequence is sound, many other similar worlds-free rewriting sequences are unsound.

$\alpha\,|0\rangle + \beta\,|1\rangle \rightsquigarrow$  $\in \mathbf{C_D}(\varnothing, \mathbb{1} \oplus \mathbb{1})$

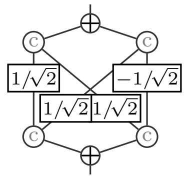
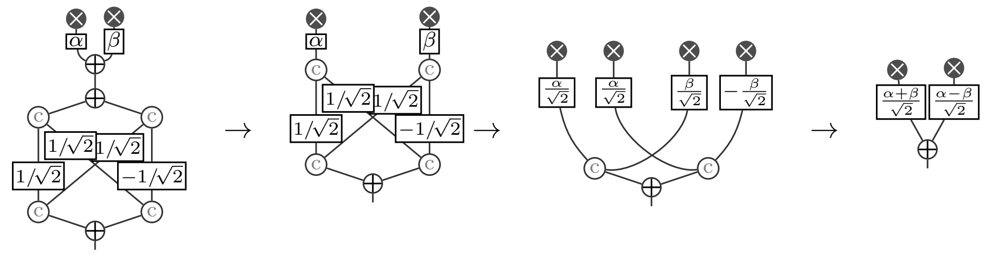
$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix} \rightsquigarrow$  $\in \mathbf{C_D}(\mathbb{1} \oplus \mathbb{1}, \mathbb{1} \oplus \mathbb{1})$

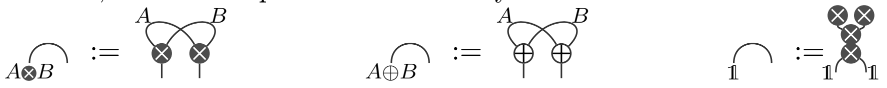
Figure 4. A Quantum Bit and the Hadamard Unitary



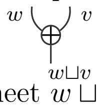
α
β
$1/\sqrt{2}$
$1/\sqrt{2}$
$1/\sqrt{2}$
$-1/\sqrt{2}$
$\rightarrow$
$\frac{\alpha}{\sqrt{2}}$
$\frac{\alpha}{\sqrt{2}}$
$\frac{\beta}{\sqrt{2}}$
$-\frac{\beta}{\sqrt{2}}$
$\frac{\alpha+\beta}{\sqrt{2}}$
$\frac{\alpha-\beta}{\sqrt{2}}$

Figure 5. Applying the Hadamard Unitary to a Quantum Bit

Remark 2.2. 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:



2.2. **Sheets and 3 Dimensional Diagrams.** We said that $A \parallel B$ corresponds to "either $A \oplus B$ or $A \otimes B$". There are circumstances, in particular when trying to define an equational theory, where it is useful to know which one of the two it is. Following [CDH20], we could draw our diagrams on sheets, leading to a 3D figures like Figures 1a, 1b and 2. In those figures, wires that are in parallel sheets are $\oplus$-related while wires that are in the same sheets are $\otimes$-related. Additionally, the nodes $\oplus$ and © allow for parallel sheets to be merged together, while every other generator has no effect on the sheet structure.

However, drawing in 3D can quickly get messy, especially when one needs to make sheets cross each others (see Figure 6). A compromise is to instead annotate each wire by the name of the sheet it is on, so for example  would mean that we have an $\oplus$ node merging the sheet w and the sheet v into the sheet $w \sqcup v$. Going even further, we can rewrite the notion of sheets into a notion of worlds:

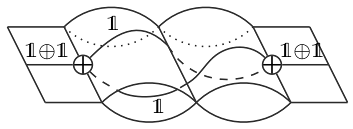


Figure 6. The Negation from $\mathbb{1} \oplus \mathbb{1}$ to $\mathbb{1} \oplus \mathbb{1}$.

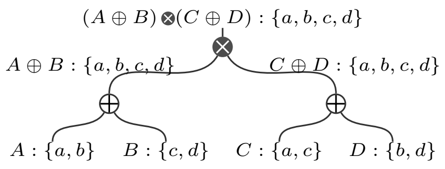
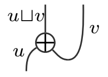


Figure 7. The Wires of Type B and of Type C are Neither $\otimes$-Related or $\oplus$-Related.

- A sheet corresponds to a set of worlds. We annotate every wire of the sheet by its world set. Wires that are $\otimes$-related have the same annotation.
- Parallel sheets are disjoint sets of worlds. Wires that are $\oplus$-related have annotations that are disjoints sets of worlds. Merging two sheets w and v correspond to a disjoint union $w \sqcup v$.
- Wires that are world sets that are neither equal or disjoint are neither $\otimes$-related or $\oplus$-related, but are instead of mixture of both. One notable case where this situation happen is when decomposing a wire of type $(A \oplus B) \otimes (C \oplus D)$ into four wires of types A, B, C and D, as in Figure 7. The possibility to have world sets in this situation also allows us to have a very lax compact structure, where wires can be bent at will, e.g. in 

2.3. **Adding Worlds Labeling.** We now label wires of our diagram with sets of worlds $w \subseteq W$ from a given world set 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 us to correlate the enabling of wires. Before making this formal, we illustrate this notion through the following example:

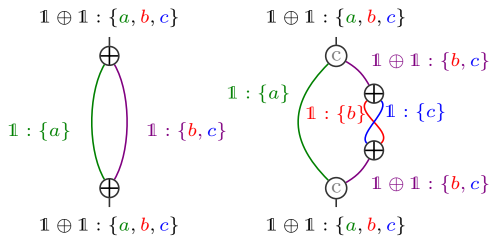
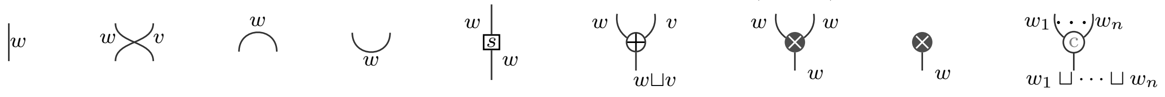


Figure 8. Controlled Not with World Set $\{a, b, c, \star\}$

Example 2.3. The "Controlled Not" on quantum bits can be represented by the Figure 8. 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 a to the case where the control quantum bit is $|0\rangle$, and the worlds b and c to the case where the control quantum bit is $|1\rangle$. Lastly, the world $\star$ appears nowhere in the labels, and corresponds to "we do not evaluate this circuit at all"[5]. On the computational part, we apply the identity within the world a, we negate $|0\rangle$ into $|1\rangle$ within b, and we negate $|1\rangle$ into $|0\rangle$ within c. The domain and codomain of this labeled diagram are $(\mathbb{1} \oplus \mathbb{1} : \{a, b, c\}) \parallel (\mathbb{1} \oplus \mathbb{1} : \{a, b, c\})$, which we will write $(\mathcal{A}, \ell_{\mathcal{A}})$ with an object $\mathcal{A} = (\mathbb{1} \oplus \mathbb{1}) \parallel (\mathbb{1} \oplus \mathbb{1})$ and a labeling function $\ell_{\mathcal{A}} : 1 \mapsto \{a, b, c\} \quad 2 \mapsto \{a, b, c\}$. Similarly, the "Controlled Not" above can be seen as a diagram $\mathcal{D}_{\text{CNOT}}$ of $\mathbf{C_D}(\mathcal{A}, \mathcal{A})$ together with a labeling function $\ell_{\mathcal{A}}$ which labels every wire with a set of worlds.

We now give a formal definition of the concept of worlds:

Definition 2.4. Given a set of worlds W, we define the auto-dual compact closed colored PROP $(\mathbf{MW}_W, \parallel, \varnothing)$ 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 $(\mathcal{A}, \ell_{\mathcal{A}})$ for the objects, where $\mathcal{A}$ is an object of $\mathbf{C_D}$ and $\ell_{\mathcal{A}}$ is a labeling function from the colors[6] composing $\mathcal{A}$ to the subsets of W.
Its morphisms $f \in \mathbf{MW}_W((\mathcal{A}, \ell_{\mathcal{A}}), (\mathcal{B}, \ell_{\mathcal{B}}))$ are pairs $(\mathcal{D}_f, \ell_f)$ of a morphism $\mathcal{D}_f \in \mathbf{C_D}(\mathcal{A}, \mathcal{B})$ 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



for all $n \geq 0$ and where $\sqcup$ denotes disjoint set-theoretic union. The constraints for the mirrored versions are similar. The sequential composition $\circ$ and the parallel composition $\parallel$ preserve the labels.

We write $f_W : \mathcal{A} \to \mathcal{B}$ for $f = (\mathcal{D}_f, \ell_f)$ a morphism of $\mathbf{MW}_W((\mathcal{A}, \ell_{\mathcal{A}}), (\mathcal{B}, \ell_{\mathcal{B}}))$, where $\ell_{\mathcal{A}}$ and $\ell_{\mathcal{B}}$ can be deduced from ℓ_f using the first restriction on labels. We note that when considering $f_W : \mathcal{A} \to \mathcal{B}$ and $g_W : \mathcal{B} \to \mathcal{C}$, there is no reason for the labels deduced on $\mathcal{B}$ to be the same, so there is no reason for $(g \circ f)_W : \mathcal{A} \to \mathcal{C}$ to be defined. And even when $(g \circ f)_W$ is defined, it might not have the meaning we intend: for example, Figure 9 shows a "broken" composition of the negation with itself, as the world b corresponds to "the input of the first negation is $|0\rangle$, its output is $|1\rangle$, then in an contradictory way the input of the second negation is back at $|0\rangle$ and its output is $|1\rangle$", which does not make any computational sense. In fact, if we use the equational theory $\equiv$ defined in Section 5, the semantics of this diagram collapses to zero.

In practical terms, the naive composition of $\mathbf{MW}_W$ can be used to decompose a diagram into fragments of said diagrams, but is ill-suited to assemble independent diagrams together.

2.4. **World-Agnostic Composition.**

When building two diagrams f_W and g_V, there is no reason for the worlds sets W and V to be equal, but we might still want to be able to compose them with one another, hence the need for a new kind of composition that better handles the world sets.

[5]While not strictly necessary, it is often practical to have a world absent from every wire. We discuss its usefulness in Remark 2.5.

[6]If the same color appears multiple times in the object, each instance might have a different label.

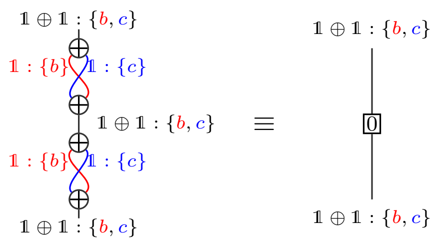

$\mathbb{1} \oplus \mathbb{1} : \{b, c\}$
$\mathbb{1} : \{b\}$
$\mathbb{1} : \{c\}$
$\mathbb{1} \oplus \mathbb{1} : \{b, c\}$
$\equiv$
0
$\mathbb{1} : \{b\}$
$\mathbb{1} : \{c\}$
$\mathbb{1} \oplus \mathbb{1} : \{b, c\}$
$\mathbb{1} \oplus \mathbb{1} : \{b, c\}$
$\mathbb{1} \oplus \mathbb{1} : \{b, c\}$

Figure 9. "Broken" Composition of two Negations

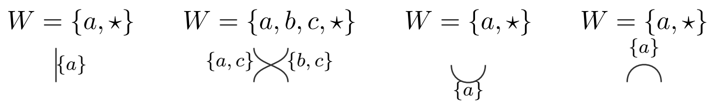


Figure 10. Canonical Labelings in $\mathbf{MW}_\forall$

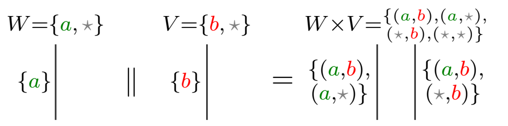


Figure 11. First Example of Worlds-Agnostic Composition

Definition 2.5. We define the auto-dual compact closed colored PROP $(\mathbf{MW}_\forall, \|, \varnothing)$ of Many-Worlds calculus as follows:

Its colors and *objects* are the same as the ones of $\mathbf{C_D}$.

Its morphisms from $\mathcal{A}$ to $\mathcal{B}$ are simply morphisms $f_W : \mathcal{A} \to \mathcal{B}$ of $\mathbf{MW}_W$ for any finite set W. See Figure 10 for the canonical labelings on the identity, swap, cup and cap. Morphisms are considered up to renaming of the worlds[7].

The parallel composition is given by $f_W \parallel g_V := (f^{-\times V} \parallel g^{W \times -})_{W\times V}$ where $f^{\sigma(-)}$ has the same diagram $\mathcal{D}_f$ and has for labels $\ell_{f^{\sigma(-)}}(x) = \sigma(\ell_f(x))$.

The sequential composition is given by $g_V \circ f_W := (g^{(W\times -)\cap Z} \circ f^{(-\times V)\cap Z})_Z$ where $Z \subseteq W \times V$ is the greatest subset such that this composition is well-defined, as illustrated by the following example and explained just after.

Example 2.6 (World Agnostic Composition)**.** In Figure 11, we show the result of the parallel composition of $f_{\{a,\star\}} = \mathbf{id}_{A:\{a\}}$ and $g_{\{b,\star\}} = \mathbf{id}_{A:\{b\}}$ for a color A. The world $(a, \star)$ corresponds to the left wire being enabled and the right one disabled, the world $(\star, b)$ is the opposite, the world (a, b) corresponds to both wires being enabled and the world $(\star, \star)$ to both disabled.

Then, in Figure 12 we continue by composing with the Cup over $A : \{c, \star\}$. To compute the composition, we proceed in two steps: first we handle the situation as if it were a parallel composition, leading to a diagram labeled over $W \times V \times U$, but with multiple contradictory

[7]More precisely, $(\mathcal{D}_f, \ell_f)_W = (\mathcal{D}_f, \sigma \circ \ell_f)_{\sigma(W)}$ for any σ describing a bijection between W and $\sigma(W)$. Without this quotient, we would not have $f_W \circ \mathbf{id} = f_W$.

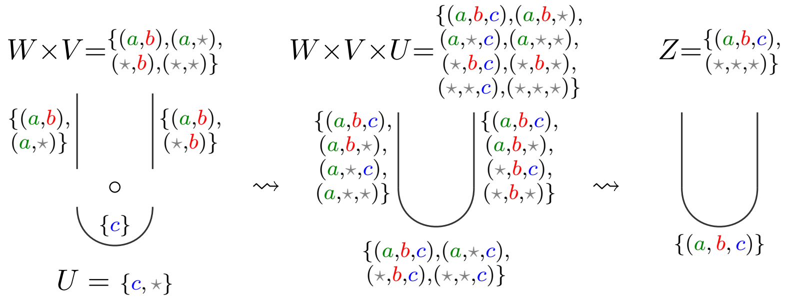

$W\times V=\{(a,b),(a,\star),(\star,b),(\star,\star)\}$
$\{(a,b),(a,\star)\}$
$\{(a,b),(\star,b)\}$
$\circ$
$\{c\}$
$U=\{c,\star\}$
$W\times V\times U=\{(a,b,c),(a,b,\star),(a,\star,c),(a,\star,\star),(\star,b,c),(\star,b,\star),(\star,\star,c),(\star,\star,\star)\}$
$\{(a,b,c),(a,b,\star),(a,\star,c),(a,\star,\star)\}$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

FIGURE 12. Second Example of Worlds-Agnostic Composition

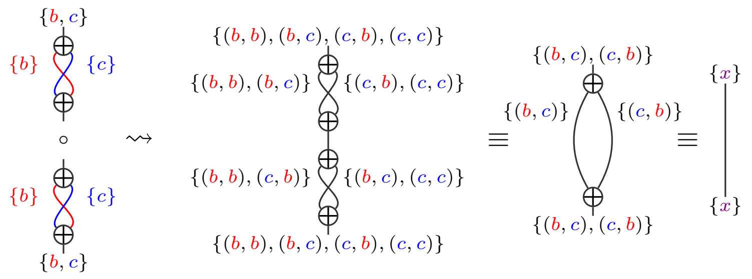


FIGURE 13. World-Agnostic Composition of two Negations

labels on the wire. Then, we eliminate as many worlds as necessary 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.
- Looking at the *right*, we eliminate $(\star, b, \star)$ and $(a, \star, c)$ for similar reasons.

We eliminated six worlds, with the only remaining ones being $Z = \{(a, b, c), (\star, \star, \star)\}$.

The above procedure can be generalized for any $g_V \circ f_W$:

- We start by computing $g^{W\times -}$ and $f^{-\times V}$.
- Then we forcefully append the two diagrams as if one was composing them as morphisms of $\mathbf{MW}_{W\times V}$. We write h for the resulting diagram, although it will rarely be a valid diagram of $\mathbf{MW}_{W\times V}$, as some wires might be labeled by multiple contradictory sets of worlds.
- For each wire e_i, we consider the various sets of worlds labeling it, let us name them $w_1^i, \dots, w_n^i$, and write $w^i = \bigcap_{k=1}^n w_k^i$. In order to make the different labelings of the wire e_i consistent with each others, we need to remove all the worlds of $w_k^i \backslash w^i$ for all k.
- So writing $u = \bigcup_i \bigcup_{k=1}^n (w_k^i \backslash w^i)$, the restricted diagram $h\backslash u$ is a valid diagram of $\mathbf{MW}_{(W\times V)\backslash u}$. In fact, this diagram $h\backslash u$ is equal to the composition $(g^{W\times -}\backslash u) \circ (f^{-\times V})\backslash u$.
- Writing $Z = (W \times V)\backslash u$, $h\backslash u$ is also equal to $(g^{(W\times -)\cap Z} \circ f^{(-\times V)\cap Z})$. We can then take:
$$g_V \circ f_W := \left(g^{(W\times -)\cap Z} \circ f^{(-\times V)\cap Z}\right)_Z$$
- And by construction, $Z = (W \times V)\backslash u$ is indeed the greatest subset of $W \times V$ such that this composition is well-defined.

