

Towards the simulation of higher-order quantum resources: a general type-theoretic approach

Samuel B. Steakley, Elia Zanoni, Carlo Maria Scandolo

Department of Mathematics & Statistics, University of Calgary,
Calgary, T2N 1N4, AB, Canada.

Institute for Quantum Science and Technology, University of Calgary,
Calgary, T2N 1N4, AB, Canada.

Contributing authors: samuel.steakley@ucalgary.ca;
elia.zanoni@ucalgary.ca; carlomaria.scandolo@ucalgary.ca;

Abstract

Quantum resources exist in a hierarchy of multiple levels. At order zero, quantum states are transformed by linear maps (channels, or gates) in order to perform computations or simulate other states. At order one, gates and channels are transformed by linear maps (superchannels) in order to simulate other gates. To develop a full hierarchy of quantum resources, beyond those first two orders, and to account for the fact that quantum protocols can interconvert resources of different orders, we need a theoretical framework that addresses all orders in a uniform manner. We introduce a framework based on a system of types, which label the different kinds of objects that are present at different orders. We equip the framework with a parallel product operation that modifies and generalizes the tensor product so as to be operationally meaningful for maps of distinct and arbitrary orders. Finally, we introduce a family of convex cones that generalize the notion of complete positivity to all orders, with the aim of characterizing the objects that are physically admissible, facilitating an operational treatment of quantum objects at any order.

1 Introduction

Quantum science and technology have been the subject of such interest in recent years [1–4] that this period has been dubbed the second quantum revolution [5]. The power of quantum information processing is being investigated in various areas, but especially

quantum sensing [6], quantum computing [7, 8], and quantum communications [9]. The overarching theme is that quantum technologies and devices constitute a resource that confers a unique quantum advantage in information processing. Therefore, it is imperative to develop a scientific theory that precisely captures the notion of quantum resource. Indeed, we must seek to understand the general laws of how quantum resources can be generated, interconverted, and optimally exploited [10, 11].

In the hierarchy of different kinds of quantum resources, the base level consists of quantum states, with valuable properties such as superposition [12] and coherence [13], being manipulated by transformations known as quantum channels. These states are called *static resources* [10, 11] and they are e.g. the subject of quantum computation: we use quantum gates to manipulate input states in order to create superpositions and entanglement, and to run quantum algorithms. We refer to this level of the hierarchy as level zero.

At the next level of the hierarchy, we consider e.g. quantum gates. In general, the quantum operations that perform the basic steps of an algorithm are not all equally easy to implement [14, 15], and so for efficiency, it may become necessary to perform optimizations on a given quantum circuit. For instance, we may wish to reduce a circuit’s T -count [16], i.e. the number of T gates it contains. The general issue here is the task of circuit compilation: an algorithm can be accomplished using a given collection of gates, but we wish to simulate those gates using another, less costly collection. We encounter a similar problem when we face a communication task in which the given channels may be noisy, in which case we want a protocol that utilizes the given channels in order to simulate a channel with as little noise as possible. A solution to either task is a protocol that transforms an input channel into an output channel, and which is necessarily a kind of quantum transformation known as a *superchannel* [17, 18]. This forms level one of the hierarchy of quantum resources: quantum channels as objects being manipulated by superchannels. In this context, valuable channels are known as *dynamical resources* [10, 19–23].

Although static and dynamical resources are of distinct kinds, they are not separate in a practical sense, because certain quantum protocols make it possible to use a static resource to simulate a dynamical resource, or *vice versa*. In effect, such a protocol allows one kind of resource to be converted into the other. For example, this is the case in the teleportation protocol [24], in which one entangled state is converted into a noiseless channel. But there are many other such protocols [25, 26]. This leads to inevitable tradeoffs between static and dynamical resources in information processing tasks, and so we need theoretical methods that treat them on a common ground, instead of separate methodologies.

Static and dynamical resources are only two levels of a much larger hierarchy. The same reasoning that leads us to consider the manipulation of channels by superchannels also invites us to consider all feasible ways of manipulating superchannels. This leads to level two of the hierarchy: superchannels as objects being transformed by an even “higher” form of quantum protocol (for lack of a commonly accepted name, and despite the awkwardness, these could be called “supersuperchannels”). In fact, there is no theoretical reason to stop at any particular level: for any number n , we may define level $n + 1$ by making its objects the transformations of level n , and making its

transformations all feasible ways of manipulating those objects. Thus, the full hierarchy includes infinitely many levels. In order to capture the full possibilities of quantum protocols, we need a theoretical methodology that applies to protocols at all levels.

By investigating the levels above one, researchers have uncovered quantum protocols that exhibit unexpected and valuable features, such as the quantum SWITCH [27–32] with its feature of indefinite causal order [33–36]. It has been found that the SWITCH speeds up certain information processing tasks [28–32, 37], and thus we should view these “higher” protocols as yet another source of quantum advantage. There are reasons for interest for fundamental science too, since the phenomenon of indefinite causal order has been theoretically linked to quantum gravity. The idea is that if quantum superpositions of spacetime occur at the quantum gravity scale, then the causal structure of local light cones may itself be in superposition, hence indefinite. Studying higher quantum protocols could provide us with candidate models for indefinite causal order at the level of spacetime.

We propose a theoretical framework that addresses both of the aforementioned desiderata: all possible levels of the hierarchy are included, and they are all unified in the sense that we can consider all ways of combining, simulating, and interconverting objects from arbitrary levels. The term of art for “level” is *order*, and so this is all to say that our formalism addresses the full scope of what is known as higher-order quantum theory.

Our framework is partly modeled on the one discussed in [38–40] (see [41–48] for related approaches). Like theirs, it addresses higher-order quantum theory for finite-dimensional systems, and it identifies and differentiates the different kinds of higher-order transformation with the aid of a simple type system. However, we introduce a novel algebraic method for describing the parallel application of two higher-order maps of any arbitrary orders. As a further technical point, we opt against using the Choi representation to define higher-order quantum objects. One advantage of this is that we avoid the ambiguities of which systems to associate with different entries of a concrete Choi matrix, as highlighted in [49].

We first build a set of types, $\mathbf{Types}_{\mathbb{A}}$, out of a given “base” set $\mathbb{A}$ of letters indicating physical systems. Types may be thought of as labels that we use to differentiate and identify all the various kinds of objects (states and higher-order transformations) that arise in higher-order quantum theory. Using a simple inductive construction, we associate each type $x \in \mathbf{Types}_{\mathbb{A}}$ with a Hilbert space $\mathbf{L}(x)$ of linear maps, which we call the space of linear maps of type x . The Hilbert spaces $\mathbf{L}(x)$ provide a universe within which the physically meaningful maps (states, channels, superchannels, etc.) are to be characterized.

We then introduce an algebraic operation on linear maps, which we call the *parallel product* and denote by $\boxtimes$. Given a pair of types $x, y \in \mathbf{Types}_{\mathbb{A}}$ as well as a map $\mathcal{M}$ of type x and a map $\mathcal{N}$ of type y , we define a parallel product type $x \boxtimes y \in \mathbf{Types}_{\mathbb{A}}$ and a parallel product map $\mathcal{M} \boxtimes \mathcal{N}$ of type $x \boxtimes y$. The parallel product is a generalization of the notion of “extended event” defined in [39, Definition 4.3], in that the parallel product allows maps of all types to be combined, rather than only allowing the extension with a base type. The motivating example for the parallel product is the tensor product of channels. We will see that the tensor product of two higher-order transformations

of different kinds is *not* suitable as a parallel product of the two, but that a general parallel product operation can be built by using the tensor product within a simple inductive formula.

Finally, we take a first step toward the general problem of characterizing the subset of physically meaningful maps within each set $L(x)$. Our point of departure consists of the fact that only the positive semidefinite operators are used to describe states of a quantum system, as well as the consequent facts that channels must be completely positive (or, “CP”) and that superchannels must be completely CP-preserving [18, 50]. Extending the same logic, a physically meaningful map of any type x should satisfy a certain generalized property of complete positivity, and this is what we define: for each $x \in \mathbf{Types}_{\mathbb{A}}$ we define a subset $K(x) \subset L(x)$, hence a property of linear maps of type x , that inductively generalizes the notions of completely positive map and completely CP-preserving map to all types. We show that each $K(x)$ forms a convex cone, just like the subset of positive semidefinite operators does.

The rest of the paper is organized as follows. In Section 2, we introduce our type system and we provide a convenient graphical representation in terms of trees. Then we define the L family of spaces of higher-order linear maps. In Section 3, we introduce the parallel product of types and maps, again with a convenient interpretation in terms of trees. In Section 4, we introduce a higher-order generalization of complete positivity. Conclusions are drawn in Section 5.