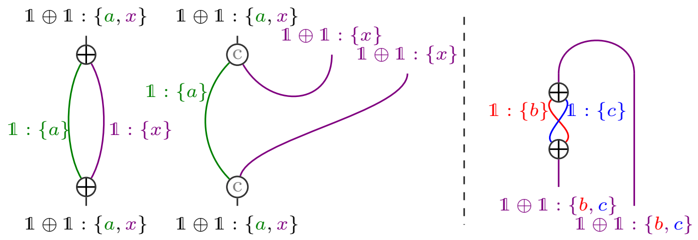

$\mathbb{1} \oplus \mathbb{1} : \{a, x\}$
$\mathbb{1} : \{a\}$
$\mathbb{1} : \{x\}$
$\mathbb{1} \oplus \mathbb{1} : \{x\}$
$\mathbb{1} : \{b\}$
$\mathbb{1} : \{c\}$
$\mathbb{1} \oplus \mathbb{1} : \{b, c\}$

Figure 14. The Conditional **Cond** and the State Encoding the Negation $\mathbf{q}^{\mathrm{NOT}}$, with Respective World Sets $\{a, x, \star\}$ and $\{b, c, \star\}$.

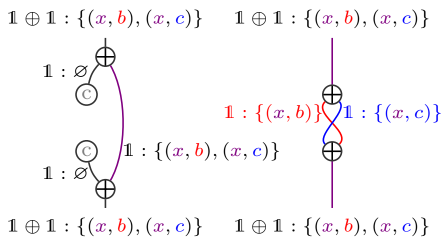

$\mathbb{1} \oplus \mathbb{1} : \{(x, b), (x, c)\}$
$\mathbb{1} : \varnothing$
$\mathbb{1} : \{(x, b), (x, c)\}$
$\mathbb{1} : \{(x, b)\}$
$\mathbb{1} : \{(x, c)\}$

Figure 15. The "Broken" CNot Obtained when Composing Without the $\star$ World. Note that every world of the form $(a, -)$ has been eliminated during the composition, and the corresponding wires have been eliminated using the equational theory $\equiv$.

Example 2.7 (Back to the Double Negation)**.** We show in Figure 13 the result of the world-agnostic composition of the negation on the worlds set $\{b, c\}$ with itself. The number of worlds grows significantly as all the pairs have to be considered, though the worlds (b, b) and (c, c) are meaningless computationally so the equational theory $\equiv$ allows us to remove them, leaving only the worlds (b, c) and (c, b) which correspond to the two different paths that the data can take through the double negation. Those two remaining worlds can then be merged together into a single world x using that same equational theory, giving the identity morphism as a final result.

2.5. **The Meaning of the $\star$ World.**

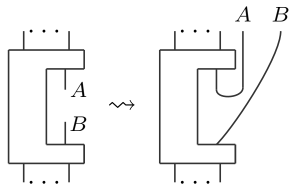
In most practical examples, we include a world which appears nowhere in the worlds annotations and we denote it $\star$. This world denotes the possibility of not evaluating the diagram at all, for example if no input is ever received. It is particularly useful in compositional approaches, where the diagram is in fact a fragment of a bigger diagram, that might not be used in some circumstances. For example Figure 14 represents the conditional **Cond** as well as the state $\mathbf{q}^{\mathrm{NOT}}$ encoding the negation, and should we compute $\mathbf{Cond}_{\{a,x,\star\}} \circ (\mathbf{id} \parallel \mathbf{id} \parallel \mathbf{q}^{\mathrm{NOT}}_{\{b,c,\star\}})$ we would obtain[8] the expected "Controlled Not" of Figure 8, while if we removed the $\star$ the semantics would collapse, the Not branch would then be forced and we would obtain[9] the result described in Figure 15.

[8]After a few rewriting steps as defined in Section 5, including the following renaming of the worlds: $(a, \star) \mapsto a$, $(x, b) \mapsto b$, $(x, c) \mapsto c$, $(\star, \star) \mapsto \star$.

[9]After a few rewriting steps as defined in Section 5.

Remark 2.8 (Quantum Conditionals)**.** This behavior is already known in quantum computation: quantum conditionals are known to be impossible to implement due to some no-go theorems [AFCB14], however, some implementations actually exists [ZRK$^+$11], this apparent contradiction can be solved by introducing a notion of sectors [VC21] to show that in the former the controlled operation was expected to "always be used" while in the latter it "might or might not be used".

Remark 2.9 (Supermaps and Currying)**.** Notice that the Many-Worlds can represent higher-order quantum processes, i.e. supermaps. This is done through its compact structure. Generaly, supermaps are represent with diagrams with holes (as shown below, on the left-hand-side) where the hole can then be filled by a diagram going from A to B. By bending the wire, we can transform this diagram into one that takes two additional input A and B.



For a concrete exemple, this is what happens in the left-hand-side diagram of Figure 14, where **Cond** is awaiting for a diagram from $\mathbb{1} \oplus \mathbb{1} \to \mathbb{1} \oplus \mathbb{1}$.

3. Representing Computation

As a motivational example, we may see how this Many-Worlds diagrams can be used to represent computations expressed in a language that explicitly uses the two compositions $\otimes$ (through pairs), and $\oplus$ (through pattern-matching), as in [SVV18, Cha23]. As this translation is not the focus of this paper, a lot of technicalities are glossed over. More details of this translation can be found in [Cha23].

Syntax of the Language. The language we present here is adapted from [SVV18], where we consider the values and types together with the enrichment that is linear combinations of terms, but without abstraction nor recursion. The syntax that is used in the language is given as follows, with scalars s ranging over the commutative semiring R:

$$\begin{array}{ll}
\text{(Base types)} & A, B ::= \mathbb{1} \mid A \oplus B \mid A \otimes B \\
\text{(Isos, first-order)} & \alpha ::= A \leftrightarrow B \\
\text{(Values)} & v ::= \langle\rangle \mid x \mid \texttt{inj}_l\, v \mid \texttt{inj}_r\, v \mid \langle v, v\rangle \\
\text{(Pattern)} & p ::= x \mid \langle p_1, p_2\rangle \\
\text{(Expressions)} & e ::= v \mid \texttt{let}\, p = \omega\ p\ \texttt{in}\ e \mid e + e \mid \alpha e \\
\text{(Isos)} & \omega ::= \{v_1 \leftrightarrow e_1 \mid \dots \mid v_n \leftrightarrow e_n\} \\
\text{(Terms)} & t ::= \langle\rangle \mid x \mid \texttt{inj}_l\, t \mid \texttt{inj}_r\, t \\
& \quad \mid \langle t_1, t_2\rangle \mid \texttt{let}\, p = t\ \texttt{in}\ t \\
& \quad \alpha t \mid t + t \mid \omega\ t
\end{array}$$

The language in particular features branching through the $\texttt{inj}_l$ and $\texttt{inj}_r$ constructors, linear combinations of terms and expressions, and crucially *isos* that have type $A \leftrightarrow B$: they

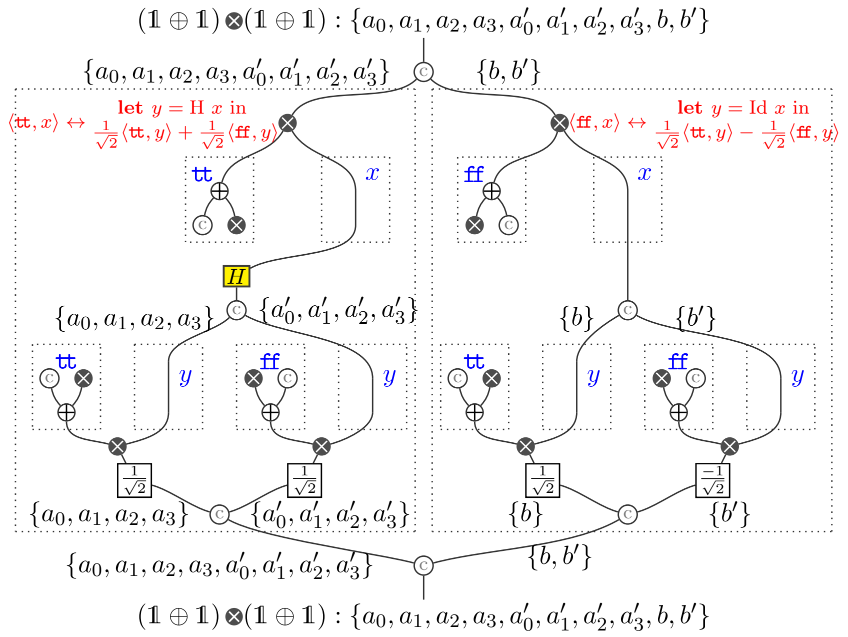

$(\mathbb{1}\oplus\mathbb{1})\otimes(\mathbb{1}\oplus\mathbb{1}) : \{a_0,a_1,a_2,a_3,a'_0,a'_1,a'_2,a'_3,b,b'\}$
$\{a_0,a_1,a_2,a_3,a'_0,a'_1,a'_2,a'_3\}$
$\{b,b'\}$
$\langle \mathtt{tt}, x\rangle \leftrightarrow$ let $y = \mathrm{H}\ x$ in $\frac{1}{\sqrt{2}}\langle \mathtt{tt}, y\rangle + \frac{1}{\sqrt{2}}\langle \mathtt{ff}, y\rangle$
$\langle \mathtt{ff}, x\rangle \leftrightarrow$ let $y = \mathrm{Id}\ x$ in $\frac{1}{\sqrt{2}}\langle \mathtt{tt}, y\rangle - \frac{1}{\sqrt{2}}\langle \mathtt{ff}, y\rangle$
tt
ff
x
H
y
$\{a_0,a_1,a_2,a_3\}$
$\{a'_0,a'_1,a'_2,a'_3\}$
$\{b\}$
$\{b'\}$
$\frac{1}{\sqrt{2}}$
$\frac{-1}{\sqrt{2}}$

Figure 16. Representation of the term t from Example 3.2.

turn a term of type A to a term of type B using pattern-matching. The language comes with a predicate (not presented here, although it could be given a diagrammatic meaning), used in the typing rule of isos, to ensure exhaustivity and the non-overlapping character of the left-hand and right-hand expressions of the clauses, allowing in particular to define unitaries (in the complex setting). Constraints on the linear combinations may also be used to enforce probabilistic constraints (i.e. that states are normalized in the quantum setting).

There are two different typing judgements, one for terms $\vdash_t$ and one specific for isos $\vdash_\omega$. In the following, we will use the shorthands $\mathtt{ff} := \mathtt{inj}_l\ \langle\rangle$ and $\mathtt{tt} := \mathtt{inj}_r\ \langle\rangle$.

Example 3.1. In the case where $R = \mathbb{C}$, one can encode the Hadamard and the CNOT gate by:

$$\mathrm{Hadamard} : \mathbb{1}\oplus\mathbb{1} \leftrightarrow \mathbb{1}\oplus\mathbb{1} = \left\{\begin{array}{l} \mathtt{ff} \leftrightarrow \frac{1}{\sqrt{2}}(\mathtt{ff}+\mathtt{tt}) \\ \mathtt{tt} \leftrightarrow \frac{1}{\sqrt{2}}(\mathtt{ff}-\mathtt{tt}) \end{array}\right\} \qquad \mathrm{CNOT} : (\mathbb{1}\oplus\mathbb{1})^{\otimes 2} \leftrightarrow (\mathbb{1}\oplus\mathbb{1})^{\otimes 2} = \left\{\begin{array}{lcl} \langle \mathtt{tt},\mathtt{ff}\rangle &\leftrightarrow& \langle \mathtt{tt},\mathtt{tt}\rangle \\ \langle \mathtt{tt},\mathtt{tt}\rangle &\leftrightarrow& \langle \mathtt{tt},\mathtt{ff}\rangle \\ \langle \mathtt{ff},x\rangle &\leftrightarrow& \langle \mathtt{ff},x\rangle \end{array}\right\}$$

In this setting, any quantum circuit can be encoded by an iso.

3.1. Encoding into the Many-Worlds.

One can encode any term of the language into a Many-Worlds diagram. Given some typing derivation ξ of a term $x_1 : A_1, \ldots, x_n : A_n \vdash_t t : B$ we write $(\!|\xi|\!)$ for the function that maps ξ to a diagram in the Many-Worlds Calculus with n input wires of type $A_1, \ldots, A_n$ and one output wire of type B. For the typing derivation ξ of an iso $\vdash_\omega \omega : A \leftrightarrow B$, $(\!|\xi|\!)$ gives a diagram with one input wire of type A and one output wire of type B.

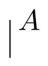
$(\!|\xi|\!)$ is defined inductively over $\vdash_t$ and $\vdash_\omega$ as shown in Figure 17, where the worlds sets are handled by world-agnostic compositions of diagrams with their canonical labelings. This encoding can be shown to be sound in regard to the programming language rewriting system [Cha23].

Example 3.2. We can represent the term

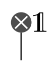
$$t := \left\{\begin{array}{l} \langle \mathtt{tt}, x\rangle \leftrightarrow \quad \mathbf{let}\ y = \mathrm{H}\ x \text{ in } \frac{1}{\sqrt{2}}\langle \mathtt{tt}, y\rangle + \frac{1}{\sqrt{2}}\langle \mathtt{ff}, y\rangle \\ \langle \mathtt{ff}, x\rangle \leftrightarrow \quad \mathbf{let}\ y = \mathrm{Id}\ x \text{ in } \frac{1}{\sqrt{2}}\langle \mathtt{tt}, y\rangle - \frac{1}{\sqrt{2}}\langle \mathtt{ff}, y\rangle \end{array}\right\}$$

of type $(\mathbb{1}\oplus\mathbb{1})^{\otimes 2} \leftrightarrow (\mathbb{1}\oplus\mathbb{1})^{\otimes 2}$ (already given in [SVV18]) as shown in Figure 16. The yellow box stands for the Hadamard gate (which one can build following Example 3.1). Each line of this isomorphism corresponds to a column of the figure. Each columns start by matching the input as $\langle \mathtt{tt}, x\rangle$ or $\langle \mathtt{ff}, x\rangle$, then computing y from x, and finally building the output by following the syntax. The world set is computed by composing each of the blocks of this term, using the world-agnostic composition of $\mathbf{MW}_\forall$. It can be seen as a subset of $\{a, b\} \times \{c, c'\} \times \{0, 1, 2, 3\}$ where $\{a, b\}$ corresponds to being on the first or second line of the matching, $\{c, c'\}$ being on the left or right of the sum, and $\{0, 1, 2, 3\}$ being the world set of the Hadamard gate.

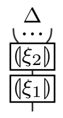
$$\left(\!\!\left(\frac{}{x : A \vdash_e x : A} \right)\!\!\right) =$$



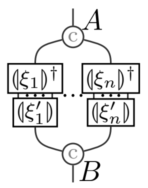
$$\left(\!\!\left(\frac{}{\vdash_e \langle\rangle : \mathbb{1}} \right)\!\!\right) =$$



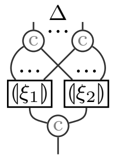
$$\left(\!\!\left(\frac{\frac{\xi_1}{\vdash_\omega \omega : A \leftrightarrow B} \qquad \frac{\xi_2}{\Delta \vdash_e t : A}}{\Delta \vdash_e \omega\ t : B} \right)\!\!\right) =$$



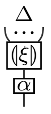
$$\left(\!\!\left(\frac{\frac{\xi_i}{\Delta_i \vdash_e v_i : A} \qquad \frac{\xi'_i}{\Delta_i \vdash_e e_i : B}}{\vdash_e \{v_1 \ \leftrightarrow\ e_1 \mid \dots \mid v_n \ \leftrightarrow\ e_n\} : A \leftrightarrow B} \right)\!\!\right) =$$



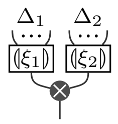
$$\left(\!\!\left(\frac{\frac{\xi_1}{\Delta \vdash_e t_1 : A} \qquad \frac{\xi_2}{\Delta \vdash_e t_2 : A}}{\Delta \vdash_e t_1 + t_2 : A} \right)\!\!\right) =$$



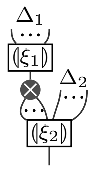
$$\left(\!\!\left(\frac{\frac{\xi}{\Delta \vdash_e t : A}}{\Delta \vdash_e \alpha t : A} \right)\!\!\right) =$$



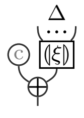
$$\left(\!\!\left(\frac{\frac{\xi_1}{\Delta_1 \vdash_e t_1 : A} \qquad \frac{\xi_2}{\Delta_2 \vdash_e t_2 : B}}{\Delta_1, \Delta_2 \vdash_e \langle t_1, t_2\rangle : A \otimes B} \right)\!\!\right) =$$



$$\left(\!\!\left(\frac{\frac{\xi_1}{\Delta_1 \vdash_e t_1 : A_1 \otimes \dots \otimes A_n} \qquad \frac{\xi_2}{x_1 : A_1, \dots, x_n : A_n, \Delta_2 \vdash_e t_2 : B}}{\Delta_1, \Delta_2 \vdash_e \mathtt{let}\ \langle x_1, \dots, x_n\rangle = t_1\ \mathtt{in}\ t_2 : B} \right)\!\!\right) =$$



$$\left(\!\!\left(\frac{\frac{\xi}{\Delta \vdash_e t : A}}{\Delta \vdash_e \mathtt{inj}_r\ t : A \oplus B} \right)\!\!\right) =$$



$$\left(\!\!\left(\frac{\frac{\xi}{\Delta \vdash_e t : B}}{\Delta \vdash_e \mathtt{inj}_l\ t : A \oplus B} \right)\!\!\right) =$$

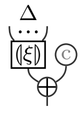


FIGURE 17. Translation of the Language

4. Semantics of the Many-Worlds Calculus

Our calculus represents linear operators between finite dimensional R-semimodules ($\mathbf{FdS}_R$), or equivalently R-weighted matrices. In particular, in the case $R = \mathbb{C}$ we use linear operators between finite dimensional Hilbert spaces, which correspond to pure quantum computations. More precisely, we will define two semantics, a world-dependent semantics $[\![-]\!]_a$ for every world $a \in W$, which will be a monoidal functor from $\mathbf{MW}_W$ to $\mathbf{FdS}_R$, and a world-agnostic semantics $[\![-]\!]$ from $\mathbf{MW}_\forall$ to $\mathbf{FdS}_R$.

4.1. Finite Dimensional R-Semimodules.

Similarly to $\mathbb{C}$-vector space, a R-semimodule is a set M in which one can compute R-weighted sums of elements on M. More precisely, we have $+ : M \times M \to M$ and $0 \in M$ forming a semigroup, and $\cdot : R \times M \to M$ which is associative, left-distributive and right-distributive. Morphisms between R-semimodules are expected to preserve R-weighted sums (including the trivial sum 0) that is:

$$f\left(\sum_{i=1}^{n} \lambda_i \cdot m_i\right) = \sum_{i=1}^{n} \lambda_i \cdot f(m_i)$$

As in the case of vector spaces, it is said finite dimensional if there exists a finite basis, that is a finite set such that every element of M can be uniquely decomposed as a R-weighted sum of that set. Relying on the uniqueness of this decomposition, a finite dimensional R-semimodule is actually isomorphic to R^n for some $n \geq 0$. Note that our definition of finite dimensionality forces the semimodule to be *freely* generated from a finite number of elements. Similarly to the vector space case, it is enough to define morphisms on a basis. For convenience, we will assume that R-semimodules come with a canonical basis, and for the remaining of this section we will write $\{m_1, \dots, m_m\}$ for the canonical basis of M and $\{n_1, \dots, n_n\}$ for the canonical basis of N. This canonical basis allows us to define an operation $\langle\ |\ \rangle : M \times M \to R$, alike to the inner product[10] of vector spaces. Writing $\{m_1, \dots, m_m\}$ for the canonical basis of M, we define:

$$\left\langle \sum_{i=1}^{m} \lambda_i \cdot m_i \,\middle|\, \sum_{i=1}^{m} \rho_i \cdot m_i \right\rangle := \sum_{i=1}^{m} \lambda_i \times \rho_i$$

While issues might arise in infinite dimensional cases, one can easily define the direct sum and the tensor product in our finite dimensional case. Indeed, we can define $M \oplus N$ and $M \otimes N$ as being the R-semimodules freely generated respectively by the pairs $(0, m_i)$ and $(1, n_j)$, and by the pairs (m_i, n_j) which we write $m_i \otimes n_j$, that is:

$$M \oplus N := \left\{ \sum_{1 \leq i \leq m} \lambda_i \cdot (0, m_i) + \sum_{1 \leq j \leq n} \rho_j \cdot (1, n_j) \,\middle|\, \lambda_i, \rho_j \in R \right\}$$

$$M \otimes N := \left\{ \sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} \lambda_{i,j} \cdot m_i \otimes n_j \,\middle|\, \lambda_{i,j} \in R \right\}$$

[10] Contrary to the inner-product of Hilbert space, this operation does not take the conjugate of its left-hand-side parameter.

Choosing a different basis for M or N yields an isomorphic R-semimodule, and this operation is associative and symmetric up to isomorphism. The $\otimes$ operation can be extended to a bilinear operation on elements of M and N as follows:

$$\left(\sum_{i=1}^{m} \lambda_i \cdot m_i\right) \otimes \left(\sum_{j=1}^{n} \rho_j \cdot n_j\right) := \sum_{i=1}^{m}\sum_{j=1}^{n} (\lambda_i \times \rho_j) \cdot m_i \otimes n_j$$

We write $\mathbf{FdS}_R(M, N)$ for morphisms between finite dimensional R-semimodules, and we note that it is itself a finite dimensional R-semimodule, meaning that we can compute R-weighted sums of morphisms. Indeed, the following is a basis of $\mathbf{FdS}_R(M, N)$:

$$\{m_i \mapsto n_j \text{ and } \forall k \neq i, m_k \mapsto 0\}_{\substack{1\leq i\leq m\\ 1\leq j\leq n}} = \left\{\sum_{k=1}^{m} \lambda_k \cdot m_k \mapsto \lambda_i \cdot n_j\right\}_{\substack{1\leq i\leq m\\ 1\leq j\leq n}}$$

This basis allows us to represent morphisms a $m \times n$ matrix with coefficients in R, and in fact composition of morphisms matches the usual matrix product. The operations $\cdot$, $+$ and $\otimes$ extend to morphisms, and correspond to multiplying every coefficient of the matrix by the same scalar, making a coefficient-by-coefficient sum, and making the Kronecker product of matrices. Taking inspiration from matrices, a transpose operation can be defined as follows. If $f \in \mathbf{FdS}_R(M, N)$ satisfies $f(m_i) = \sum_{j=1}^{n} \lambda_j^i n_j$, then $f^t \in \mathbf{FdS}_R(N, M)$ is defined as:

$$f^t\left(\sum_{j} \rho_j \cdot n_j\right) := \sum_{\substack{1\leq i\leq m\\ 1\leq j\leq n}} \rho_j \times \lambda_j^i \cdot m_i$$

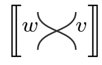
We also note that block matrices work as expected: a morphism $f \in \mathbf{FdS}_R(M_0 \oplus M_1, N_0 \oplus N_1)$ can be seen as four morphisms $f_{ij} \in \mathbf{FdS}_R(M_i, N_j)$ assembled together as $f = \begin{pmatrix} f_{00} & f_{10} \\ f_{01} & f_{11}\end{pmatrix}$, that is, writing m_k^i and n_k^j for the elements of the canonical basis of M_i and N_j respectively, and writing $f_{ij}(m_k^i) = \sum_\ell \lambda_\ell^{i,j,k} \cdot n_k^j$ then we have:

$$f\left(\sum_{i\in\{0,1\}}\sum_{k} \rho_k^i \cdot (i, m_k^i)\right) = \sum_{i,j\in\{0,1\}}\sum_{k,\ell} \rho_k^i \times \lambda_\ell^{i,j,k} \cdot (j, n_k^j)$$

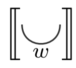
From the above properties it follows that $\mathbf{FdS}_R$ forms a category, with $\otimes$ and $\oplus$ being two symmetric monoidal products, $\otimes$ distributive over $\oplus$ and $\oplus$ being a biproduct.

4.2. **Semantics of Objects.** We start by defining the semantics of $\mathbf{MW}_W$ and $\mathbf{MW}_\forall$ on the objects. For every object $\mathcal{A}$ of $\mathbf{C_D}$, we define its *enablings* $\mathcal{A}^\bullet$ as "replacing any number of wire types by $\bullet$". For example $(A \parallel B)^\bullet = \{\bullet \parallel \bullet, \bullet \parallel B, A \parallel \bullet, A \parallel B\}$. Then, for any object $(\mathcal{A}, \ell_\mathcal{A})$ of $\mathbf{MW}_W$ and any world $a \in W$, we define $(\mathcal{A}, \ell_\mathcal{A}) \cap a$ to be the enabling of $\mathcal{A}$ 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 $(A : \{a\} \parallel B : \{b\}) \cap a = A \parallel \bullet$. To each enabling $\mathcal{E} \in \mathcal{A}^\bullet$ we associate an R-semimodule $\mathcal{M}_\mathcal{E}$ as follows:

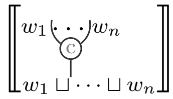
$$\mathcal{M}_{A_1\parallel\cdots\parallel A_n} := \mathcal{M}_{A_1} \otimes \cdots \otimes \mathcal{M}_{A_n}$$
$$\mathcal{M}_\varnothing := R \qquad \mathcal{M}_\bullet := R \qquad \mathcal{M}_\mathbb{1} := R \qquad \mathcal{M}_\mathbb{0} := \{0\}$$
$$\mathcal{M}_{A \otimes B} := \mathcal{M}_A \otimes \mathcal{M}_B \qquad \mathcal{M}_{A\oplus B} := \mathcal{M}_A \oplus \mathcal{M}_B$$



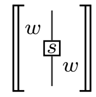
$$[\![\;\cdot\;]\!]_a = \begin{cases} \mathrm{Id} & \in \mathbf{FdS}_R(\mathcal{M}_A, \mathcal{M}_A) & \text{if } a \in w\backslash v \\ \mathrm{Id} & \in \mathbf{FdS}_R(\mathcal{M}_B, \mathcal{M}_B) & \text{if } a \in v\backslash w \\ h \otimes h' \mapsto h' \otimes h & \in \mathbf{FdS}_R(\mathcal{M}_{A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



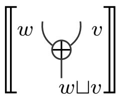
$$[\![\;\cdot\;]\!]_a = \begin{cases} h \otimes h' \mapsto \langle h | h' \rangle & \in \mathbf{FdS}_R(\mathcal{M}_{A \otimes A}, R) & \text{if } a \in w \\ (1) & \in \mathbf{FdS}_R(R, R) & \text{otherwise} \end{cases}$$



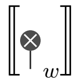
$$[\![\;\cdot\;]\!]_a = \begin{cases} \mathrm{Id} & \in \mathbf{FdS}_R(\mathcal{M}_A, \mathcal{M}_A) & \text{if } a \in \bigsqcup_i w_i \\ (1) & \in \mathbf{FdS}_R(R, R) & \text{otherwise} \end{cases}$$



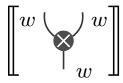
$$[\![\;\cdot\;]\!]_a = \begin{cases} s \cdot \mathrm{Id} & \in \mathbf{FdS}_R(\mathcal{M}_A, \mathcal{M}_A) & \text{if } a \in w \\ (1) & \in \mathbf{FdS}_R(R, R) & \text{otherwise} \end{cases}$$



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



$$[\![\;\cdot\;]\!]_a = \begin{cases} (1) & \in \mathbf{FdS}_R(\mathcal{M}_\varnothing, \mathcal{M}_{\mathbb{1}}) & \text{if } a \in w \\ (1) & \in \mathbf{FdS}_R(R, R) & \text{otherwise} \end{cases}$$



$$[\![\;\cdot\;]\!]_a = \begin{cases} \mathrm{Id} & \in \mathbf{FdS}_R(\mathcal{M}_A \otimes \mathcal{M}_B, \mathcal{M}_{A \otimes B}) & \text{if } a \in w \\ (1) & \in \mathbf{FdS}_R(R, R) & \text{otherwise} \end{cases}$$

Figure 18. Semantics of the Generators of $\mathbf{MW}_W$ in a World $a \in W$. Mirrored generators use the transposed for their semantics.

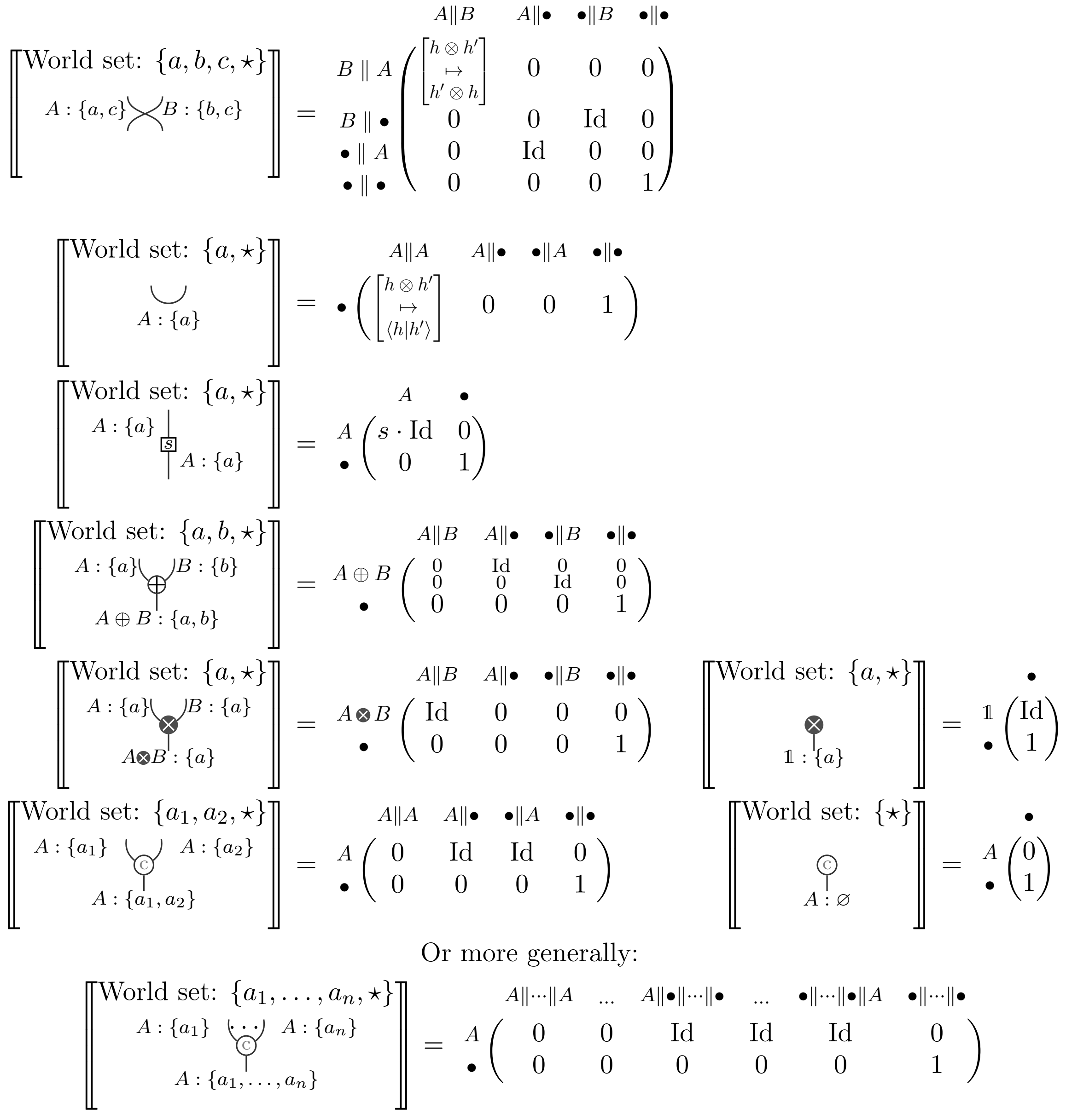
We can then define the semantics $[\![-]\!]_a : \mathbf{MW}_W \to \mathbf{FdS}_R$ and $[\![-]\!] : \mathbf{MW}_\forall \to \mathbf{FdS}_R$ on objects as $[\![(\mathcal{A}, \ell_\mathcal{A})]\!]_a := \mathcal{M}_{(\mathcal{A},\ell_\mathcal{A})\cap a}$ and $[\![\mathcal{A}]\!] := \bigoplus_{\mathcal{E}\in\mathcal{A}^\bullet} \mathcal{M}_\mathcal{E}$.

4.3. **World-Dependent Semantics.** 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 $[\![f \parallel g]\!]_a := [\![f]\!]_a \otimes [\![g]\!]_a$. The semantics of all the generators is given in Figure 18.

4.4. **World-Agnostic Semantics.** The world-agnostic semantics is defined from the world-dependent semantics, as follows. Consider f a morphism of $\mathbf{MW}_W((\mathcal{A}, \ell_\mathcal{A}), (\mathcal{B}, \ell_\mathcal{B}))$, we define its world-agnostic semantics $[\![f_W]\!] \in \mathbf{FdS}_R\left(\bigoplus_{\mathcal{E}\in\mathcal{A}^\bullet} \mathcal{M}_\mathcal{E}, \bigoplus_{\mathcal{F}\in\mathcal{B}^\bullet} \mathcal{M}_\mathcal{B}\right)$ as:

$$[\![f_W]\!] := \left\{ \sum_{\substack{a\in W \\ (\mathcal{A},\ell_\mathcal{A})\cap a = \mathcal{A}' \\ (\mathcal{B},\ell_\mathcal{B})\cap a = \mathcal{B}'}} [\![f]\!]_a \right\}_{\mathcal{A}'\in\mathcal{A}^\bullet, \mathcal{B}'\in\mathcal{B}^\bullet}$$

For example, the worlds-agnostic semantics of the generators (see Figure 19) are simply the collection of all their world-dependent semantics assembled into a single linear operator,



$$\begin{array}{c|cccc} & A\|B & A\|\bullet & \bullet\|B & \bullet\|\bullet \\ \hline B \parallel A & \begin{bmatrix} h\otimes h' \\ \mapsto \\ h'\otimes h \end{bmatrix} & 0 & 0 & 0 \\ B \parallel \bullet & 0 & 0 & \mathrm{Id} & 0 \\ \bullet \parallel A & 0 & \mathrm{Id} & 0 & 0 \\ \bullet \parallel \bullet & 0 & 0 & 0 & 1 \end{array}$$

$$\begin{array}{c|cccc} & A\|A & A\|\bullet & \bullet\|A & \bullet\|\bullet \\ \hline \bullet & \begin{bmatrix} h\otimes h' \\ \mapsto \\ \langle h|h'\rangle \end{bmatrix} & 0 & 0 & 1 \end{array}$$

$$\begin{array}{c|cc} & A & \bullet \\ \hline A & s\cdot\mathrm{Id} & 0 \\ \bullet & 0 & 1 \end{array}$$

$$\begin{array}{c|cccc} & A\|B & A\|\bullet & \bullet\|B & \bullet\|\bullet \\ \hline A\oplus B & \begin{smallmatrix}0\\0\end{smallmatrix} & \begin{smallmatrix}\mathrm{Id}\\0\end{smallmatrix} & \begin{smallmatrix}0\\\mathrm{Id}\end{smallmatrix} & \begin{smallmatrix}0\\0\end{smallmatrix} \\ \bullet & 0 & 0 & 0 & 1 \end{array}$$

$$\begin{array}{c|cccc} & A\|B & A\|\bullet & \bullet\|B & \bullet\|\bullet \\ \hline A\otimes B & \mathrm{Id} & 0 & 0 & 0 \\ \bullet & 0 & 0 & 0 & 1 \end{array} \qquad \begin{array}{c|c} & \bullet \\ \hline \mathbb{1} & \mathrm{Id} \\ \bullet & 1 \end{array}$$

$$\begin{array}{c|cccc} & A\|A & A\|\bullet & \bullet\|A & \bullet\|\bullet \\ \hline A & 0 & \mathrm{Id} & \mathrm{Id} & 0 \\ \bullet & 0 & 0 & 0 & 1 \end{array} \qquad \begin{array}{c|c} & \bullet \\ \hline A & 0 \\ \bullet & 1 \end{array}$$

Or more generally:

$$\begin{array}{c|cccccc} & A\|\cdots\|A & \dots & A\|\bullet\|\cdots\|\bullet & \dots & \bullet\|\cdots\|\bullet\|A & \bullet\|\cdots\|\bullet \\ \hline A & 0 & 0 & \mathrm{Id} & \mathrm{Id} & \mathrm{Id} & 0 \\ \bullet & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

FIGURE 19. World Agnostic Semantics of the Generators.
Mirrored generators use the transposed for their semantics.

as one can see by comparing Figure 18 to Figure 19. We first show that the worlds-agnostic semantics is functorial:

Proposition 4.1. *The worlds-agnostic semantics $[\![-]\!]$ defined in Section 4 is a monoidal functor from* $\mathbf{MW}_\forall$ *to* $\mathbf{FdS}_R$.