2 Types and linear maps

The first main task in this section is to construct a set $\mathbf{Types}_{\mathbb{A}}$ of types. For our purposes, types may be thought of as *labels* that we use to identify and differentiate all the different kinds of objects that arise in higher-order quantum theory.

At the most basic level, we assume that we are interested in studying some particular quantum system and that we have chosen a label to refer to that system. For example, if we wish to consider a generic 2-dimensional quantum system, then we may label this system by the letter B , and we may choose the Hilbert space $\mathcal{H}_B = \mathbb{C}^2$ to model it. In general, we let $\mathbb{A}$ be a set of labels referring to the quantum systems we wish to study, and these labels form the base of our type system: for each $A \in \mathbb{A}$, states of system A are *states of type* A .

For any pair of systems, there is a corresponding composite system. We assign the composite system a type by combining the two labels into a “word”: for each $A, B \in \mathbb{A}$, states of the corresponding composite system are *states of type* AB . Composite systems with three or more components are handled in the same way; e.g. for any three systems $A, B, C \in \mathbb{A}$, states of the corresponding composite system are *states of type* ABC .

In order to describe transformations, rather than states, we introduce a new symbol into our types: the arrow symbol $\rightarrow$. For example, the type of a channel consists of an arrow with an input system on its left and an output system on its right: for each $A, B \in \mathbb{A}$, channels with input A and output B are *channels of type* $A \rightarrow B$. By extension, a generic superchannel [17, 18] would be of type $(A \rightarrow B) \rightarrow (C \rightarrow D)$, and a generic transformation that takes a channel as input and returns a superchannel as output would be of type $(A \rightarrow B) \rightarrow ((C \rightarrow D) \rightarrow (E \rightarrow F))$. (Notice the use of

parentheses to separate arrow symbols, which is necessary in order to disambiguate the “order of operations.”) The full set contains infinitely many types, for describing the infinitely many kinds of higher-order transformations.

The second main task is to put the set $\mathbf{Types}_{\mathbb{A}}$ of types to its first major use: to inductively define a finite-dimensional Hilbert space $\mathbf{L}(x)$, for each $x \in \mathbf{Types}_{\mathbb{A}}$, whose elements we call the *linear maps of type x* . Informally, we refer to $\mathbf{L}$ as a *family* of Hilbert spaces, because it consists of one space for each type. The $\mathbf{L}$ family provides us with a universe of higher-order linear maps within which the physically meaningful maps are to be characterized.

2.1 Types

Let $\mathbb{A}$ be a set, whose elements we think of as labels for physical systems. We now present a construction that generalizes the former motivating examples into a full set of types generated by $\mathbb{A}$.

First we construct the set of elementary types. These types describe all base systems, all composites of base systems, and also the “trivial” system (to be explained momentarily).

Definition 2.1. $\mathbf{EleTypes}_{\mathbb{A}}$ is the set of all finite-length words made of elements of $\mathbb{A}$.

“Word” simply means sequence of symbols. For each base system $A \in \mathbb{A}$, we identify the label A with the length-one sequence $A \in \mathbf{EleTypes}_{\mathbb{A}}$. Composite systems are described by sequences of length greater than one, including repeated instances of the same system; e.g. $A, B, C \in \mathbb{A}$ entails $AABCB \in \mathbf{EleTypes}_{\mathbb{A}}$.

The elementary types also include a unique sequence of length zero (hence, $\mathbf{EleTypes}_{\mathbb{A}}$ is non-empty even if $\mathbb{A}$ is empty). We call this type the *trivial type*, and we denote it by I (in particular, we assume that I is *not* an element of $\mathbb{A}$). We use the trivial type to refer to the *trivial system*, which is a sort of fictitious system that indicates the lack of a physical system, or “nothing.”

The full set of types is constructed by inductively expanding $\mathbf{EleTypes}_{\mathbb{A}}$ with all “arrow types,” as follows.

Definition 2.2. $\mathbf{Types}_{\mathbb{A}}$ is the smallest set of formal expressions satisfying the following two rules:

$$\mathbf{EleTypes}_{\mathbb{A}} \subseteq \mathbf{Types}_{\mathbb{A}} \tag{R1}$$

$$\text{If } x, y \in \mathbf{Types}_{\mathbb{A}}, \text{ then } x \rightarrow y \in \mathbf{Types}_{\mathbb{A}} \tag{R2}$$

The notation $x \rightarrow y$ denotes the concatenation of the expressions x and y , with an arrow symbol placed in between; also, round brackets must be placed around x (respectively, around y) if x (resp. y) contains an arrow symbol. The purpose of the brackets is to disambiguate the “order of operations.” E.g. if $x = AB$ and $y = C$ then $x \rightarrow y = AB \rightarrow C$, but if $y = B \rightarrow C$ then $x \rightarrow y = AB \rightarrow (C \rightarrow D)$.

Due to rule (R2), $\text{Types}_{\mathbb{A}}$ is necessarily an infinite set. Even if $\mathbb{A}$ is empty, $\text{EleTypes}_{\mathbb{A}}$ contains the trivial type (i.e. the empty sequence), denoted I . Then $I \in \text{Types}_{\mathbb{A}}$ by (R1). Then, by repeated application of (R2), it follows that $I \rightarrow I \in \text{Types}_{\mathbb{A}}$, and $I \rightarrow (I \rightarrow I) \in \text{Types}_{\mathbb{A}}$, and $I \rightarrow (I \rightarrow (I \rightarrow I)) \in \text{Types}_{\mathbb{A}}$, and so on.

2.1.1 Types as trees

In the fields of mathematics that study formal languages (e.g. mathematical logic), the set $\text{Types}_{\mathbb{A}}$ would be recognized as a simple kind of formal language. We have chosen to introduce types as “formal expressions,” by which we mean sequences of symbols, but it is well-known that the elements of this kind of language can be formalized in more than one useful way. They can also be formalized as *syntax trees*. It would take us too far afield to go over the precise details, but we can still profit from visualizing a type in two equivalent ways: as a formal expression or as a syntax tree. This is best demonstrated by way of example.

A syntax tree is a certain kind of tree, which means that it consists of *vertices* that are connected by *edges* (an edge is drawn as a line from one vertex to another). Suppose $A, B, C, \dots$ are elements of $\mathbb{A}$. When we render the type $A \rightarrow B$ as a syntax tree, as in (1), this is done as follows. Each of the three symbols becomes a vertex. We place the $\rightarrow$ vertex at the top, and we call this vertex the *root* of the tree. Finally, we add edges connecting A and B to the root, placing A below left and B below right.

$$A \rightarrow B \quad \rightsquigarrow \quad \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ A \quad B \end{array} \quad (1)$$

For the type $(A \rightarrow B) \rightarrow (C \rightarrow D)$, we apply a similar procedure, but now in multiple steps proceeding from the bottom up. We begin with the $\rightarrow$ symbols which are the “deepest” with respect to nesting of parentheses. By this we mean each $\rightarrow$ symbol meeting the following description: a $\rightarrow$ symbol such that the symbols to its immediate left and right are both letters (elements of $\mathbb{A}$), as opposed to parenthesis symbols. In the type $(A \rightarrow B) \rightarrow (C \rightarrow D)$, there are two deepest $\rightarrow$ symbols: the $\rightarrow$ with A on its left and B on its right, and the $\rightarrow$ with C on its left and D on its right. We take these two and build their sub-trees according to the same procedure described above, which results in the following two trees:

$$A \rightarrow B \quad \rightsquigarrow \quad \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ A \quad B \end{array} \quad C \rightarrow D \quad \rightsquigarrow \quad \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ C \quad D \end{array} \quad (2)$$

Finally, we form one big tree by placing the outermost $\rightarrow$ at the top and joining the two sub-trees underneath it, with an edge from the overall root to the root of each

sub-tree:

$$(A \rightarrow B) \rightarrow (C \rightarrow D) \rightsquigarrow \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ \rightarrow \quad \rightarrow \\ \swarrow \searrow \quad \swarrow \searrow \\ A \quad B \quad C \quad D \end{array} \quad (3)$$

Extending the same “bottom-up” procedure, one can translate any type to a tree. We provide a few more examples:

$$(A \rightarrow BA) \rightarrow ABC \rightsquigarrow \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ \rightarrow \quad ABC \\ \swarrow \searrow \\ A \quad BA \end{array} \quad (4)$$

$$ABC \rightarrow (A \rightarrow BA) \rightsquigarrow \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ ABC \quad \rightarrow \\ \quad \swarrow \searrow \\ \quad A \quad BA \end{array} \quad (5)$$

$$(AB \rightarrow C) \rightarrow (DEF \rightarrow (GH \rightarrow K)) \rightsquigarrow \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ \rightarrow \quad \rightarrow \\ \swarrow \searrow \quad \swarrow \searrow \quad \swarrow \searrow \\ AB \quad C \quad DEF \quad \rightarrow \\ \quad \quad \quad \quad \quad \quad \quad \swarrow \searrow \\ \quad \quad \quad \quad \quad \quad \quad GH \quad K \end{array} \quad (6)$$

Through the correspondence that these examples illustrate, one sees that in a tree there is no need for parentheses because the edges between $\rightarrow$ vertices make the “order of operations” clear. And for large types, a tree is more legible because it is laid out in a 2-dimensional form, instead of a mess of brackets.

2.1.2 Basic properties of types

It is known that the set $\mathbf{Types}_{\mathbb{A}}$, by being constructed in the manner of Definition 2.2, has the property of being freely inductively generated with respect to rules (R1) and (R2). For background on this fact and the precise meaning of “freely inductively generated,” we refer the reader to [51]. However, we are mainly interested in two consequences of this.

First, there is Lemma 2.3, which is a close corollary of the fact that $\mathbf{Types}_{\mathbb{A}}$ is freely inductively generated with respect to rules (R1) and (R2), and which provides a simple way of understanding all of the elements of $\mathbf{Types}_{\mathbb{A}}$ by dividing them into two distinct categories. A type x must either be elementary, or else be the arrow type $a \rightarrow b$ for some unique choice of types a and b (which clearly must be “smaller” than x , i.e. with fewer symbols).

Lemma 2.3. Every $x \in \mathbf{Types}_{\mathbb{A}}$ satisfies exactly one of the following statements:

- (i) $x \in \text{EleTypes}_{\mathbb{A}}$.
- (ii) $x = a \rightarrow b$ for a unique pair $a, b \in \text{Types}_{\mathbb{A}}$.

We need not dwell on the proof, but for details the interested reader can refer to [52] or [53]. One can easily judge which of the two above cases a given type falls into: either the type contains no $\rightarrow$ symbol, or else there is exactly one $\rightarrow$ symbol on the outside of all brackets. In that case, a is the type on the left of the outermost $\rightarrow$, and b is the type on the right. When viewing a type as a tree, one instead finds the uppermost $\rightarrow$ symbol, in which case a is the type given by the full sub-tree attached to the root's left-hand edge, and b is the one attached to the right-hand edge.

As a consequence of Lemma 2.3, each non-elementary type has a uniquely defined input type and output type.

Definition 2.4. Given $x = a \rightarrow b \in \text{Types}_{\mathbb{A}}$, the *input type* of x is defined

$$\text{in}(x) = a, \quad (7)$$

and the *output type* of x is defined

$$\text{out}(x) = b. \quad (8)$$

The second important consequence that we take from $\text{Types}_{\mathbb{A}}$ being freely inductively generated is that it is valid to carry out proofs and more general constructions on $\text{Types}_{\mathbb{A}}$ by induction over rules (R1) and (R2). We make constant use of these techniques throughout this work. Definition 2.5 provides a first, simple example.

The term “order,” which we have used informally up to this point, is now defined as a property of types, describing the degree of “nestedness” of arrow symbols in a type, or equivalently describing the height of the syntax tree. A simple inductive formula is used.

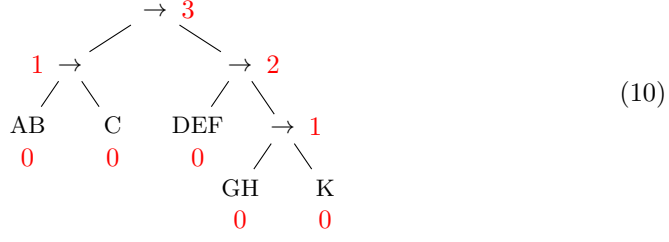
Definition 2.5. Let $x \in \text{Types}_{\mathbb{A}}$. The *order* of x is defined:

$$\text{ord}(x) = \begin{cases} 0 & \text{if } x \in \text{EleTypes}_{\mathbb{A}} \\ 1 + \max\{\text{ord}(a), \text{ord}(b)\} & \text{if } x = a \rightarrow b \end{cases} \quad (9)$$

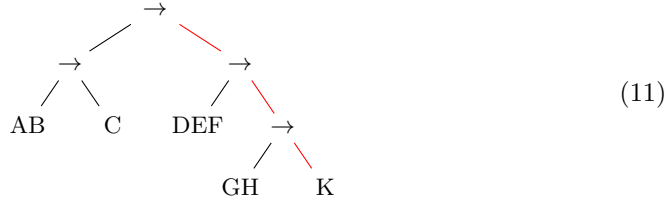
We demonstrate how to calculate the order of a type in the following example.

Example 2.6. Let $x = (AB \rightarrow C) \rightarrow (DEF \rightarrow (GH \rightarrow K))$. One way to compute the order of x is to apply Definition 2.5 in an algorithmic manner, compute the order

“from the bottom up.”



In this figure we have annotated in red the result of assigning zero to the elementary types in the tree, and assigning to each $\rightarrow$ one plus the maximum of the numbers at the vertices immediately beneath it. This shows that $\text{ord}(x) = 3$. The same result is also given, in complete generality, by finding the maximum length of all paths (with non-repeat edges) from the root vertex to one of the bottom vertices. In the case of x , this path is indicated in red in the figure below:



Because of this interpretation, this quantity is sometimes known as the height of the tree.

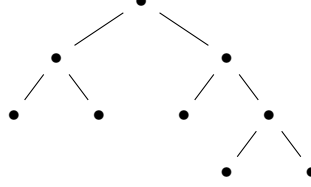
Elementary types can now be characterized as being precisely the types of order zero. Types of order one are exactly the non-elementary types whose input and output types are both of order zero. Types of order two are exactly the non-elementary types whose input and output types are both of order no greater than one, but not both of order zero. And so on and so forth.

By the same token, states are described by types of order zero, channels are described by types of order one, and superchannels are described by types of order two. However, types of order two do not exclusively describe superchannels. A transformation that sends an input state to an output channel would be of type e.g. $A \rightarrow (B \rightarrow C)$, which is of order two. Conversely, a transformation that sends an input channel to an output state would also be described by a type of order two.

The structure of a type is a property that corresponds to the kind of object the type describes irrespective of the particular physical systems the type involves. By “kind of object” we mean state, channel, or superchannel, etc. We use the syntax tree perspective on types to give a clear, simple definition of structure. Notice that in all our examples of types as trees, each vertex of the syntax tree is marked with some label, which is either the $\rightarrow$ symbol or some elementary type. Mathematically speaking, labels are not essential to trees, and in particular there is a precise notion of *unlabeled tree*.

Definition 2.7. Let $x \in \text{Types}_{\mathbb{A}}$. The *structure* of x , $\text{st}(x)$, is the underlying unlabeled tree of x .

Example 2.8. Revisiting the type from Example 2.6, the type's structure is the following unlabeled tree:



Note that the “underlying unlabeled tree” of any elementary type would simply be the tree with exactly one vertex, and this means that all elementary types have the same structure. And clearly, the tree with one vertex is different from the structure of any non-elementary type $a \rightarrow b$.

2.2 Typed linear maps

The main ingredient of the definition of the **L** family is the standard construction which makes $\mathcal{L}(\mathcal{H}_1, \mathcal{H}_2)$, the set of all linear maps with domain $\mathcal{H}_1$ and codomain $\mathcal{H}_2$, into a Hilbert space for any pair of finite-dimensional Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$. $\mathcal{L}(\mathcal{H}_1, \mathcal{H}_2)$ is equipped with the Hilbert-Schmidt inner product, which, we recall, is defined so that, for any pair of linear maps $f, g \in \mathcal{L}(\mathcal{H}_1, \mathcal{H}_2)$,

$$\langle f | g \rangle_{\text{HS}} = \text{Tr}[f^\dagger \circ g]. \quad (12)$$

Note that $f^\dagger$ is defined with respect to the inner products of the Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$. If $\mathcal{H}_1$ and $\mathcal{H}_2$ are both finite-dimensional, then $\mathcal{L}(\mathcal{H}_1, \mathcal{H}_2)$ is finite-dimensional with dimension given by $\dim \mathcal{L}(\mathcal{H}_1, \mathcal{H}_2) = \dim \mathcal{H}_1 \cdot \dim \mathcal{H}_2$.