Proof. We recall here the definition of the worlds-agnostic semantics of $f_W : \mathcal{A} \to \mathcal{B}$:

$$\llbracket f_W \rrbracket := \left\{ \sum_{\substack{a \in W \\ (\mathcal{A}, \ell^f_{\mathcal{A}}) \cap a = \mathcal{A}' \\ (\mathcal{B}, \ell^f_{\mathcal{B}}) \cap a = \mathcal{B}'}} \llbracket f \rrbracket_a \right\}_{\mathcal{A}' \in \mathcal{A}^\bullet, \mathcal{B}' \in \mathcal{B}^\bullet}$$

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

$$\llbracket f_W \parallel g_V \rrbracket_{(a,b)} = \llbracket f_W \rrbracket_a \otimes \llbracket g_V \rrbracket_b$$
$$\text{and} \qquad \llbracket g_V \circ f_W \rrbracket_{(a,b)} = \llbracket g_V \rrbracket_b \circ \llbracket f_W \rrbracket_a$$

Remember that in the first case the set of worlds is $W \times V$, while in the second case, it is included in it. The functoriality with respect to the parallel composition is then immediate:

$$\llbracket f_W \parallel g_V \rrbracket = \left\{ \sum \llbracket f_W \parallel g_V \rrbracket_{(a,b)} \right\} = \left\{ \sum \llbracket f_W \rrbracket_a \right\} \otimes \left\{ \sum \llbracket g_V \rrbracket_b \right\} = \llbracket f_W \rrbracket \otimes \llbracket g_V \rrbracket$$

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:

$$(a, b) \notin w \iff (\mathcal{B}, \ell^f_{\mathcal{B}}) \cap a = (\mathcal{B}, \ell^g_{\mathcal{B}}) \cap b$$

where $f : (\mathcal{A}, \ell^f_{\mathcal{A}}) \to (\mathcal{B}, \ell^f_{\mathcal{B}})$ and $g : (\mathcal{B}, \ell^g_{\mathcal{C}}) \to (\mathcal{C}, \ell^g_{\mathcal{C}})$.

Then, we have

$$\llbracket g_V \circ f_W \rrbracket = \left\{ \sum \llbracket g_V \circ f_W \rrbracket_{(a,b)} \right\} = \left\{ \sum \llbracket g_V \rrbracket_b \right\} \circ \left\{ \sum \llbracket f_W \rrbracket_a \right\} = \llbracket g_V \rrbracket \circ \llbracket f_W \rrbracket \qquad \square$$

In order to show the universality of our language, we first start by defining an equational theory, that we show is sound, then define a notion of normal form. Furthermore, we will show that the normal form is unique, which is needed to prove the completeness of the language.

5. The Equational Theory

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

(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 $\llbracket - \rrbracket_a$ for every $a \in W$, hence sound for $\llbracket - \rrbracket$ too. We write $\equiv_W$ for the induced congruence[11] over $\mathbf{MW}_W$. We list those equations in Figure 20. For all but one of those equations that is restricted to the type $\mathbb{0}$, they can be applied for wires of any type. Quite notably, the last two rows describe the fact that the contraction is a natural transformation.

(2) Five additional equations with side effects on the set of worlds, which will be sound and complete for $\llbracket - \rrbracket$ (but not for $\llbracket - \rrbracket_a$). We write $\equiv$ for the induced equivalence relation, which is a congruence over $\mathbf{MW}_\forall$. 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 $(\mathcal{D}_f, \ell_f)_W \equiv (\mathcal{D}_f, i \circ \ell_f)_V$; Two equations allowing the annihilation (or creation, when looking at them from right to left) of worlds due to coproducts or scalars (first row

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

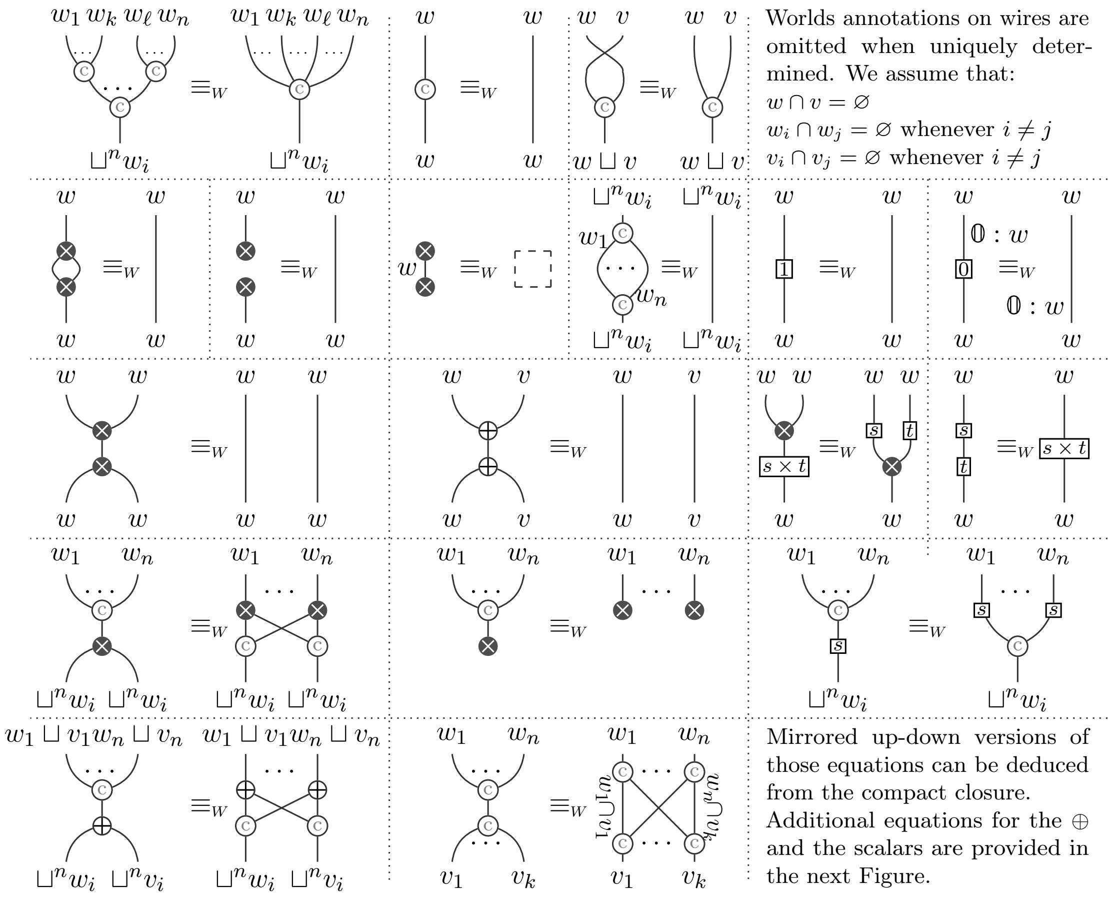
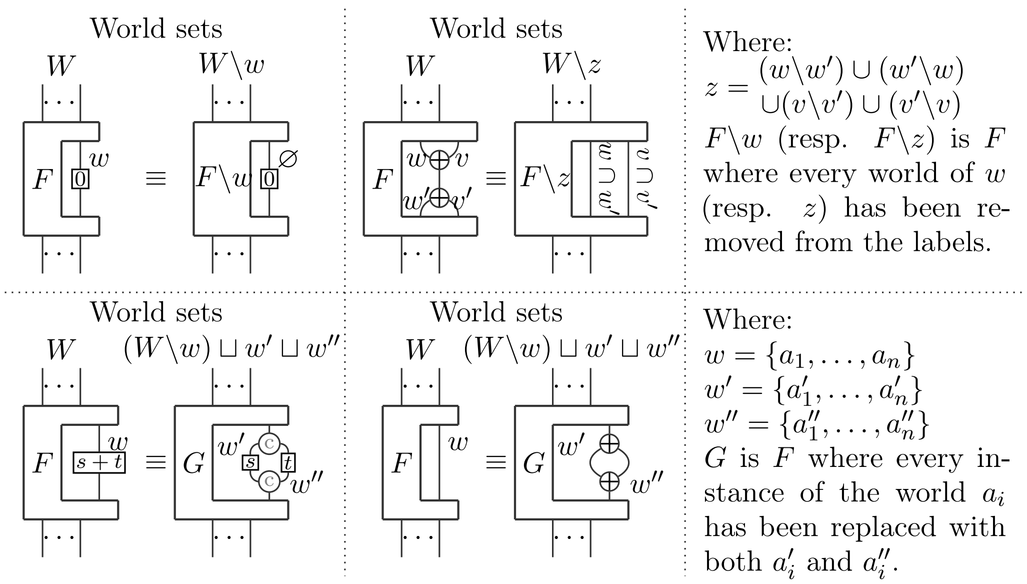

Worlds annotations on wires are omitted when uniquely determined. We assume that:
$w \cap v = \varnothing$
$w_i \cap w_j = \varnothing$ whenever $i \neq j$
$v_i \cap v_j = \varnothing$ whenever $i \neq j$
Mirrored up-down versions of those equations can be deduced from the compact closure. Additional equations for the $\oplus$ and the scalars are provided in the next Figure.

FIGURE 20. Equations with a Fixed World Set W

of Figure 21); 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 21).

Proposition 5.1 (Soundness). *For f a morphism of $\mathbf{MW}_W$ and g of $\mathbf{MW}_V$, whenever $f_W \equiv g_V$ we have $[\![f_W]\!] = [\![g_V]\!]$. Additionally if $W = V$, whenever $f \equiv_W g$ we have $\forall a \in W, [\![f]\!]_a = [\![g]\!]_a$.*

Proof. Given that most of the time, $[\![-]\!]_a$ 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 world 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 $\mathbf{FdS}_R$, 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 $[\![-]\!]$.



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

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

Figure 21. Equations with Side-Effects on World Sets

Splitting due to Scalars. Since $\mathbf{FdS}_R(R, S)$ is a R-semimodule, 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 $\mathbf{FdS}_R$ 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 $[\![-]\!]$. □

Example 5.2 (The Quantum Switch)**.** The Quantum Switch is a prime example of a process that makes use of quantum control flows. It assumes two operators U and V that can be applied as black boxes to a target system, and a control qubit that determines the order of application of these two operators. While the most direct implementation (e.g. using the language defined in Remark 2.5) makes two calls to the operators, one for each branch following the semantics:

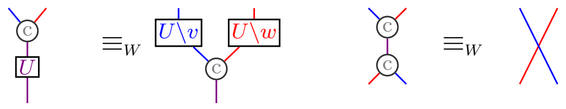
$$(\alpha\,|0\rangle + \beta\,|1\rangle) \otimes |\Psi\rangle \mapsto \alpha\,|0\rangle \otimes U \circ V\,|\Psi\rangle + \beta\,|1\rangle \otimes V \circ U\,|\Psi\rangle\,,$$

physical implementations exist that only require one occurrence of each of the two operators.

The leftmost diagram in Figure 22 represents, within the Many-Worlds calculus, the switch with a single occurrence of the operators, while the rightmost diagram uses the more direct, branch-wise representation. The equational theory offers the possibility to formally jump from one perspective to the other (as shown in the figure).

In those diagrams, the world set is $W = w \sqcup v$ and we rely on violet, blue and red wires to indicate respectively worlds labels $w \sqcup v$, w and v. Each figure has a control side which operates on a quantum bit (type $\mathbb{1} \oplus \mathbb{1}$) and binds the world w to $|0\rangle$ and the world v to $|1\rangle$; and a computational side which operates on some 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 step relies on the two following lemmas:



both of which being deducible from the equational theory (see Section 6). The second rewriting step is simply using the properties of a compact closed category.

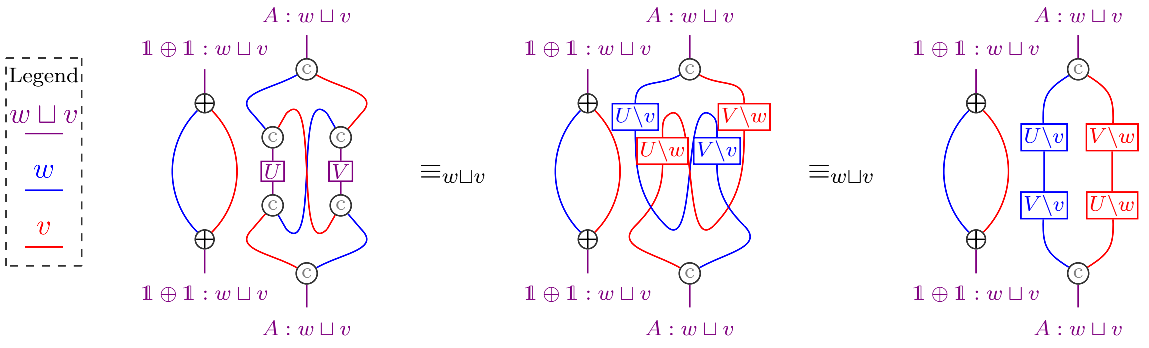


Figure 22. Rewriting the Quantum Switch

To emphasise the fact that each black box is queried once, the quantum switch is often presented as a higher-order operation where U and V appear as arguments. Using Remark 2.9 we can modify the leftmost diagram to represent the switch as a higher order operation, as shown in Figure 23.

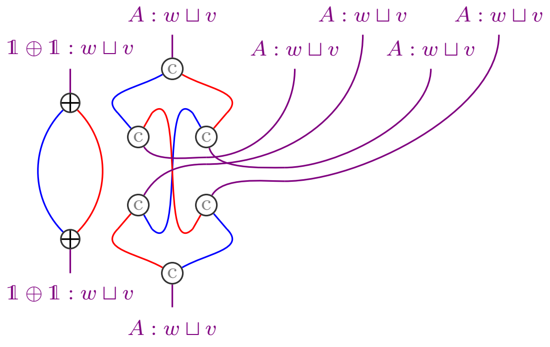


Figure 23. Higher Order Quantum Switch

6. Induced Equations

In Figure 20, we presented a set of equation reasonably small, by having equations parameterized by the arity of the contraction, and by omitting a lot of useful equations that can be deduced from them. In Figure 24, we take the opposite approach: we give equations using the contractions of arity zero and two (which are sufficient to generate all the other contractions) and we provide additional axioms that follows from Lemma 6.1 and Lemma 6.2.

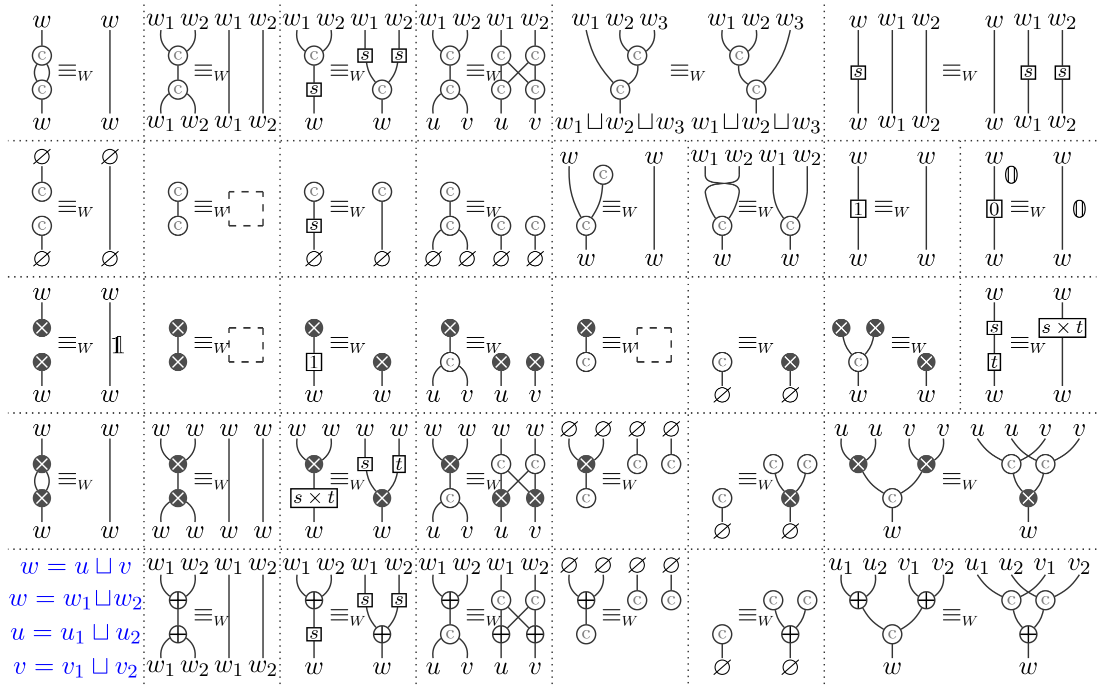
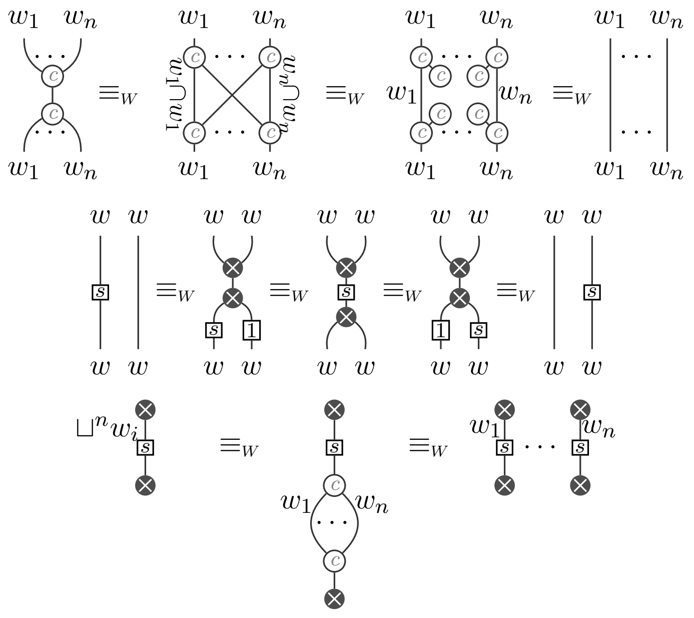


FIGURE 24. Alternative Presentation of the Equational Theory with a Fixed World Set W

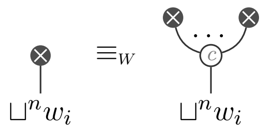
Note that this is only an alternative presentation to the equational theory for $\equiv_W$, the axioms of Figure 21 are still required for $\equiv$.