For each system $A \in \mathbb{A}$, let $\mathcal{H}_A$ be a finite-dimensional complex Hilbert space chosen to model system A . Given an elementary type $\epsilon = A_1 \cdots A_n$ of length $n > 0$, with $A_j \in \mathbb{A}$ for each $j \in \{1, \dots, n\}$, we define

$$\mathcal{H}_\epsilon := \bigotimes_{j=1}^n \mathcal{H}_{A_j}. \quad (13)$$

Now we define the **L** family so that the trivial type **I** is associated with the canonical one-dimensional Hilbert space, elementary types are associated with operators on (tensor products of) the base Hilbert spaces, and arrow types are associated with the construction of Hilbert spaces of linear maps. Note that we write $\mathcal{L}(\mathcal{H}_x)$ as an abbreviation for $\mathcal{L}(\mathcal{H}_x, \mathcal{H}_x)$.

Definition 2.9. Let $x \in \text{Types}_{\mathbb{A}}$. Let

$$L(x) = \begin{cases} \mathbb{C} & \text{if } x = I \\ \mathcal{L}(\mathcal{H}_x) & \text{if } x = A_1 \cdots A_n \in \text{EleTypes}_{\mathbb{A}} \setminus \{I\} \\ \mathcal{L}(L(a), L(b)) & \text{if } x = a \rightarrow b \end{cases} \quad (14)$$

Note that instead of letting $L(I)$ be $\mathcal{L}(\mathbb{C})$, which would directly match the definition of L for other elementary types, we let it be $\mathbb{C}$. This is more convenient, and the overall definition is still consistent with I being an elementary type, because $\mathbb{C}$ and $\mathcal{L}(\mathbb{C})$ are isomorphic.

Quantum objects such as states, channels, and superchannels now fall within the domain encompassed by the L family. Given some arbitrary base systems $A, B, C, D \in \mathbb{A}$, by definition we have

$$L(A) = \mathcal{L}(\mathcal{H}_A) \quad (15)$$

$$L(A \rightarrow B) = \mathcal{L}(\mathcal{L}(\mathcal{H}_A), \mathcal{L}(\mathcal{H}_B)) \quad (16)$$

$$L((A \rightarrow B) \rightarrow (C \rightarrow D)) = \mathcal{L}(\mathcal{L}(\mathcal{L}(\mathcal{H}_A), \mathcal{L}(\mathcal{H}_B)), \mathcal{L}(\mathcal{L}(\mathcal{H}_C), \mathcal{L}(\mathcal{H}_D))) \quad (17)$$

Hence by the standard definitions of quantum state, channel, and superchannel, a (mixed) state of system A is an operator $\rho \in L(A)$; a channel with input system A and output system B is a linear map $\mathcal{M} \in L(A \rightarrow B)$; and a superchannel with input a channel of input system A and output system B , and output a channel of input system C and output system D , is a linear map $\Theta \in L((A \rightarrow B) \rightarrow (C \rightarrow D))$.

Of course, states form a strict subset of $L(A)$, channels a subset of $L(A \rightarrow B)$, and superchannels a subset of $L((A \rightarrow B) \rightarrow (C \rightarrow D))$. Indeed, one of our goals in this work is to identify within each space $L(x)$ a subset of physically meaningful maps (see Section 4).

Following the type theory tradition, we use a colon as notation to indicate the type of a linear map.

Notation 2.10. We write the statement $\mathcal{M} \in L(x)$ equivalently as

$$\mathcal{M} : x \quad (18)$$

Composition makes sense for our typed linear maps. Given maps $\mathcal{M} : a \rightarrow b$ and $\mathcal{N} : b \rightarrow c$, by definition we have

$$\mathcal{M} \in \mathcal{L}(L(a), L(b)), \quad (19)$$

and

$$\mathcal{N} \in \mathcal{L}(L(b), L(c)). \quad (20)$$

Thus $\mathcal{M}$ and $\mathcal{N}$ are composable as functions, and since the composite of linear maps is again linear, we have

$$\mathcal{N} \circ \mathcal{M} \in \mathcal{L}(L(a), L(c)). \quad (21)$$

In other words, we have $\mathcal{N} \circ \mathcal{M} : a \rightarrow c$, just as one would want.

3 The parallel product

In this section, we seek to construct an operation on typed linear maps, to be denoted $\boxtimes$, suitable for taking any two maps $\mathcal{M} : x$ and $\mathcal{N} : y$ and producing a map $\mathcal{M} \boxtimes \mathcal{N} : x \boxtimes y$ to be called the *parallel product* of $\mathcal{M}$ and $\mathcal{N}$. The map $\mathcal{M} \boxtimes \mathcal{N}$ will express the idea that these two linear maps are applied in parallel in an operational scenario. We must define an operation on two different levels: we must define a type $x \boxtimes y$, and we must define a map $\mathcal{M} \boxtimes \mathcal{N}$ of type $x \boxtimes y$.

The parallel product operation should work in a reasonable way for maps of any types, even when the types have different orders, or more generally, different structures. We have chosen the term “parallel product” because it is descriptive of the role that the tensor product operation plays in ordinary quantum theory. However, our starting point is the observation that the tensor product is *not* suitable for higher-order maps of arbitrary types x and y .

3.1 The tensor product as parallel product?

Let us recall the usual role of the tensor product. For a pair of systems A and B, modeled by Hilbert spaces $\mathcal{H}_A$ and $\mathcal{H}_B$, the composite system AB is modeled by the tensor product space $\mathcal{H}_A \otimes \mathcal{H}_B$. This is the role of the tensor product of spaces. Given a state ρ of system A and a state σ of system B, the tensor product state $\rho \otimes \sigma$ is a fully unentangled state of the composite system AB. This state asserts that the condition of system AB is that the A subsystem is prepared in state ρ while independently the B subsystem is prepared in state σ . We might say of this situation that the two states are prepared simultaneously, in parallel.

The tensor product has yet another role in the way it operates on quantum transformations. Recall that for any two linear maps $f \in \mathcal{L}(\mathcal{H}_1, \mathcal{H}_2)$ and $g \in \mathcal{L}(\mathcal{H}_3, \mathcal{H}_4)$, their tensor product is a linear map

$$f \otimes g \in \mathcal{L}(\mathcal{H}_1 \otimes \mathcal{H}_3, \mathcal{H}_2 \otimes \mathcal{H}_4). \quad (22)$$

Moreover, $f \otimes g$ may be defined by the following formula:

$$\forall u \in \mathcal{H}_1, \forall v \in \mathcal{H}_3, (f \otimes g)(u \otimes v) = f(u) \otimes g(v). \quad (23)$$

This formula extends to a unique linear map on the domain $\mathcal{H}_1 \otimes \mathcal{H}_3$, even though it specifies the action only for inputs in the form of a tensor product, because the domain is spanned by these elements.

Let A, B, C, D be systems, and suppose we have two channels:

$$\begin{aligned} \mathcal{M} &\in \mathcal{L}(\mathcal{L}(\mathcal{H}_A), \mathcal{L}(\mathcal{H}_B)) \\ \mathcal{N} &\in \mathcal{L}(\mathcal{L}(\mathcal{H}_C), \mathcal{L}(\mathcal{H}_D)) \end{aligned} \quad (24)$$

Then, in accordance with (22), the tensor product of $\mathcal{M}$ with $\mathcal{N}$ is a linear map

$$\mathcal{M} \otimes \mathcal{N} \in \mathcal{L}(\mathcal{L}(\mathcal{H}_A) \otimes \mathcal{L}(\mathcal{H}_C), \mathcal{L}(\mathcal{H}_B) \otimes \mathcal{L}(\mathcal{H}_D)). \quad (25)$$

Now, by the properties of finite-dimensional complex Hilbert spaces, for any finite-dimensional $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4$ we have

$$\mathcal{L}(\mathcal{H}_1, \mathcal{H}_2) \otimes \mathcal{L}(\mathcal{H}_3, \mathcal{H}_4) = \mathcal{L}(\mathcal{H}_1 \otimes \mathcal{H}_3, \mathcal{H}_2 \otimes \mathcal{H}_4) \quad (26)$$

Therefore, we can rewrite (25) as follows:

$$\mathcal{M} \otimes \mathcal{N} \in \mathcal{L}(\mathcal{L}(\mathcal{H}_A \otimes \mathcal{H}_C), \mathcal{L}(\mathcal{H}_B \otimes \mathcal{H}_D)). \quad (27)$$

Thus, the tensor product channel operates on input states of the composite system AC and returns output states of the composite system BD. In other words, $\mathcal{M} \otimes \mathcal{N}$ is of type AC $\rightarrow$ BD. And in accordance with (23), any product state $\rho \otimes \sigma$ of system AC is sent to the product state $\mathcal{M}(\rho) \otimes \mathcal{N}(\sigma)$. We might say that in the action of $\mathcal{M} \otimes \mathcal{N}$, the two channels act in parallel, with $\mathcal{M}$ acting on the A subsystem and $\mathcal{N}$ acting on the C subsystem.

Now having seen that the tensor product operation provides something like a parallel product of objects both at the level of states and that of channels, and having seen that in both cases this notion of parallel product is suitably related to the notion of composite system, we must observe that the tensor product gives an inadequate result if we try using it to combine a state with a channel. We emphasize that there is nothing *mathematically* wrong, rather it is a matter of inadequacy with respect to the way we use the mathematics for the operational description of quantum theory.

Suppose we have a state ρ of system A and a channel $\mathcal{M}$ from system C to system D:

$$\rho \in \mathcal{L}(\mathcal{H}_A) \quad \mathcal{M} \in \mathcal{L}(\mathcal{L}(\mathcal{H}_C), \mathcal{L}(\mathcal{H}_D)) \quad (28)$$

In accordance with (22), the tensor product of ρ and $\mathcal{M}$ is a linear map

$$\rho \otimes \mathcal{M} \in \mathcal{L}(\mathcal{H}_A \otimes \mathcal{L}(\mathcal{H}_C), \mathcal{H}_A \otimes \mathcal{L}(\mathcal{H}_D)). \quad (29)$$

Now something has gone wrong, because neither of the Hilbert spaces $\mathcal{H}_A \otimes \mathcal{L}(\mathcal{H}_C)$ or $\mathcal{H}_A \otimes \mathcal{L}(\mathcal{H}_D)$ has a clear operational meaning for the way we model quantum theory. $\mathcal{H}_A$ is a Hilbert space we have chosen to model system A, but we do that by defining states of system A to be a subset of the space of linear operators $\mathcal{L}(\mathcal{H}_A)$. Even if we decided to view a (non-zero) $v \in \mathcal{H}_A$ as a representative of a pure state of system A, there is no operational meaning to the output $\rho(v) \otimes \mathcal{M}(\sigma)$ that results from applying $\rho \otimes \mathcal{M}$ to a product input $v \otimes \sigma$, where σ is a state of system C. Ultimately, there is no kind of quantum object that $\rho \otimes \mathcal{M}$ could be understood as modeling.

The map $\rho \otimes \mathcal{M}$ may be unsuitable for our purposes, but there is another way to use the tensor product with ρ and $\mathcal{M}$ such that the result is always a legitimate channel. This is the channel that will be formally defined as $\rho \boxtimes \mathcal{M}$ when the full $\boxtimes$

operation is defined later in this section. The idea of $\rho \boxtimes \mathcal{M}$ is simple: since $\mathcal{M}$ takes a state as input and returns a state as output, and since ρ takes no input, we can define $\rho \boxtimes \mathcal{M}$ as a map that takes an input $\sigma \in \mathcal{L}(\mathcal{H}_C)$ and passes it to $\mathcal{M}$, appends the resulting output $\mathcal{M}(\sigma)$ to ρ using the tensor product, and therefore returns $\rho \otimes \mathcal{M}(\sigma)$ as the final output. It is straightforward to prove that this results in a legitimate linear map (moreover, a channel)

$$\rho \boxtimes \mathcal{M} \in \mathcal{L}(\mathcal{L}(\mathcal{H}_C), \mathcal{L}(\mathcal{H}_A \otimes \mathcal{H}_D)) \quad (30)$$

which acts according to the formula

$$\forall \sigma \in \mathcal{L}(\mathcal{H}_C), (\rho \boxtimes \mathcal{M})(\sigma) = \rho \otimes \mathcal{M}(\sigma). \quad (31)$$

Note that according to (30), $\rho \boxtimes \mathcal{M}$ is of type $C \rightarrow AD$.

3.2 Defining the generalized parallel product

In defining a general parallel product of typed linear maps, we are guided by several specific insights from the prior discussion. We saw that the usual tensor product is perfectly appropriate in its role of describing composite systems, as well as in its role as a parallel product operation for pairs of states and pairs of channels, and all this must be precisely preserved in our generalized parallel product. From this we glean the following desiderata for the parallel product operation on types:

- (PT1) For elementary types $A, B \in \mathbb{A}$, we must ensure that $A \boxtimes B = AB$.
- (PT2) For map types of order one (which describe channels) $A \rightarrow B$ and $C \rightarrow D$, we must ensure that $(A \rightarrow B) \boxtimes (C \rightarrow D) = AC \rightarrow BD$.

We also saw that the generalized parallel product must act differently from the usual parallel product when combining a state with a channel. In particular:

- (PT3) For an elementary type $A \in \mathbb{A}$ and a map type of order one $C \rightarrow D$, we must ensure that $A \boxtimes (C \rightarrow D) = C \rightarrow AD$.

However, we can rewrite desiderata (PT2) and (PT3) in light of desideratum (PT1), as follows:

- (PT2') For map types of order one (which describe channels) $A \rightarrow B$ and $C \rightarrow D$, we must ensure that $(A \rightarrow B) \boxtimes (C \rightarrow D) = A \boxtimes C \rightarrow B \boxtimes D$.
- (PT3') For an elementary type $A \in \mathbb{A}$ and a map type of order one $C \rightarrow D$, we must ensure that $A \boxtimes (C \rightarrow D) = C \rightarrow A \boxtimes D$.

In order to fulfill the above desiderata, the key decision is to mimic the formula $(A \rightarrow B) \boxtimes (C \rightarrow D) = A \boxtimes C \rightarrow B \boxtimes D$ for any two types of *equal order*, and to mimic the formula $A \boxtimes (C \rightarrow D) = C \rightarrow A \boxtimes D$ for any two types of *unequal order*.

Definition 3.1 (Parallel product type). Let $x, y \in \mathbf{Types}_{\mathbb{A}}$. The type $x \boxtimes y \in \mathbf{Types}_{\mathbb{A}}$ is defined as:

$$x \boxtimes y = \begin{cases} A_1 \cdots A_j B_1 \cdots B_k & \text{if } x = A_1 \cdots A_j \text{ and } y = B_1 \cdots B_k \\ c \rightarrow x \boxtimes d & \text{if } y = c \rightarrow d \text{ and } \text{ord}(x) < \text{ord}(y) \\ a \boxtimes c \rightarrow b \boxtimes d & \text{if } x = a \rightarrow b, y = c \rightarrow d, \text{ and } \text{ord}(x) = \text{ord}(y) \\ a \rightarrow b \boxtimes y & \text{if } x = a \rightarrow b \text{ and } \text{ord}(x) > \text{ord}(y) \end{cases} \quad (32)$$

Note that we write $A_1 \cdots A_j B_1 \cdots B_k$ to denote the concatenation of the sequences $A_1 \cdots A_j$ and $B_1 \cdots B_k$, which means in particular that $IA = A = AI$, as I denotes the empty sequence.

The inductive form of Definition 3.1 is such that each time an expression $w \boxtimes z$ is used therein to help define $x \boxtimes y$, the operands w and z are both of lower order than the maximum order of x and y . Moreover, we have a valid base case for when $\text{ord}(x) = \text{ord}(y) = 0$. Thus, the definition works by induction on the number $\max\{\text{ord}(x), \text{ord}(y)\}$.

For the parallel product operation on *maps*, the discussion of Section 3.1 leads us to the following desiderata:

- (PM1) For elementary-typed maps $\rho : A$ and $\sigma : B$, we must ensure that $\rho \boxtimes \sigma = \rho \otimes \sigma$.
- (PM2) For maps of order one $\mathcal{M} : A \rightarrow B$ and $\mathcal{N} : C \rightarrow D$, we must ensure that the parallel product map $\mathcal{M} \boxtimes \mathcal{N}$ acts according to the formula $\mathcal{M} \boxtimes \mathcal{N}(\rho \otimes \sigma) = \mathcal{M}(\rho) \otimes \mathcal{N}(\sigma)$.
- (PM3) For a elementary-type map $\rho : A$ and a map of order one $\mathcal{M} : C \rightarrow D$, we want $\rho \boxtimes \mathcal{M}$ to act according to the formula $(\rho \boxtimes \mathcal{M})(\sigma) = \rho \otimes \mathcal{M}(\sigma)$.