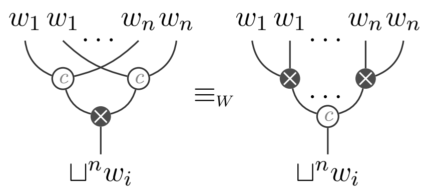
Lemma 6.1. *Whenever w_i are disjoints sets of worlds, we have the following:*

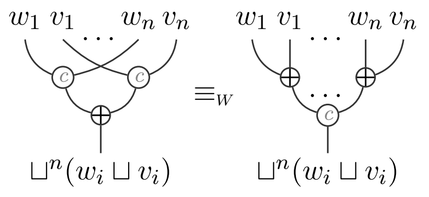


□

Lemma 6.2. *Whenever w_i and v_j are disjoint sets of worlds, we have the following:*

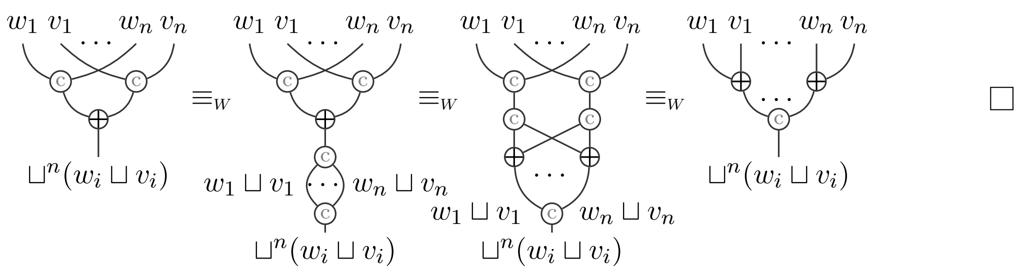
- 

- 

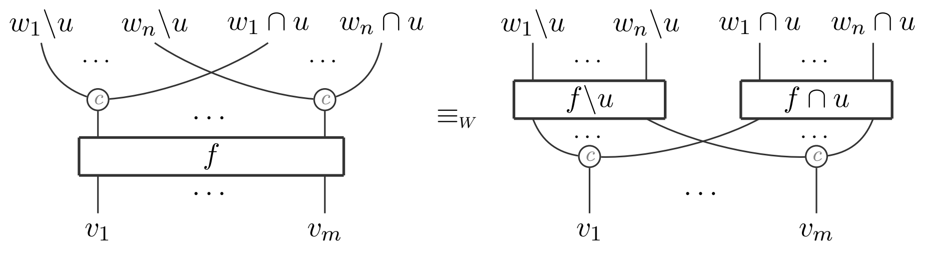
- 

$w_1\, v_1 \dots w_n\, v_n$
$\equiv_W$
$w_1\, v_1 \dots w_n\, v_n$
$\sqcup^n (w_i \sqcup v_i)$
$\sqcup^n (w_i \sqcup v_i)$

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



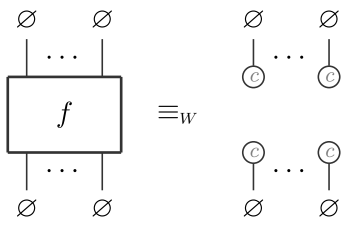
Corollary 6.3 (Naturality of the Binary Contraction)**.** *For every $f :\|^n (A_i : w_i) \to \|^m (B_j : v_j)$ with world set W and every $u \subseteq W$, we have*



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

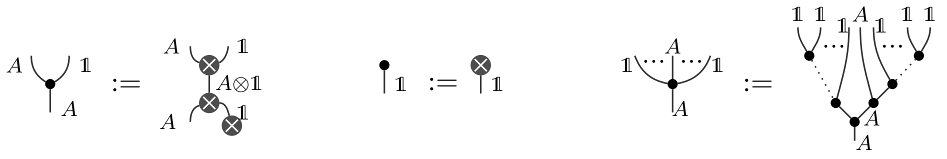
Proof. This is proven by induction over f. All the generator cases (including Cup and Cap) follow directly from the equations given in Figure 20 and Lemma 6.2 together with the properties of a compact close category. $\square$

Lemma 6.4 (Empty World)**.** *For every $f :\|^n (A_i : \varnothing) \to\|^m (B_j : \varnothing)$ with world set W but such that every worlds label of f is $\varnothing$, we have*

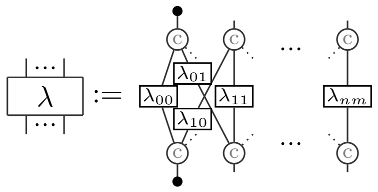


Proof. This is proven by replacing every wire by two contractions of arity zero (sixth axiom of Figure 20 with $n = 0$), and then using the naturality of the contraction of arity zero (last two lines of Figure 20 with $n = 0$) to consume every generator. □

6.1. **Normal Form.** For the remaining of this section, we use $\equiv$ instead of $\equiv_W$, as we will occasionally use equations from Figure 21. We are now ready to define the normal form of our diagrams, for that, it will be practical to define the following syntactic sugar, which we call the *unitor*, its *unit* and its *generalized form*:

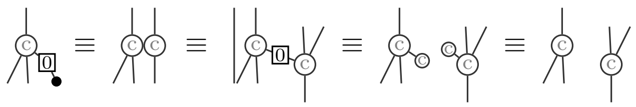


We define the following short-hand:

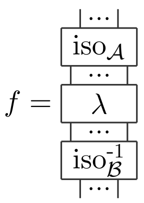


with the assumption that the world set is in bijection with the set of scalars, in other words the scalars λ_{ij} have for worlds label $\{a_{ij}\}$. In particular, all the input (resp. output) wires live in mutually exclusive worlds.

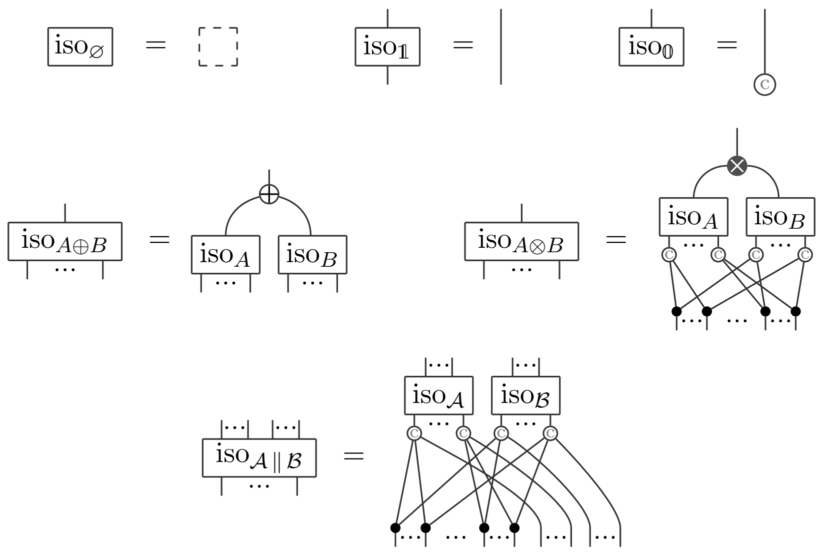
An important observation is that any permutation of wires (all mutually exclusive, and of type $\mathbb{1}$) can easily be put in this form using the following equations:



Definition 6.5 (Normal Form)**.** The normal form of a morphism $f : \mathcal{A} \to \mathcal{B}$ is defined as follows:



where the morphisms $\mathrm{iso}_{\mathcal{A}}$ and $\mathrm{iso}_{\mathcal{B}}^{-1}$ are defined inductively as:



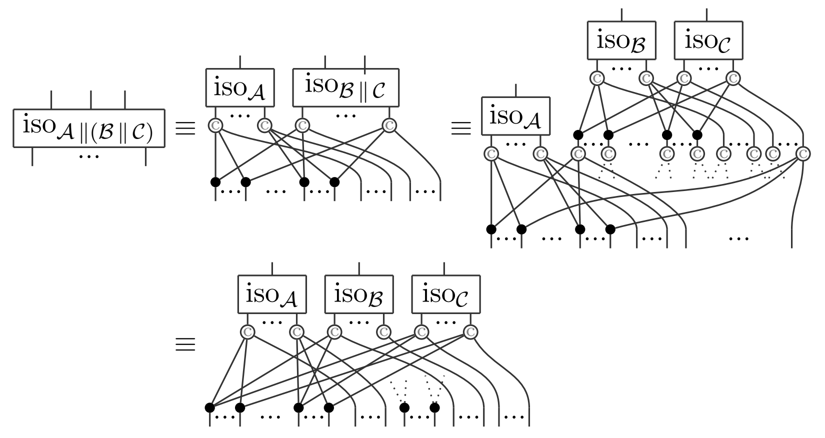
$\mathrm{iso}_{\varnothing}$ =
$\mathrm{iso}_{\mathbb{1}}$ =
$\mathrm{iso}_{\mathbb{0}}$ =
c
$\mathrm{iso}_{A\oplus B}$ = iso_A iso_B
$\mathrm{iso}_{A\otimes B}$ = iso_A iso_B
$\mathrm{iso}_{\mathcal{A}\,\|\,\mathcal{B}}$ = $\mathrm{iso}_{\mathcal{A}}$ $\mathrm{iso}_{\mathcal{B}}$

The output wires of $\mathrm{iso}_{\mathcal{A}}$ for any $\mathcal{A}$ 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 $\mathcal{A} \parallel (\mathcal{B} \parallel \mathcal{C})$ and $(\mathcal{A} \parallel \mathcal{B}) \parallel \mathcal{C}$, in other words $\parallel$ is strictly associative. However $\mathrm{iso}_{\mathcal{A}\|(\mathcal{B}\|\mathcal{C})}$ and $\mathrm{iso}_{(\mathcal{A}\|\mathcal{B})\|\mathcal{C}}$ are different, but they are equivalent up to a rearranging of the output wires:

Lemma 6.6. *There exists a wire permutation σ such that* $\mathrm{iso}_{\mathcal{A}\|(\mathcal{B}\|\mathcal{C})} = \sigma \circ \mathrm{iso}_{(\mathcal{A}\|\mathcal{B})\|\mathcal{C}}$. □

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

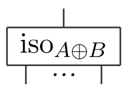


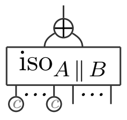
$\mathrm{iso}_{\mathcal{A}\,\|(\mathcal{B}\,\|\,\mathcal{C})}$ ≡ $\mathrm{iso}_{\mathcal{A}}$ $\mathrm{iso}_{\mathcal{B}\,\|\,\mathcal{C}}$ ≡ $\mathrm{iso}_{\mathcal{A}}$ $\mathrm{iso}_{\mathcal{B}}$ $\mathrm{iso}_{\mathcal{C}}$
≡ $\mathrm{iso}_{\mathcal{A}}$ $\mathrm{iso}_{\mathcal{B}}$ $\mathrm{iso}_{\mathcal{C}}$

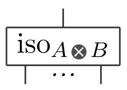
and similarly for $\mathrm{iso}_{(A\|B)\|C}$. It then suffices to check which contractions the bottom unitors are linked to. Naming the i-th contraction exiting iso_A as a_i, and similarly for B and C, we can see that for each triple (a_i, b_j, c_k) there is exactly one unitor connected to precisely 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. □

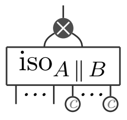
We hence have a choice to make here for canonicity, and choose $\mathrm{iso}_{A_0\|A_1\|A_2\|\ldots} := \mathrm{iso}_{(\ldots((A_0\|A_1)\|A_2)\|\ldots)}$.

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

Lemma 6.7. *Notice that* 

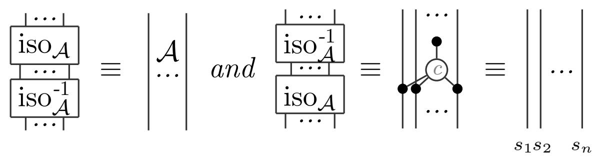
$\equiv$ 

and 

$\equiv$ 

.

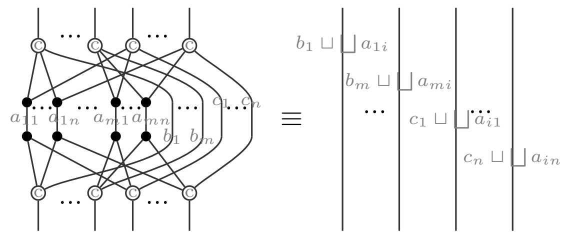
Lemma 6.8. *We have the following identities:*



$\mathrm{iso}_{\mathcal{A}}$
$\mathrm{iso}^{-1}_{\mathcal{A}}$
$\equiv$
$\mathcal{A}$
and
$\mathrm{iso}^{-1}_{\mathcal{A}}$
$\mathrm{iso}_{\mathcal{A}}$
$\equiv$
$\equiv$
$s_1 s_2$
s_n

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

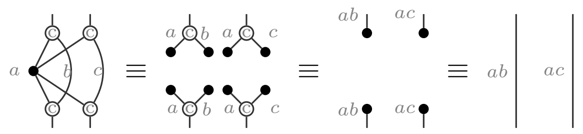
Proof. We will use the following identities:

 (6.1)

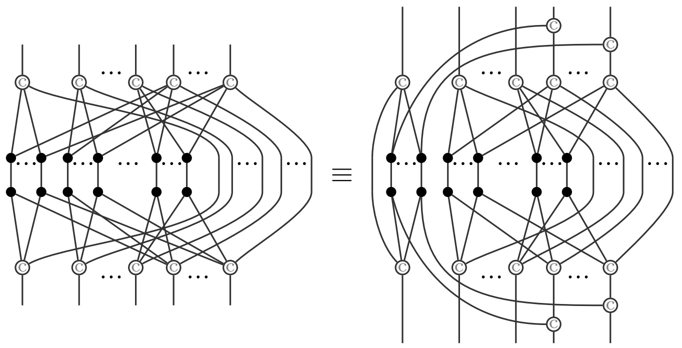
a_{11} a_{1n} a_{m1} a_{mn}
b_1 b_m
c_1 c_n
$\equiv$
$b_1 \sqcup \bigsqcup a_{1i}$
$b_m \sqcup \bigsqcup a_{mi}$
$c_1 \sqcup \bigsqcup a_{i1}$
$c_n \sqcup \bigsqcup a_{in}$

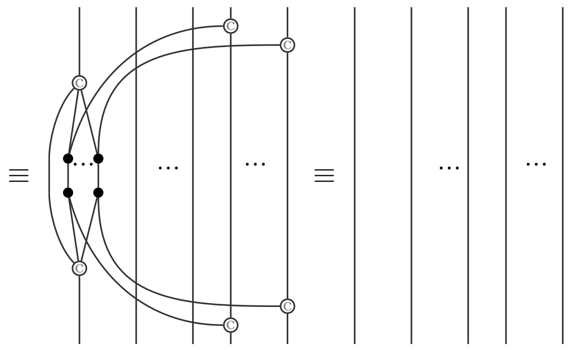
when all the wires a_{ij}, b_i, c_j are mutually exclusive.

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

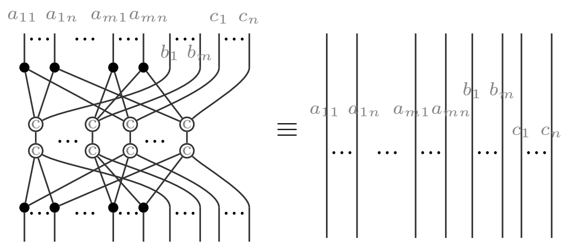


For any n and m, we can then prove the case $(n + 1, m)$ using the cases (n, m) and $(1, m)$ (the case $(n, m + 1)$ is completely symmetric):

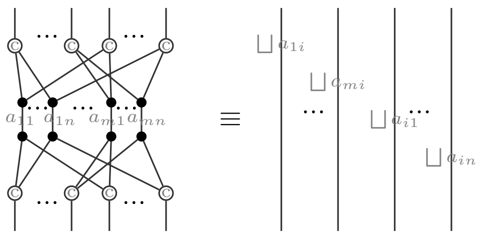




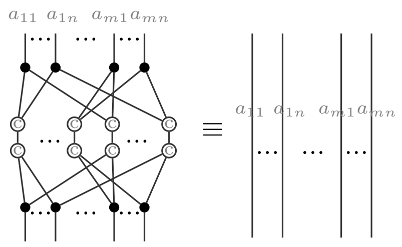
In a similar way, it is possible to show the following three identities:



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

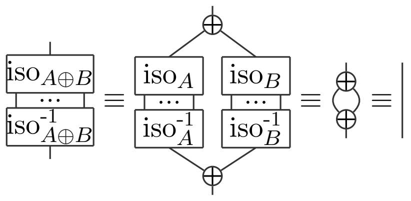


when all the wires a_{ij}.



when all the wires a_{ij} are mutually exclusive.