But again, we rewrite desiderata (PM2) and (PM3) in light of desideratum (PM1), as follows:

- (PM2') For maps of order one $\mathcal{M} : A \rightarrow B$ and $\mathcal{N} : C \rightarrow D$, we must ensure that the parallel product map $\mathcal{M} \boxtimes \mathcal{N}$ acts according to the formula $\mathcal{M} \boxtimes \mathcal{N}(\rho \otimes \sigma) = \mathcal{M}(\rho) \boxtimes \mathcal{N}(\sigma)$.
- (PM3') For a elementary-type map $\rho : A$ and a map of order one $\mathcal{M} : C \rightarrow D$, we want $\rho \boxtimes \mathcal{M}$ to act according to the formula $(\rho \boxtimes \mathcal{M})(\sigma) = \rho \boxtimes \mathcal{M}(\sigma)$.

Generalizing much in the same way we did for the parallel product of types, we choose to mimic the formula $(\mathcal{M} \boxtimes \mathcal{N})(\rho \otimes \sigma) = \mathcal{M}(\rho) \otimes \mathcal{N}(\sigma)$ for any two maps $\mathcal{M}$ and $\mathcal{N}$ of *equal order*, and to mimic the formula $(\rho \boxtimes \mathcal{N})(\sigma) = \rho \otimes \mathcal{N}(\sigma)$ for any two maps ρ and $\mathcal{N}$ of *unequal order* (specifically, when $\mathcal{N}$ is of greater order than ρ).

Definition 3.2 (Parallel product map). Let $\mathcal{M} : x$ and $\mathcal{N} : y$ for $x, y \in \mathbf{Types}_{\mathbb{A}}$. The map $\mathcal{M} \boxtimes \mathcal{N} : x \boxtimes y$ is defined according to the following formulae: for all $\rho : a$ and for

all $\sigma : c$

$$\mathcal{M} \boxtimes \mathcal{N} := \begin{cases} \mathcal{M} \otimes \mathcal{N} & \text{if } x, y \in \mathbf{EleTypes}_{\mathbb{A}} \\ (\mathcal{M} \boxtimes \mathcal{N})(\sigma) = \mathcal{M} \boxtimes \mathcal{N}(\sigma) & \text{if } y = c \rightarrow d \text{ and } \text{ord}(x) < \text{ord}(y) \\ (\mathcal{M} \boxtimes \mathcal{N})(\rho \boxtimes \sigma) = \mathcal{M}(\rho) \boxtimes \mathcal{N}(\sigma) & \text{if } x = a \rightarrow b, y = c \rightarrow d, \text{ and } \text{ord}(x) = \text{ord}(y) \\ (\mathcal{M} \boxtimes \mathcal{N})(\rho) = \mathcal{M}(\rho) \boxtimes \mathcal{N} & \text{if } x = a \rightarrow b \text{ and } \text{ord}(x) > \text{ord}(y) \end{cases} \quad (33)$$

It is non-trivial, but we are able to prove that the latter specification gives a well-defined linear map $\mathcal{M} \boxtimes \mathcal{N} : x \boxtimes y$ in all cases. The main point of interest in this regard is the formula given in the case where $x = a \rightarrow b$, $y = c \rightarrow d$, and $\text{ord}(x) = \text{ord}(y)$. In order for this formula to specify a definite linear map, it must be by linear extension to the full domain $\mathbf{L}(a \boxtimes c)$. What must be shown then is that the subset of elements of the form $\rho \boxtimes \sigma$ is a spanning subset of $\mathbf{L}(a \boxtimes c)$. This is indeed the case, as we demonstrate in our full results which may be found in [52–54].

Inspecting the three cases of (33) for non-elementary maps, each case is defined according to a simple formula. If $\mathcal{M}$ and $\mathcal{N}$ are of equal order (the “symmetric case”), the input of $\mathcal{M} \boxtimes \mathcal{N}$ is factored and the left and right factors are passed to $\mathcal{M}$ and $\mathcal{N}$, respectively. If $\mathcal{M}$ and $\mathcal{N}$ are of unequal order (the “asymmetric case(s)”), the input of $\mathcal{M} \boxtimes \mathcal{N}$ is passed directly to the map of higher order, and the map of lower order is simply appended as part of the output. However, for a complete understanding of the parallel product operation, we must be able to compute the parallel product *type* as well, and we now present a series of examples to illustrate how this works.

First, in the following example, we step through Definition 3.1 and confirm that it fulfills desideratum (PT2).

Example 3.3. Let $x = A \rightarrow B$ and $y = C \rightarrow D$. x and y are both of order one, and so according to (33) we must use the symmetric case of the parallel product:

$$x \boxtimes y = \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ A \quad B \end{array} \boxtimes \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ C \quad D \end{array} = \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ A \boxtimes C \quad B \boxtimes D \end{array} \quad (34)$$

For elementary types, $\boxtimes$ is computed by concatenation, and so we have $A \boxtimes C = AC$ and $B \boxtimes D = BD$. Therefore, the final result is

$$x \boxtimes y = \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ AC \quad BD \end{array} \quad (35)$$

In general, to apply Definition 3.1 to compute $x \boxtimes y$ involves several steps: first compare the orders of x and y , and depending on the result we recurse (compute $\boxtimes$ again) onto subtrees of x and y . Here we see an example where computing $x \boxtimes y$ involves alternating invocations of both the symmetric and asymmetric cases.

Example 3.4. Let $x = A \rightarrow (B \rightarrow C)$ and $y = (D \rightarrow E) \rightarrow F$. Note that $\text{st}(x) \neq \text{st}(y)$, that is,

$$\begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} \neq \begin{array}{c} \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \\ / \quad \backslash \\ \bullet \quad \bullet \end{array} \quad (36)$$

x and y are both of order two, so the first step is symmetric:

$$\begin{array}{c} \rightarrow \\ / \quad \backslash \\ A \quad \rightarrow \\ / \quad \backslash \\ B \quad C \end{array} \boxtimes \begin{array}{c} \rightarrow \\ / \quad \backslash \\ \rightarrow \quad F \\ / \quad \backslash \\ D \quad E \end{array} = \begin{array}{c} \rightarrow \\ / \quad \backslash \\ A \boxtimes (D \rightarrow E) \quad (B \rightarrow C) \boxtimes F \end{array} \quad (37)$$

Then we compute the recursive applications of $\boxtimes$ demanded by the right-hand side of (37):

$$A \boxtimes \begin{array}{c} \rightarrow \\ / \quad \backslash \\ D \quad E \end{array} = \begin{array}{c} \rightarrow \\ / \quad \backslash \\ D \quad A \boxtimes E \end{array} = \begin{array}{c} \rightarrow \\ / \quad \backslash \\ D \quad AE \end{array} \quad (38)$$

$$\begin{array}{c} \rightarrow \\ / \quad \backslash \\ B \quad C \end{array} \boxtimes F = \begin{array}{c} \rightarrow \\ / \quad \backslash \\ B \quad C \boxtimes F \end{array} = \begin{array}{c} \rightarrow \\ / \quad \backslash \\ B \quad CF \end{array} \quad (39)$$

Plugging these results back into (37), the final result is:

$$x \boxtimes y = \begin{array}{c} \rightarrow \\ / \quad \backslash \\ \rightarrow \quad \rightarrow \\ / \quad \backslash \quad / \quad \backslash \\ D \quad AE \quad B \quad CF \end{array} \quad (40)$$

Finally, we introduce by example an alternative way to compute the parallel product type (for more details, see [52–54]), by splitting the process into two separate phases. We revisit the types of Example 3.4.

Example 3.5. Let $x = A \rightarrow (B \rightarrow C)$ and $y = (D \rightarrow E) \rightarrow F$. Graphically, we have:

$$x = \begin{array}{c} \rightarrow \\ / \quad \backslash \\ A \quad \rightarrow \\ / \quad \backslash \\ B \quad C \end{array} \quad y = \begin{array}{c} \rightarrow \\ / \quad \backslash \\ \rightarrow \quad F \\ / \quad \backslash \\ D \quad E \end{array} \quad (41)$$

The first phase consists in modifying x and y , by augmenting their trees with new vertices labeled by the trivial type I (each one placed under a new arrow-labeled vertex), until (i) the modified forms of x and y are of equal order, and (ii) every pair

of matching subtrees, of the modified forms of x and y , are of equal order. In addition, we must only add new I-labeled vertices *on the left*.

Now, proceeding with x and y : x and y are the same order, and thus we need not change anything at this level. So, we pass to the subtrees of x and y . First the left subtrees: we compare A with $D \rightarrow E$ and see that A is of lesser order. So, we change A into $I \rightarrow A$ within x :

$$\begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ A \quad \rightarrow \\ \swarrow \quad \searrow \\ B \quad C \end{array} \rightsquigarrow \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ \rightarrow \quad \rightarrow \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ I \quad A \quad B \quad C \end{array} \quad (42)$$

Now the right subtrees of x and y : we compare $B \rightarrow C$ with F and see that F is of lesser order. So, we change F into $I \rightarrow F$:

$$\begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ \rightarrow \quad F \\ \swarrow \quad \searrow \\ D \quad E \end{array} \rightsquigarrow \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ \rightarrow \quad \rightarrow \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ D \quad E \quad I \quad F \end{array} \quad (43)$$

Now, on the right-hand sides of (42) and (43), we see modified forms of x and y that meet the requirements (i) and (ii) described above. Let us call these x' and y' , so that $x' = (I \rightarrow A) \rightarrow (B \rightarrow C)$ and $y' = (D \rightarrow E) \rightarrow (I \rightarrow F)$. In fact, there is another way to verify that (i) and (ii) are satisfied, namely by the fact that x' and y' have the same structure (i.e. the same tree shape).

Now that we have the correct trees x' and y' , the final step is simply to compute the parallel product type $x' \boxtimes y'$. However, computing the parallel product type is especially simple for a pair of types that have the same structure. One may compute $x' \boxtimes y'$ simply by overlaying the two trees, one over the other, and combining the matching labels together:

$$\begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ \rightarrow \quad \rightarrow \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ I \quad A \quad B \quad C \end{array} \boxtimes \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ \rightarrow \quad \rightarrow \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ D \quad E \quad I \quad F \end{array} = \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ \rightarrow \quad \rightarrow \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ I \boxtimes D \quad A \boxtimes E \quad B \boxtimes I \quad C \boxtimes F \end{array} \quad (44)$$

$$= \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ \rightarrow \quad \rightarrow \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ D \quad AE \quad B \quad CF \end{array} \quad (45)$$

As we can see, the final result on the right-hand side of (45) is exactly the same as the final result of Example 3.4, which is shown on the right-hand side of (40).

Remark 3.6 (Comparison with prior work). As we noted in the introduction, the parallel product is a generalization of the notion of “extended event” defined in [39, Definition 4.3]. Using extension as they define it, one arbitrary type $x \in \mathbf{Types}_{\mathbb{A}}$ can be combined with one elementary type $E \in \mathbb{A}$. In the elementary case, $x = A_1 \cdots A_n$, the extension, denoted $x \parallel E$, is computed by concatenation just like the parallel product: $x \parallel E = A_1 \cdots A_n E$. In the non-elementary case, $x = a \rightarrow b$, the extension is defined exactly how we define the parallel product: $x \parallel E = a \rightarrow b \parallel E$. Thus, by the definitions presented in this section, we extend this operation on types to pairs of arbitrary types, and we also provide a corresponding operation for arbitrary maps.

4 Generalizing completely positive maps

The main task of this section is to define the K family of convex cones, whose purpose is to generalize the established notion of complete positivity to higher-order linear maps of all types in our framework. The key ingredient of the construction is to define $K(a \rightarrow b)$ inductively as the subset of “completely K -preserving” maps within $L(a \rightarrow b)$. Notably, the definition of the K family is stated in terms of the parallel product operation of Section 3.

4.1 Defining the K family

In quantum theory, complete positivity arises as one of the properties that characterize channels as a subset of the linear maps $\mathcal{L}(\mathcal{L}(\mathcal{H}_A), \mathcal{L}(\mathcal{H}_B))$, for some given input and output systems A and B , respectively. Channels are required to send states to states, and this entails that they must send positive semidefinite input operators to positive semidefinite output operators. The reason channels are required to be *completely* positive is that a map $\mathcal{M} \in \mathcal{L}(\mathcal{L}(\mathcal{H}_A), \mathcal{L}(\mathcal{H}_B))$ is *not* completely positive precisely when there exists a bipartite input state that, when acted upon by $\mathcal{M}$ on one subsystem, is sent to an output operator that is not positive semidefinite and hence not a valid state. Such an $\mathcal{M}$ is therefore unsuitable for representing a physical process. Indeed, imagine the input system of $\mathcal{M}$ is entangled with another system C , unbeknownst to the experimenter. We still want the overall output to be a valid state. The insistence on considering the local action of a transformation is particularly important for a good theory of quantum information, given the notorious difficulty of isolating a quantum system from its environment, even under ideal laboratory conditions.

To recall the precise definition, a map $\mathcal{M} \in \mathcal{L}(\mathcal{L}(\mathcal{H}_A), \mathcal{L}(\mathcal{H}_B))$ is *completely positive* if for all finite-dimensional Hilbert spaces $\mathcal{H}_C$ and all $\rho \in \text{Pos}(\mathcal{H}_A \otimes \mathcal{H}_C)$, we have

$$(\mathcal{M} \otimes \mathcal{I}_{C \rightarrow C})(\rho) \in \text{Pos}(\mathcal{H}_B \otimes \mathcal{H}_C). \quad (46)$$

Note that we write $\text{Pos}(\mathcal{H}_B \otimes \mathcal{H}_C)$ for the subset of positive semidefinite operators within $\mathcal{L}(\mathcal{H}_B \otimes \mathcal{H}_C)$, and that we write $\mathcal{I}_{C \rightarrow C}$ for the identity map of type $C \rightarrow C$.

In [17, 18, 50], the authors extended the same logic to the analysis of superchannels. They argued that a superchannel must be completely CP-preserving (or, “CCPP”) because, otherwise, such a map would send some valid bipartite input channel to a non-CP (thus invalid) output when applied to only one part. The same

logic can be extended yet another step upward, resulting in the notion of completely CCPP-preserving (or, “CCCPPP”) maps, and yet another step, resulting in completely CCCPPP-preserving maps, and so on indefinitely. We introduce the K family to define all these properties with a single inductive construction, and in a way that avoids the problem of exploding terminology exemplified by the terms “completely CP-preserving,” “completely CCPP-preserving,” etc.

As we now see, the definition of the K family is stated with a straightforward inductive formula. For elementary types x , $K(x)$ is the subset of positive semidefinite operators $\text{Pos}(\mathcal{H}_x) \subset \mathcal{L}(\mathcal{H}_x)$ (*caveat*, note the remark about the trivial type below). For a non-elementary type $x = a \rightarrow b$, $K(a \rightarrow b)$ is the subset of *completely* K -preserving maps. Note that for $z \in \text{Types}_{\mathbb{A}}$, we write $\mathcal{I}_{L(z)}$ for the identity map on $L(z)$, and that $\mathcal{I}_{L(z)}$ is of type $z \rightarrow z$.

Definition 4.1. Let $x \in \text{Types}_{\mathbb{A}}$ and $\mathcal{M} : x$. Let $K(x) \subset L(x)$ be defined by the following logical equivalence:

$$\mathcal{M} \in K(x) \iff \begin{cases} \mathcal{M} \in \text{Pos}(L(x)) & \text{if } x \in \text{EleTypes}_{\mathbb{A}} \\ \mathcal{M} \text{ is completely } K\text{-preserving} & \text{if } x = a \rightarrow b \end{cases}, \quad (47)$$

where a map $\mathcal{M} : a \rightarrow b$ is called *completely* K -preserving if

$$(\mathcal{M} \boxtimes \mathcal{I}_{z \rightarrow z})(\rho) \in K(b \boxtimes z) \quad (48)$$

for all $z \in \text{Types}_{\mathbb{A}}$ such that $\text{ord}(z \rightarrow z) = \text{ord}(x)$ and for all $\rho \in K(a \boxtimes z)$.

Note that in the case of the trivial type, we write $\text{Pos}(L(I))$ for the subset $\mathbb{R}_{\geq 0} \subset \mathbb{C}$ of positive real numbers.