- [iso$^{-1}_{\mathcal{A}}$ ∘ iso$_{\mathcal{A}}$]: The result is obvious in cases $\mathbb{1}$, $\mathbb{0}$ and $\varnothing$. For $A \oplus B$:



  The proof is similar for ⊗ and ∥ using the previous identities.

- [iso$_{\mathcal{A}}$ ∘ iso$^{-1}_{\mathcal{A}}$]: The result is again obvious for $\mathbb{1}$, $\mathbb{0}$ and $\varnothing$. The general result is easy to prove by induction using the above identities. □

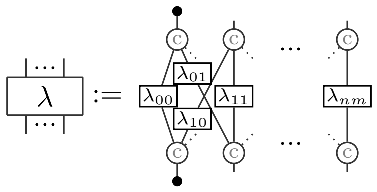
6.2. **Universality.** We are now ready to show the universality of the language:

Theorem 6.9 (Universality).
For every objects $\mathcal{A}, \mathcal{B}$ of $\mathbf{C_D}$, and for every linear operator

$$\Lambda \in \mathbf{FdS}_R\left(\bigoplus_{\mathcal{E}\in\mathcal{A}^\bullet}\mathcal{H}_\mathcal{E}, \bigoplus_{\mathcal{F}\in\mathcal{B}^\bullet}\mathcal{H}_\mathcal{B}\right)$$
$$\Lambda = \begin{pmatrix} \lambda_{1\,1} & \cdots & \lambda_{n\,1} & \lambda_{10} \\ \vdots & & \vdots & \vdots \\ \lambda_{1\,m} & \cdots & \lambda_{n\,m} & \lambda_{m0} \\ \lambda_{01} & \cdots & \lambda_{0n} & \lambda_{00} \end{pmatrix}$$

there exists a set of worlds W, and a morphism $f_W \in \mathbf{MW}_\forall(\mathcal{A}, \mathcal{B})$ such that its worlds-agnostic semantics $[\![f]\!]$ is equal to U.

Proof. We take $f = \mathrm{iso}_\mathcal{B}^{-1} \circ \lambda \circ \mathrm{iso}_\mathcal{A}$ as defined above and show that $[\![f]\!] = \Lambda$. As stated in Section 6.1, there is one world for each scalar in λ, so we write $a_{i\,j}$ the world associated to $\lambda_{i\,j}$ and W the set of all those worlds. Let us write $\mathbb{1}_0^k = \bullet^k$ for $\bullet \parallel \cdots \parallel \bullet$ (with k elements) and $\mathbb{1}_i^k$ for $\bullet^k$ where the i-th $\bullet$ has been replaced by $\mathbb{1}$ if $1 \le i \le k$. Beware that we consider $\mathbb{1}_0^k$ to be the *last* element of the canonical basis. We then have $[\![\lambda]\!]_{a_{i\,j}} : \mathbb{1}_i^n \to \mathbb{1}_j^m : x \mapsto \lambda_{i\,j} \cdot x$ where:



Additionally, one can show by induction that $[\![\mathrm{iso}_\mathcal{A}]\!]_{a_{i\,j}}$ is simply the projection on the i-th element of the canonical basis, and $[\![\mathrm{iso}_\mathcal{B}^{-1}]\!]_{a_{i\,j}}$ is simply the injection on the j-th element of the canonical basis. Since we have $[\![f]\!]_a = [\![\mathrm{iso}_\mathcal{B}^{-1}]\!]_a \circ [\![\lambda]\!]_a \circ [\![\mathrm{iso}_\mathcal{A}]\!]_a$ for every $a \in W$, and $[\![f]\!]$ being the collection of all the $[\![f]\!]$, we obtain that $[\![f]\!] = \Lambda$. □

6.3. **Uniqueness of the Normal Form.** A crucial feature of the normal form for the completeness is its uniqueness:

Proposition 6.10. *The normal form is unique.*

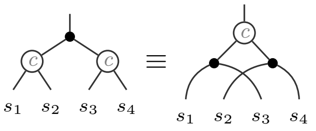
Proof. Let f and g be two diagrams in normal form (with respectively λ and μ as inner block), such that $[\![f]\!] = [\![g]\!]$ (the naming of the worlds is taken to be the same in both diagrams, and is the same as in the previous proof). By the definition of $[\![.]\!]$, we have $[\![f]\!]_a = [\![g]\!]_a$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

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

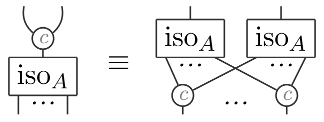
$$\begin{aligned} \lambda_{ij} &= e_j^{\mathcal{B}\dagger}\, [\![\mathrm{iso}_\mathcal{B}^{-1}]\!]_{a_{ij}} \circ [\![\lambda]\!]_{a_{ij}} \circ [\![\mathrm{iso}_\mathcal{A}]\!]_{a_{ij}}\, e_i^\mathcal{A} \\ &= e_j^{\mathcal{B}\dagger}\, [\![\mathrm{iso}_\mathcal{B}^{-1}]\!]_{a_{ij}} \circ [\![\mu]\!]_{a_{ij}} \circ [\![\mathrm{iso}_\mathcal{A}]\!]_{a_{ij}}\, e_i^\mathcal{A} = \mu_{ij} \end{aligned}$$

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. □

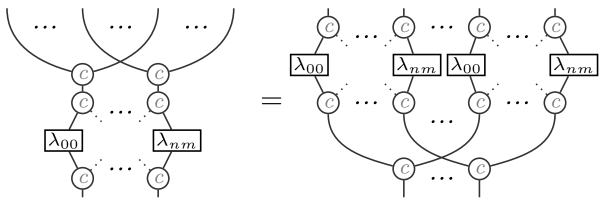
6.4. **Completeness.** We can now use this normal form to show that our equational theory is complete for arbitrary morphisms. To do so, we need to show that all the generators can be put in normal form, and then that any composition of morphisms in normal form can be put in normal form. To do so we will first derive a few lemmas:

Corollary 6.11 (of Lemma 6.3)**.**  *when* $s_1 \cap s_4 = s_2 \cap s_3 = \varnothing$

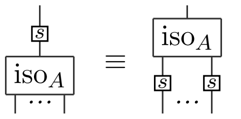
Corollary 6.12 (of Lemma 6.3)**.** *Single-colored isos distribute over the contraction:*



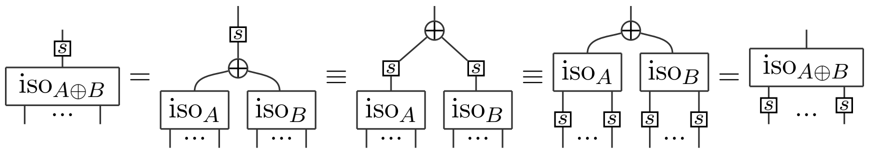
Corollary 6.13 (of Lemma 6.3)**.**



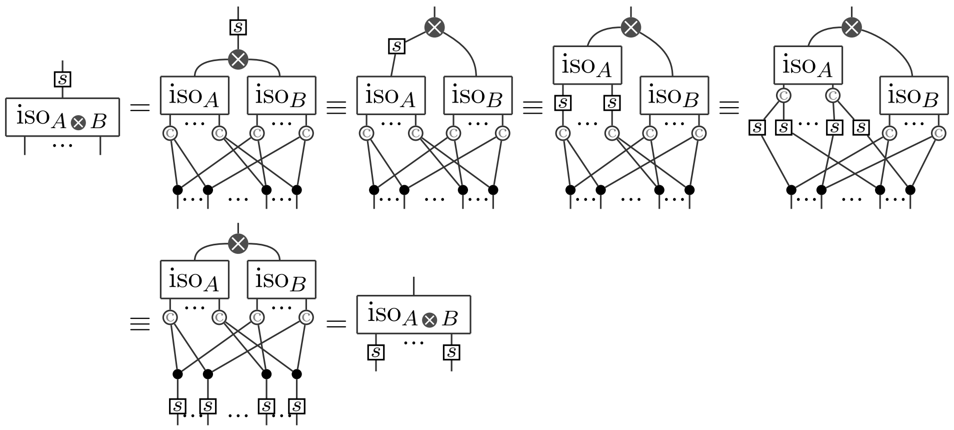
Lemma 6.14. *Scalars distribute over single-colored isos:*



Proof. The result is obvious for $\mathbb{1}$ and $\varnothing$. For $A \oplus B$:

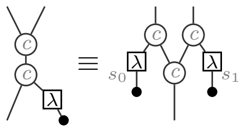


For $A \otimes B$:



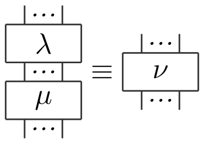
$\square$

Corollary 6.15 (of Lemma 6.3)**.**



with $s_0 \cap s_1 = \varnothing$*.*

Lemma 6.16.



with $\nu_{ij} = \sum_k \lambda_{ik}\mu_{kj}$*.*

Proof. Suppose the worlds of λ are the $\{a_{ij}\}_{ij}$ and that of μ are the $\{b_{k\ell}\}_{k\ell}$. We count inputs/outputs starting at 1, hence $p \geq 1$ in the following.

The p-th output of λ has world set $\{a_{ip}\}_i$ and the p-th input of μ has world set $\{b_{p\ell}\}_\ell$. When composing the two in sequence, in the first step, each singleton world $\{a_{ij}\}$ (resp. $\{b_{k\ell}\}$) is mapped to $\{(a_{ij}, b_{k\ell})\}_{k\ell}$ (resp. $\{(a_{ij}, b_{k\ell})\}_{ij}$), and unions of singleton worlds to unions of the worlds each is mapped to.

In particular, the p-th output of λ now has world set $\{(a_{ip}, b_{k\ell})\}_{ik\ell}$, and the p-th input of μ now has world set $\{(a_{ij}, b_{p\ell})\}_{ij\ell}$. After composition, the two sets have to match. They become $\{(a_{ip}, b_{p\ell})\}_{i\ell}$, and all pairs of world that were removed from these two sets are removed globally.

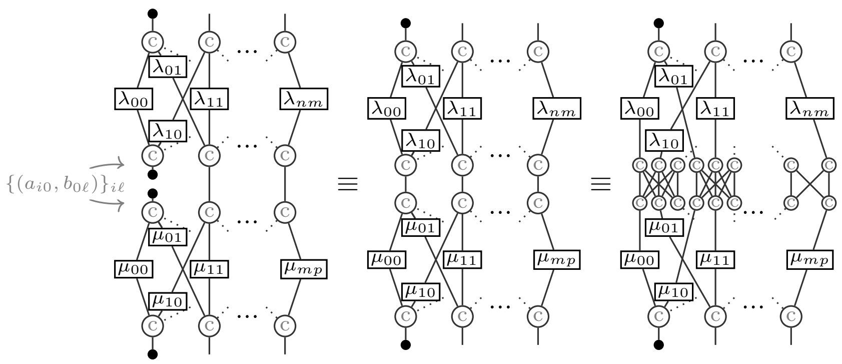
We do so for all $p \geq 1$, which means all worlds in:

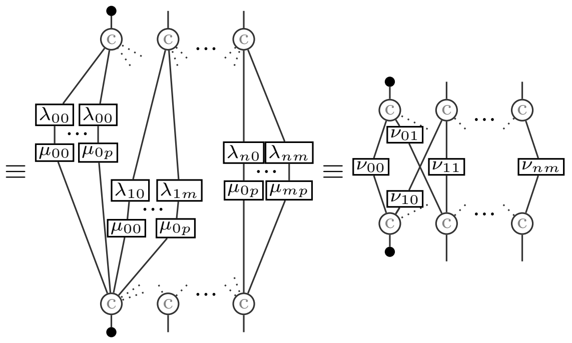
$$\begin{aligned}\{(a_{ij}, b_{k\ell}) \mid j \geq 1, j \neq k\} \cup \{(a_{ij}, b_{k\ell}) \mid k \geq 1, j \neq k\}\\ = \{(a_{ij}, b_{k\ell}) \mid j \neq k\}\end{aligned}$$

are removed. Crucially, this means that the world sets $\{a_{i0}\}_i$ and $\{b_{0\ell}\}_\ell$ are mapped to the same world set:

$$\begin{aligned}\{a_{i0}\}_i \mapsto \{(a_{i0}, b_{k\ell})\}_{ik\ell} \setminus \{(a_{ij}, b_{k\ell}) \mid j \neq k\} &= \{(a_{i0}, b_{0\ell})\}_{i\ell}\\ &= \{(a_{ij}, b_{0\ell})\}_{ij\ell} \setminus \{(a_{ij}, b_{k\ell}) \mid j \neq k\} \mapsfrom \{b_{0\ell}\}_\ell\end{aligned}$$

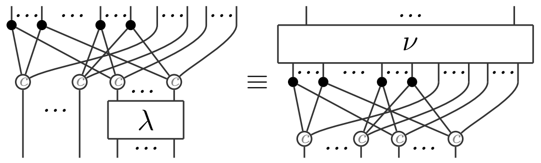
Hence, the sequential composition of λ and μ becomes:



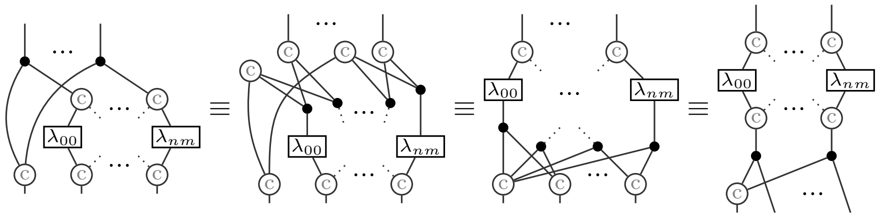


with $\nu_{ij} = \sum_k \lambda_{ik}\mu_{kj}$. □

Lemma 6.17. *For any "matrix block" λ, there exists a "matrix block" ν such that:*

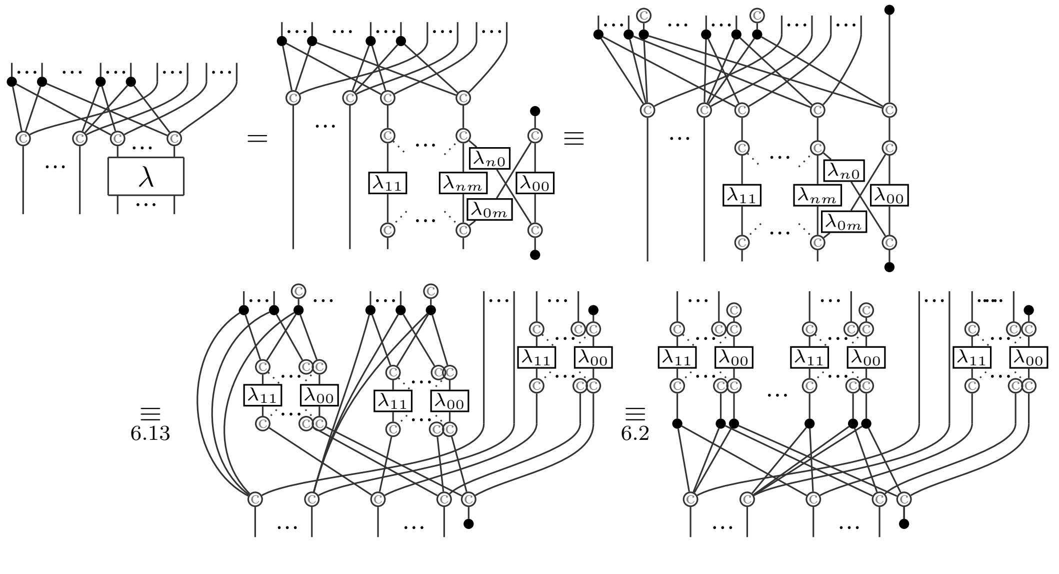


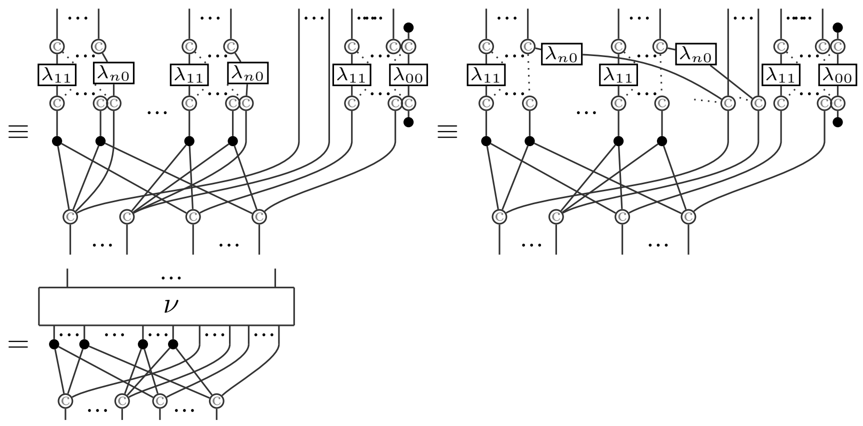
Proof. First, notice the following:



(6.2)

We may then prove the lemma:

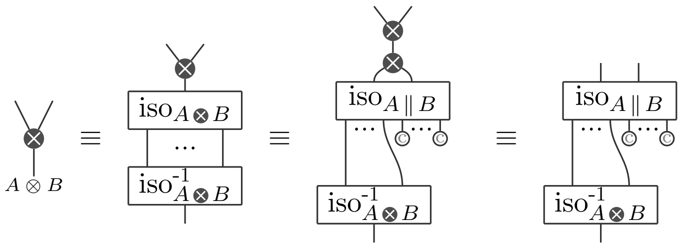



□

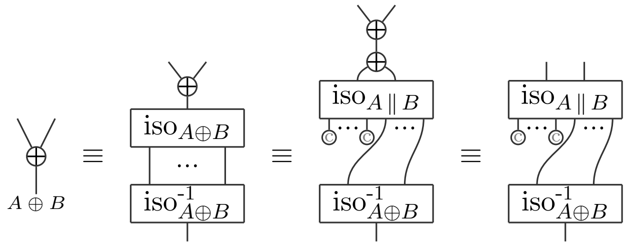
We can now move on to show that generators can be put in normal form:

Proposition 6.18. *The generators can be put in normal form.*

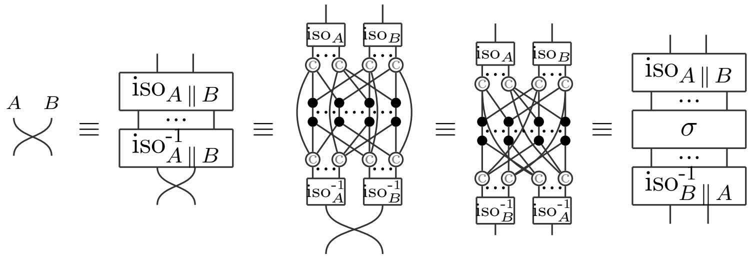
Proof. First for the tensor:



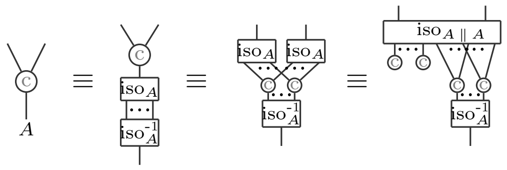
Then, for the plus:



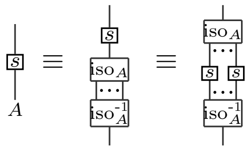
The swap:



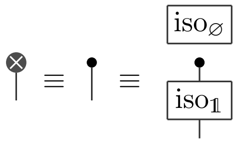
with σ a simple permutation of wires. For the case of the contraction, the unary case is dealt with thanks to Lemma 6.4. The binary case is dealt with:



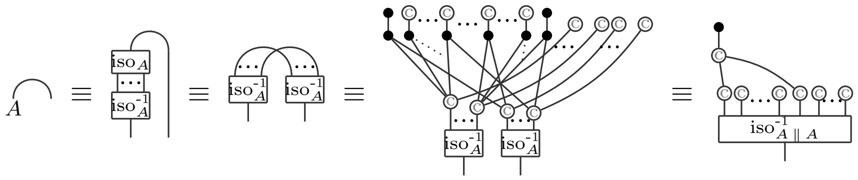
For the general, n-ary case of the contraction, it suffices to decompose any contraction into binary contractions, and use the above equality.



This normal form encompasses that of the identity, by simply taking $s = 1$. The unit is simply obtained as:



Finally, the cap is obtained as follows:

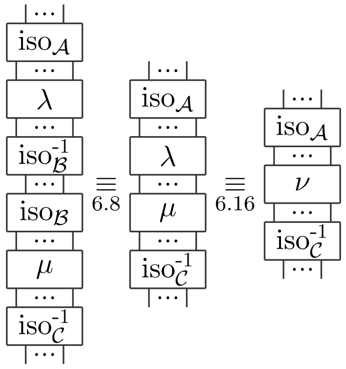


The upside-down versions of the generators are provided in exactly the same way (but upside-down). □

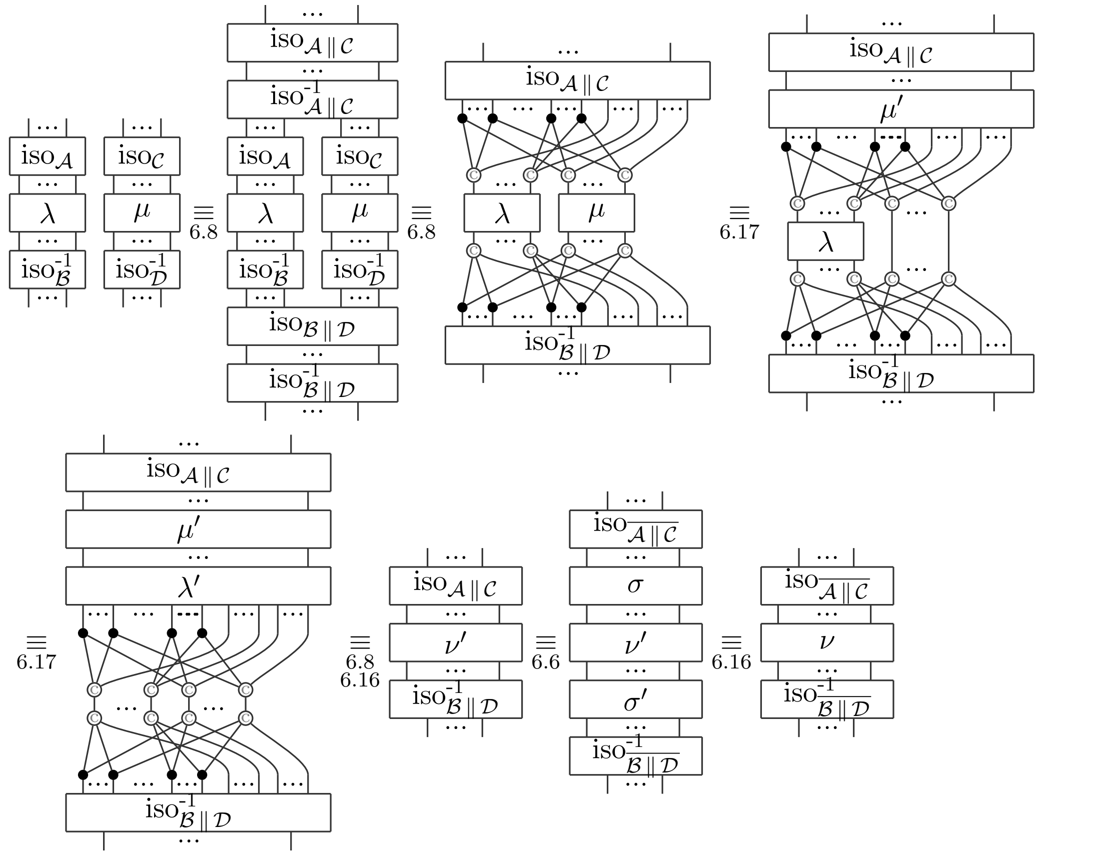
Proposition 6.19. *Compositions of diagrams in normal form can be put in normal form.*

It is then possible to show that compositions of diagrams in normal form can be put in normal form:

Proof. In the case of sequential composition:



In the case of parallel composition:



iso$_{\mathcal{A}}$
iso$_{\mathcal{C}}$
λ
μ
iso$^{-1}_{\mathcal{B}}$
iso$^{-1}_{\mathcal{D}}$
$\equiv_{6.8}$
iso$_{\mathcal{A}\parallel\mathcal{C}}$
iso$^{-1}_{\mathcal{A}\parallel\mathcal{C}}$
iso$_{\mathcal{B}\parallel\mathcal{D}}$
iso$^{-1}_{\mathcal{B}\parallel\mathcal{D}}$
$\equiv_{6.8}$
$\equiv_{6.17}$
μ'
λ'
$\equiv_{6.17}$
$\equiv_{6.8, 6.16}$
ν'
$\equiv_{6.6}$
iso$_{\overline{\mathcal{A}\parallel\mathcal{C}}}$
σ
σ'
iso$^{-1}_{\overline{\mathcal{B}\parallel\mathcal{D}}}$
$\equiv_{6.16}$
ν

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

We are now ready to show the completeness of the Many-Worlds Calculus:

Theorem 6.20 (Completeness)**.** *For every $f_W : \mathcal{A} \to \mathcal{B}$ and $g_V : \mathcal{A} \to \mathcal{B}$, $[\![f_W]\!] = [\![g_V]\!] \iff f_W \equiv g_V$.*

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

Let f_1 and f_2 be two morphisms such that $[\![f_1]\!] = [\![f_2]\!]$. Both morphisms can be put in normal form, resp. f_1^{NF} and f_2^{NF}, with $f_i \equiv f_i^{NF}$ and thus $[\![f_i^{NF}]\!] = [\![f_i]\!]$. By uniqueness of the normal form, and since $[\![f_1^{NF}]\!] = [\![f_2^{NF}]\!]$, we get $f_1^{NF} \equiv f_2^{NF}$, which ends the proof that $f_1 \equiv f_2$. □

7. Comparison with Other Graphical Languages

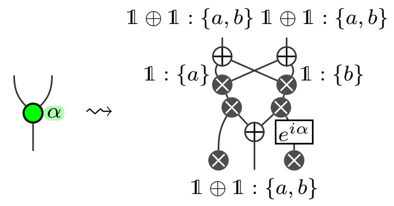
The distinctive feature of the Many-Worlds Calculus is that it graphically puts the tensor and the biproduct on an equal footing. By comparison, other graphical language for quantum computing are inherently centered around either one of them. The ZX-calculus [CD11] and cousin languages ZW- and ZH-Calculi [BK19, Had15], as well as Duncan's Tensor-Sum Logic [Dun09], use the tensor product as the default monoid, while more recent language –

particularly for linear optics [CP20, dFC23, CHM$^+$22] – use the biproduct. We have a closer look at each of them in the following, and show how – at least part of – each language can be encoded naturally in the Many-Worlds Calculus. Most of them comes equipped with an equational theory. By completeness of our language (Theorem 6.20), all the equations expressible in the fragments we consider can be derived in our framework.

7.1. **ZX-Calculus.**

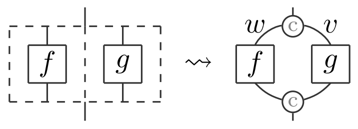
The first difference is the restrictions of the ZX-calculus to computations between qubits, in other words linear map from $\mathbb{C}^{2^n} \mapsto \mathbb{C}^{2^m}$, while our language can encode any linear map from $\mathbb{C}^n \mapsto \mathbb{C}^m$. The Tensor generator allowing the decomposition of $\mathbb{C}^{2^n}$ into instances of $\mathbb{C}^2$ was already present in the *scalable* extension of the ZX-calculus [CHP19], but the main difference comes from the Plus (and the Contraction).

Additionally, every ZX-diagram can be encoded in our graphical language. The identity, swap, cup and cap of the ZX calculus are encoded by the similar generators over the type $\mathbb{1} \oplus \mathbb{1}$, the Hadamard gate is encoded as in Figure 4, and the green spider is encoded as shown below. An encoding for the red spider can then be deduced from those. Diagrams are composed together with the world-agnostic composition of $\mathbf{MW}_\forall$.


α
$\rightsquigarrow$
$\mathbb{1} \oplus \mathbb{1} : \{a, b\}$
$\mathbb{1} \oplus \mathbb{1} : \{a, b\}$
$\mathbb{1} : \{a\}$
$\mathbb{1} : \{b\}$
$e^{i\alpha}$
$\mathbb{1} \oplus \mathbb{1} : \{a, b\}$

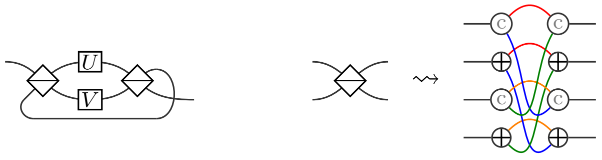
7.2. **Tensor-Sum Logic.**

The core difference between the Tensor-Sum Logic [Dun09] 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 below 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 superpositions 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.



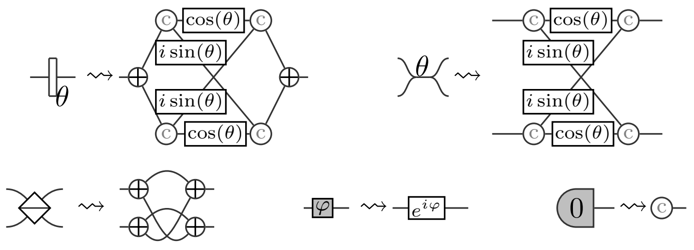
7.3. **PBS-Calculus.**

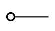
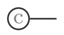
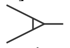
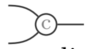
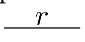
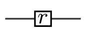
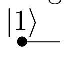
The PBS-Calculus [CP20] allows one to represent coherent quantum control by the use of *polarizing beam splitters* (pbs): whenever a qubit enters a pbs node, depending on the polarity of the qubit it will either go through or be reflected. By making implicit the target system, controlled by the optical system represented by the diagram, the PBS-Calculus allows one to encode the Quantum Switch (depicted on the left). The pbs generator is related to the $\oplus$ of the Many-Worlds.



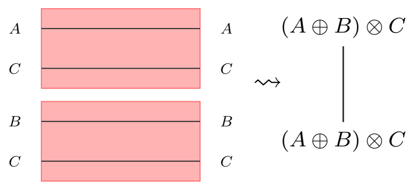
The first main difference with our language is that, since the generators of the PBS-Calculus represent physical components, any PBS-diagram is by construction physical, while our approch is more atomic and decomposes physical components into abstract smaller ones. The second main difference lie in the trace: while they can allow a particle to pass through a wire at most twice, in our system, each wire can be used at most once: more formally, their trace is based on the coproduct while ours is on the tensor product. *If* we are assured that each wire can only be used once during the computation, any PBS-diagram can be translated to the Many-Worlds calculus directly, with the transformation on the right, where we distinguish the control system (the part of the diagram connected to ⊕s) from the target system (connected to ©s) which is implicit in the PBS-Calculus. Finally, as we mentionned in Section 1, whereas the PBS-calculus uses black boxes, there is no need for them in the Many-Worlds as they can directly be encoded in our system. Also, the distinction between the control and controlled system is no longer present, as they can change during the computation.

7.4. **LOv-Calculus.** In the PBS-Calculus, the qubit in the control system (the one explicitly represented) cannot be put in arbitrary superpositions of $|0\rangle$ and $|1\rangle$ *during* the computation. To allow this feature, we may add some linear optical components to the language's generators, and end up with the LOv-Calculus [CHM+22]. In this language, there is no trace and there is a unique photon traveling the circuit, which relieves us of the previous constraint. There is also no need for an implicit target system anymore. All wires at the interface between the generators are of type $\mathbb{1} \oplus \mathbb{1}$, and parallel wires have disjoint sets of worlds. Each generator can then be interpreted as follows:



7.5. **Path-Calculus.** The Path-Calculus is another recent graphical language for linear optical circuits [dFC23]. Its generators correspond directly to a subset of the Many-Worlds' with  $\rightsquigarrow$  ,  $\rightsquigarrow$  and  $\rightsquigarrow$ ; where each wire has type $\mathbb{1}$ and where parallel wires are on disjoint sets of worlds. This language is then used as the core for a more expressive language called QPath, which this time cannot be directly encoded in our language, except when restricting the set of generators (specifically to $n = 1$), in which case  $\rightsquigarrow$.

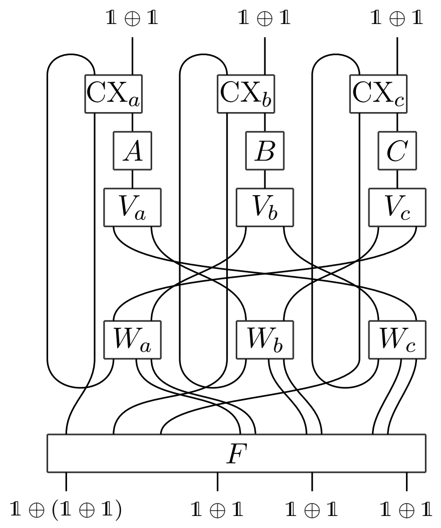
7.6. **Tapes diagrams.** In [BDGS23], *tape diagrams* are introduced to display both tensor product and biproduct, and a sound and complete equational theory is presented. There, systems that are in tensors are bundled in "tapes", and these tapes can then be put side-by-side, representing the coproduct of the two tensors. This framework turns out to be less verbose, since no world annotation is needed, and the distinction between tensor and coproduct is completely handled by the tapes. However, it forces the types to be fully distributed, at any point, which may lead to an exponential blowup in the number of tapes required. For example, the identity on $(A \oplus B) \otimes C$ has to be distributed and is therefore a diagram on $(A \otimes C) \oplus (A \otimes B)$, while in the Many-Worlds it consists of a simple wire of the non-distributed type:



7.7. **Multiplicative Additive Proof Nets.** The syntax of the Many-Worlds is very close to the one of multiplicative additive proof nets, in fact, the connexion between proof nets and quantum computation has already been studied in the multiplicative setting [Dun04, Dun06] and also in the Tensor-Sum Logic (that we mentioned above) [Dun09]. The Many-Worlds Calculus acts as a model of multiplicative additive linear logic [Gir87], where both additive connectives are collapsed to the $\oplus$ and both multiplicative connectives are collapsed to the $\otimes$. Meanwhile, the multiplicative (resp. additive) units are collapsed to the type $\mathbb{1}$(resp. $\mathbb{0}$). While the notion of worlds is very similar to the notion of weights [Gir96, Mai07], the correspondence is not exact, and the notion of validity criterion of proof nets such as the one present in [HVG05] cannot be applied to the Many-Worlds Calculus (as everything is self-dual).

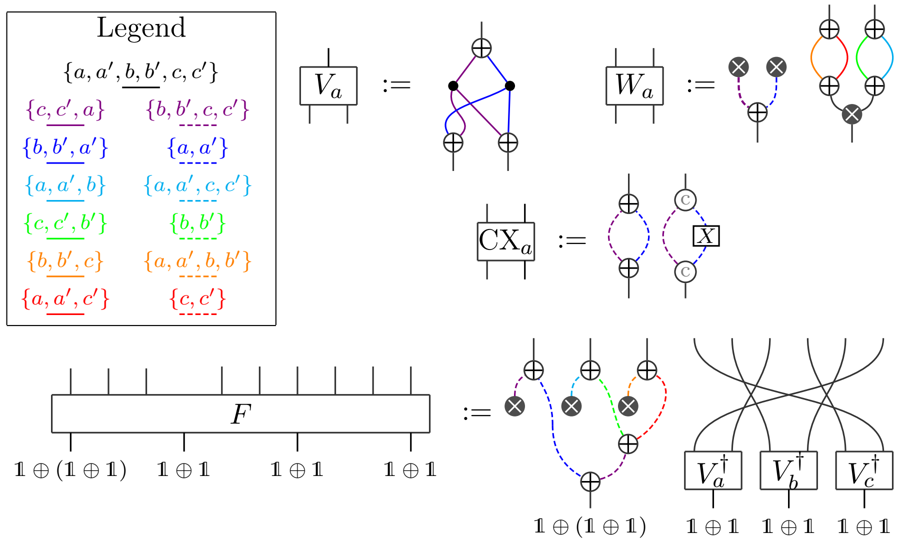
7.8. **Routed Circuits.** Routed circuits [VKB21] are a generalization of usual quantum circuits, designed to accommodate indefinite causal orders. This setting allows for "feedback loops" (partial traces), and uses *sectorial constraints* on the different Hilbert spaces to ensure these do not break unitarity. The quantum switch is the prime example of a non-causally ordered process that can be expressed as a routed circuit. We have already shown how it can also be represented as a Many-Worlds diagram. More generally, we conjecture that all processes expressed as routed circuits can be turned into Many-Worlds diagrams, in a way that preserves the semantics, by using the worlds system to represent the sectorial constraints. While the precise connection is left as future work, we illustrate it through an additional example of a non-causally ordered process, namely the Lugano process [BW16, VOKB23]. The Lugano process is a tri-partite process whose purpose is to elect a leader among the three parties. Each party can only vote for either of the two other parties, and the party that gets the majority is elected. What makes this process non-causally ordered, is the fact that each party knows whether they have won or not, *before* they even cast their ballot. Literature shows that, surprisingly, this does not create a grandfather-like paradox, and that this technically classical process can be made quantum. The Lugano process is described as a routed circuit as follows:

$\mathrm{Lugano}(A, B, C) \ :=$



$\mathbb{1} \oplus \mathbb{1}$
$\mathbb{1} \oplus \mathbb{1}$
$\mathbb{1} \oplus \mathbb{1}$
CX_a
CX_b
CX_c
A
B
C
V_a
V_b
V_c
W_a
W_b
W_c
F
$\mathbb{1} \oplus (\mathbb{1} \oplus \mathbb{1})$
$\mathbb{1} \oplus \mathbb{1}$
$\mathbb{1} \oplus \mathbb{1}$
$\mathbb{1} \oplus \mathbb{1}$

where the boxes A, B and C are the parameters of the process, all the wires except the first output are of type $\mathbb{1} \oplus \mathbb{1}$, and the sectorial constraints are implicitly contained in the other components that are given a Many-Worlds representation in Figure 25 (all wires in the above diagram have worlds set $\{a, a', b, b', c, c'\}$ in accordance with the legend in this figure).

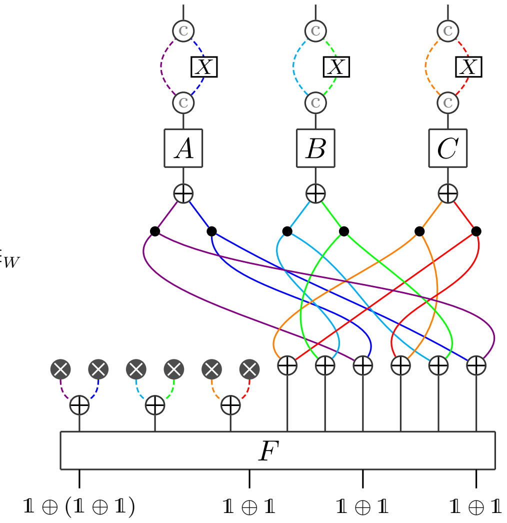


Legend
$\{a, a', \underline{b, b'}, c, c'\}$
$\{c, c', a\}$
$\{b, b', c, c'\}$
$\{b, b', a'\}$
$\{a, a'\}$
$\{a, a', b\}$
$\{a, a', c, c'\}$
$\{c, c', b'\}$
$\{b, b'\}$
$\{b, b', c\}$
$\{a, a', b, b'\}$
$\{a, a', c'\}$
$\{c, c'\}$
V_a :=
W_a :=
CX_a :=
c
X
c
F
:=
$\mathbb{1} \oplus (\mathbb{1} \oplus \mathbb{1})$
$\mathbb{1} \oplus \mathbb{1}$
$\mathbb{1} \oplus \mathbb{1}$
$\mathbb{1} \oplus \mathbb{1}$
$V_a^\dagger$
$V_b^\dagger$
$V_c^\dagger$
$\mathbb{1} \oplus (\mathbb{1} \oplus \mathbb{1})$
$\mathbb{1} \oplus \mathbb{1}$
$\mathbb{1} \oplus \mathbb{1}$
$\mathbb{1} \oplus \mathbb{1}$

Figure 25. Components of the Lugano process. V_i, W_i and CX_i for $i \in \{b, c\}$ are defined similarly by adequately permuting the worlds a, a', b, b', c and c'. Box X is the usual Not gate, as represented e.g. in Figure 14. We keep it as a box here simply to avoid having to use yet another 6 colors (for worlds sets $\{a\}$, $\{a'\}$, $\{b\}$, $\{b'\}$, $\{c\}$ and $\{c'\}$).

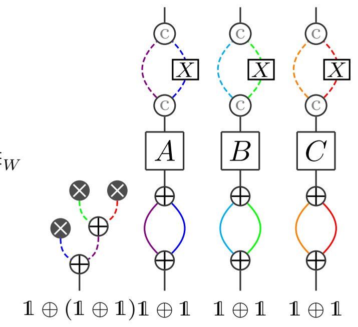
Using the equational theory, one can simplify the diagram, especially by making use of wires of type $\mathbb{1}$, and using the same color code as in Figure 25 to represent worlds sets:

Lugano$(A, B, C) \equiv_W$



X
A
B
C
F
$\mathbb{1} \oplus (\mathbb{1} \oplus \mathbb{1})$
$\mathbb{1} \oplus \mathbb{1}$
$\mathbb{1} \oplus \mathbb{1}$
$\mathbb{1} \oplus \mathbb{1}$

$\equiv_W$



X
A
B
C
$\mathbb{1} \oplus (\mathbb{1} \oplus \mathbb{1})$
$\mathbb{1} \oplus \mathbb{1}$
$\mathbb{1} \oplus \mathbb{1}$
$\mathbb{1} \oplus \mathbb{1}$

The intuition behind the behavior of the process can be recovered once we make sense of the different world sets involved. In particular, $\{c, c', a\}$ can be understood as "A votes for C", and similarly for all the world sets used in boxes V_i. Then, worlds set $\{a, a'\}$ can be understood as "A wins the vote", which makes sense as $\{a, a'\} = \{a, a', b\} \cap \{a, a', c'\}$ i.e. "C votes for A *and* B votes for A". The leftmost output then is a qutrit whose value indicates which of the three parties won the election.

8. Conclusion

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

This language is a first step towards the unification of languages based on the tensor $\otimes$ and those based on the biproduct $\oplus$. This allows us to reason about both systems in parallel, and superposition of executions, as shown by the encoding of the Quantum Switch in Example 5.2 and the translation from term of the language in Section 3.

Following this translation, a natural development of the Many-Worlds calculus consists in accommodating function abstraction and recursion in the language. The question of a complete equational theory for the language on mixed states (e.g. via the discard construction [CJPV19]) is also left open.

Finally, while our language allows for quantum control and can faithfully represent the Quantum Switch, it does not entirely capture another language that aims at formalizing quantum control, namely the PBS-Calculus [CP20]. How and in which context could we capture the PBS-Calculus is left for future work.

Acknowledgement

This work has been partially funded by the French National Research Agency (ANR) by the project RECIPROG ANR-21-CE48-0019, PPS ANR-19-CE48-0014, TaQC ANR-22-CE47-0012 and within the framework of "Plan France 2030", under the research projects EPIQ

ANR-22-PETQ-0007, OQULUS ANR-22-PETQ-0013 and HQI-R&D ANR-22-PNCQ-0002. This work has also been partially funded by MIUR FARE project CAFFEINE, “Compositional and Effectful Program Distances”, R20LW7EJ7L.

References

[AFCB14] Mateus Araú jo, Adrien Feix, Fabio Costa, and Časlav Brukner. Quantum circuits cannot control unknown operations. *New Journal of Physics*, 16(9):093026, sep 2014. URL: `https://doi.org/10.1088%2F1367-2630%2F16%2F9%2F093026`, `doi:10.1088/1367-2630/16/9/093026`.

[AG05] 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`.

[Bar77] M. Bartha. A finite axiomatization of flowchart schemes. *Acta Inf.*, 8(2):203–217, jun 1977. URL: `https://dl.acm.org/doi/abs/10.5555/43192.43202`.

[BDGS23] Filippo Bonchi, Alessandro Di Giorgio, and Alessio Santamaria. Deconstructing the calculus of relations with tape diagrams. *Proc. ACM Program. Lang.*, 7(POPL), January 2023. `doi:10.1145/3571257`.

[BK19] Miriam Backens and Aleks Kissinger. ZH: A complete graphical calculus for quantum computations involving classical non-linearity. In Peter Selinger and Giulio Chiribella, editors, *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018*, volume 287 of *Electronic Proceedings in Theoretical Computer Science*, pages 23–42, 2019. `doi:10.4204/EPTCS.287.2`.

[BW16] Ämin Baumeler and Stefan Wolf. The space of logically consistent classical processes without causal order. *New Journal of Physics*, 18(1):013036, January 2016. URL: `http://dx.doi.org/10.1088/1367-2630/18/1/013036`, `doi:10.1088/1367-2630/18/1/013036`.

[Car21] Titouan Carette. *Wielding the ZX-calculus, Flexsymmetry, Mixed States, and Scalable Notations. (Manier le ZX-calcul, flexsymétrie, systèmes ouverts et limandes)*. PhD thesis, University of Lorraine, Nancy, France, 2021. URL: `https://tel.archives-ouvertes.fr/tel-03468027`.

[CBB+21] Christophe Chareton, Sébastien Bardin, François Bobot, Valentin Perrelle, and Benoît Valiron. An automated deductive verification framework for circuit-building quantum programs. In Nobuko Yoshida, editor, *Programming Languages and Systems*, pages 148–177, Cham, 2021. Springer International Publishing.

[CD11] Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, April 2011. URL: `http://dx.doi.org/10.1088/1367-2630/13/4/043016`, `doi:10.1088/1367-2630/13/4/043016`.

[CDH20] Cole Comfort, Antonin Delpeuch, and Jules Hedges. Sheet diagrams for bimonoidal categories, 2020. URL: `https://arxiv.org/abs/2010.13361`, `doi:10.48550/ARXIV.2010.13361`.

[CDP08] 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`.

[CDPV13] G. Chiribella, G. M. D'Ariano, P. Perinotti, and B. Valiron. Quantum computations without definite causal structure. *Physical Review A*, 88:022318, 2013. `doi:10.1103/PhysRevA.88.022318`.

[Cha23] Kostia Chardonnet. *Towards a Curry-Howard Correspondence for Quantum Computation*. Theses, Université Paris-Saclay, January 2023. URL: `https://theses.hal.science/tel-03959403`.

[CHM+22] Alexandre Clément, Nicolas Heurtel, Shane Mansfield, Simon Perdrix, and Benoît Valiron. LOv-Calculus: A Graphical Language for Linear Optical Quantum Circuits. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.MFCS.2022.35`, `doi:10.4230/LIPIcs.MFCS.2022.35`.

[CHP19] Titouan Carette, Dominic Horsman, and Simon Perdrix. SZX-Calculus: Scalable Graphical Quantum Reasoning. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 55:1–55:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/10999`, `doi:10.4230/LIPIcs.MFCS.2019.55`.

[CJPV19] Titouan Carette, Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Completeness of Graphical Languages for Mixed States Quantum Mechanics. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 108:1–108:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/10684`, `doi:10.4230/LIPIcs.ICALP.2019.108`.

[CK17] 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`.

[CP20] Alexandre Clément and Simon Perdrix. PBS-Calculus: A Graphical Language for Coherent Control of Quantum Computations. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.MFCS.2020.24`, `doi:10.4230/LIPIcs.MFCS.2020.24`.

[Dal17] 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`.

[dFC23] Giovanni de Felice and Bob Coecke. Quantum linear optics via string diagrams. *Electronic Proceedings in Theoretical Computer Science*, 394:83–100, November 2023. URL: `http://dx.doi.org/10.4204/EPTCS.394.6`, `doi:10.4204/eptcs.394.6`.

[Dun04] Ross Duncan. Believe it or not, bell states are a model of multiplicative linear logic. Technical report, 2004.

[Dun06] R Duncan. *Types for Quantum Computing*. PhD thesis, Oxford University, 2006.

[Dun09] Ross Duncan. Generalized proof-nets for compact categories with biproducts. In Simon Gay and Ian Mackie, editors, *Semantic Techniques in Quantum Computation*, chapter 3, pages 70–134. Cambridge University Press, 2009.

[FH65] Richard P. Feynman and A. R. Hibbs. *Quantum Mechanics and Path Integrals*. McGraw-Hill Publishing Company, 1965.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical computer science*, 50(1):1–101, 1987.

[Gir96] Jean-Yves Girard. Proof-nets: the parallel syntax for proof-theory. *Lecture Notes in Pure and Applied Mathematics*, pages 97–124, 1996.

[GLR$^+$13] 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`.

[Had15] Amar Hadzihasanovic. A diagrammatic axiomatisation for qubit entanglement. In *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 573–584, Jul 2015. `doi:10.1109/LICS.2015.59`.

[HR15] Philip Hackney and Marcy Robertson. On the category of props. *Appl. Categorical Struct.*, 23(4):543–573, 2015. `doi:10.1007/s10485-014-9369-4`.

[HV19] Chris Heunen and Jamie Vicary. *Categories for Quantum Theory*. Oxford University Press, nov 2019. URL: `https://doi.org/10.1093%2Foso%2F9780198739623.001.0001`, `doi:10.1093/oso/9780198739623.001.0001`.

[HVG05] Dominic J. D. Hughes and Rob J. Van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic. *ACM Trans. Comput. Logic*, 6(4):784–842, October 2005. `doi:10.1145/1094622.1094629`.

[Lac04] Stephen Lack. Composing PROPs. In *Theory and Applications of Categories*, volume 13, pages 147–163, 2004. URL: `http://www.tac.mta.ca/tac/volumes/13/9/13-09abs.html`.

[Laf89] Yves Lafont. Interaction nets. In *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '90, page 95–108, New York, NY, USA, 1989. Association for Computing Machinery. doi:10.1145/96709.96718.

[Mai07] Roberto Maieli. Cut Elimination for Monomial Proof Nets of the Purely Multiplicative and Additive Fragment of Linear Logic. Research report, Department of Mathematics and Physics, Roma TRE University, December 2007. URL: https://hal.science/hal-01153910.

[Mel14] Paul-André Melliès. Local states in string diagrams. In Gilles Dowek, editor, *Rewriting and Typed Lambda Calculi*, pages 334–348, Cham, 2014. Springer International Publishing.

[PMA+15] Lorenzo M. Procopio, Amir Moqanaki, Mateus Araújo, Fabio Costa, Irati Alonso Calafell, Emma G. Dowd, Deny R. Hamel, Lee A. Rozema, Časlav Brukner, and Philip Walther. Experimental superposition of orders of quantum gates. *Nature Communications*, 6(1):7913, 2015. doi:10.1038/ncomms8913.

[PMM+17] 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.

[PRZ17] 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.

[Sta15] 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.

[SVV18] 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.

[TCM+21] Márcio M. Taddei, Jaime Cariñe, Daniel Martínez, Tania García, Nayda Guerrero, Alastair A. Abbott, Mateus Araújo, Cyril Branciard, Esteban S. Gómez, Stephen P. Walborn, Leandro Aolita, and Gustavo Lima. Computational advantage from the quantum superposition of multiple temporal orders of photonic gates. *PRX Quantum*, 2:010320, 2021. doi:10.1103/PRXQuantum.2.010320.

[VC21] Augustin Vanrietvelde and Giulio Chiribella. Universal control of quantum processes using sector-preserving channels. *Quantum Information and Computation*, 21(15&16):1320–1352, nov 2021. URL: https://doi.org/10.26421%2Fqic21.15-16-5, doi:10.26421/qic21.15-16-5.

[VKB21] Augustin Vanrietvelde, Hlér Kristjánsson, and Jonathan Barrett. Routed quantum circuits. *Quantum*, 5:503, 2021. doi:10.22331/q-2021-07-13-503.

[VOKB23] Augustin Vanrietvelde, Nick Ormrod, Hlér Kristjánsson, and Jonathan Barrett. Consistent circuits for indefinite causal order, 2023. URL: https://arxiv.org/abs/2206.10042, arXiv:2206.10042.

[WDAB21] 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.

[ZRK+11] Xiao-Qi Zhou, Timothy C. Ralph, Pruet Kalasuwan, Mian Zhang, Alberto Peruzzo, Benjamin P. Lanyon, and Jeremy L. O'Brien. Adding control to arbitrary unknown quantum operations. *Nature Communications*, 2(1), aug 2011. URL: https://doi.org/10.1038%2Fncomms1392, doi:10.1038/ncomms1392.