Let us discuss the validity of Definition 4.1 as an inductive construction. In the case of $x = a \rightarrow b$, the definition of $K(x)$ depends on the definition of $K(a \boxtimes z)$ and $K(b \boxtimes z)$ for every type z satisfying a certain condition. Crucially, the assumptions guarantee that $a \boxtimes z$ and $b \boxtimes z$ are necessarily of order strictly less than the order of x , and consequently, the construction is valid by induction on the order of x . To conclude this discussion, we mention how to prove that $\text{ord}(a \boxtimes z)$ and $\text{ord}(b \boxtimes z)$ are strictly less than $\text{ord}(x)$. There is a general fact that the order $\text{ord}(y \boxtimes y')$ of a parallel product type is equal to the maximum of the orders of the operands, i.e. we have $\text{ord}(y \boxtimes y') = \max\{\text{ord}(y), \text{ord}(y')\}$. This fact is proven in our fuller result, which can be found in [52–54]. Therefore, given $\text{ord}(z \rightarrow z) = \text{ord}(x)$ we can deduce that $\text{ord}(x \boxtimes (z \rightarrow z)) = \text{ord}(x)$. It follows immediately that both $\text{in}(x \boxtimes (z \rightarrow z)) = a \boxtimes z$ and $\text{out}(x \boxtimes (z \rightarrow z)) = b \boxtimes z$ are of order strictly less than x , because by definition we have

$$\text{ord}(x \boxtimes (z \rightarrow z)) = 1 + \max\{\text{ord}(\text{in}(x \boxtimes (z \rightarrow z))), \text{ord}(\text{out}(x \boxtimes (z \rightarrow z)))\}. \quad (49)$$

According to Definition 4.1, checking if a map is completely K -preserving looks like a daunting task. However, thanks to the Choi isomorphism introduced in [52–54], we

can perform this check fairly easily, as is discussed in greater detail therein (see also Section 4.2).

4.2 Properties of the $\mathbf{K}$ family

In [54] and in forthcoming work [52, 53], we consider several properties of the $\mathbf{K}$ family, which fall into two categories. On the one hand, algebraic properties of completely $\mathbf{K}$ -preserving maps, including e.g. the fact that completely $\mathbf{K}$ -preserving maps are closed under composition, as well as the fact that the parallel product of two completely $\mathbf{K}$ -preserving maps is again completely $\mathbf{K}$ -preserving.

On the other hand, we show that $\mathbf{K}(x)$ forms a convex cone for each $x \in \mathbf{Types}_{\mathbb{A}}$, and we examine its properties as a cone. Recall that a convex cone in a real vector space V is a subset $\Gamma \subset V$ which is closed under non-negative linear combinations. For instance, for any finite-dimensional Hilbert space $\mathcal{H}_{\mathbb{A}}$, the Hermitian operators $\text{Herm}(\mathcal{H}_{\mathbb{A}})$ form a real vector space, and the set of positive semidefinite operators $\text{Pos}(\mathcal{H}_{\mathbb{A}}) \subset \text{Herm}(\mathcal{H}_{\mathbb{A}})$ is a convex cone.

To show that $\mathbf{K}(x)$ is a convex cone, we need to situate it inside a suitable real vector space. For this, we appeal to a family of real vector spaces $\mathbf{H}(x) \subset \mathbf{L}(x)$, for each $x \in \mathbf{Types}_{\mathbb{A}}$, which is defined in a way quite similar to the $\mathbf{K}$ family. The difference is that for elementary types x , $\mathbf{H}(x) \subset \mathbf{L}(x)$ is the subset of Hermitian operators, and for a non-elementary type $x = a \rightarrow b$, $\mathbf{H}(x) \subset \mathbf{L}(x)$ is the subset of $\mathbf{H}$ -preserving maps.

Definition 4.2. Let $x \in \mathbf{Types}_{\mathbb{A}}$ and $\mathcal{M} : x$. Let $\mathbf{H}(x) \subset \mathbf{L}(x)$ be defined by the following logical equivalence:

$$\mathcal{M} \in \mathbf{H}(x) \iff \begin{cases} \mathcal{M} \in \text{Herm}(\mathbf{L}(x)) & \text{if } x \in \mathbf{EleTypes}_{\mathbb{A}} \\ \mathcal{M} \text{ is } \mathbf{H}\text{-preserving} & \text{if } x = a \rightarrow b \end{cases}, \quad (50)$$

where a map $\mathcal{M} : a \rightarrow b$ is called *$\mathbf{H}$ -preserving* if $\mathcal{M}(\rho) \in \mathbf{H}(b)$ for all $\rho \in \mathbf{H}(a)$.

We omit the modifier “completely” not because we are deemphasizing local application of transformations, but because in the case of the $\mathbf{H}$ family, we are able to show that there is a logical equivalence between the notions of $\mathbf{H}$ -preserving map and completely $\mathbf{H}$ -preserving map [52–54].

Note that in the case of the trivial type, we write $\text{Herm}(\mathbf{L}(\mathbf{I}))$ for the subset $\mathbb{R} \subset \mathbb{C}$ of real numbers.

Having defined the $\mathbf{H}$ family, we are able to demonstrate several facts, which are presented in [52–54]. For each $x \in \mathbf{Types}_{\mathbb{A}}$, $\mathbf{H}(x)$ is a real vector space, $\mathbf{K}(x)$ is a subset of $\mathbf{H}(x)$, and $\mathbf{K}(x)$ is closed under nonnegative linear combinations. Hence, $\mathbf{K}(x)$ is a convex cone in $\mathbf{H}(x)$. Moreover, by defining an appropriate form of the Choi isomorphism in our framework, we are able to show that there is an isomorphism of convex cones between each $\mathbf{K}(x)$ and a cone of positive semidefinite operators $\text{Pos}(\mathcal{H})$, for a suitably chosen finite-dimensional Hilbert space $\mathcal{H}$. Since these cones are isomorphic, properties of $\text{Pos}(\mathcal{H})$ transfer to $\mathbf{K}(x)$. For instance, the property that $\text{Pos}(\mathcal{H})$ spans $\mathcal{L}(\mathcal{H})$ translates to the fact that $\mathbf{K}(x)$ spans $\mathbf{H}(x)$, which is to say that for every

$\mathcal{M} \in \mathbf{H}(x)$, there exist maps $\mathcal{M}_+, \mathcal{M}_- \in \mathbf{K}(x)$ such that $\mathcal{M} = \mathcal{M}_+ - \mathcal{M}_-$. The property of self-duality, which plays a useful role in conic linear programs for optimization, also transfers to $\mathbf{K}(x)$ from $\mathbf{Pos}(\mathcal{H})$.

We therefore show that the $\mathbf{K}$ family, a generalization of the positive semidefinite cone to higher orders, preserves the geometric properties of such a cone to all orders. This family is also a crucial step towards characterizing the physically meaningful maps at any order. The other aspects of such a characterization, e.g. the higher-order generalization of trace-preservation, will be examined in future work.

5 Conclusions

We have introduced a framework for higher-order quantum theory in the finite-dimensional case. The basic components of the framework are a simple type system for describing different kinds of higher-order maps and a Hilbert space of linear maps for each type. These spaces include all forms of higher-order quantum protocols. In addition to the usual sequential composition of linear maps, we have equipped the framework with a parallel compositional structure in the form of the parallel product operation. From this we gain an operation that fulfills in full generality the role that the tensor product plays for pairs of states and pairs of channels, namely providing an operationally meaningful notion of parallel combination of maps.

The inductive constructions at the heart of our framework, which are made possible by its type system, provide us with economical notation and terminology in a context where economy is much needed, due to the infinitely many different types of object that higher-order quantum theory inevitably involves. For instance, we can write $\mathbf{L}((A \rightarrow B) \rightarrow (C \rightarrow D))$ instead of a cumbersome expression such as $\mathcal{L}(\mathcal{L}(\mathcal{L}(\mathcal{H}_A), \mathcal{L}(\mathcal{H}_B)), \mathcal{L}(\mathcal{L}(\mathcal{H}_C), \mathcal{L}(\mathcal{H}_D)))$, and we can say “completely $\mathbf{K}$ -preserving maps” instead of “completely CP-preserving maps,” “completely CCPP-preserving maps,” etc. The type system also facilitates the use of powerful inductive constructions that are based on simple principles, such as when we defined the parallel product operations for types and maps.

The definitive point of departure of our framework in comparison to previous work [39, 41] is the methodological decision to avoid using the Choi representation to define the framework’s basic objects. However, because the Choi representation derives from certain linear maps between Hilbert spaces, it can be defined within our framework (for details, see [52–54]). By introducing the Choi representation itself from our framework, rather than incorporating it into the formalism at the very foundation, we can avoid notational ambiguities that can arise when using Choi matrices in calculations, as noted in [49]. And because the Choi representation is used only where it is strictly pertinent, it promotes the “decoupling” of the study of higher-order quantum theory from the Choi formalism, paving the way to an extension to the infinite-dimensional case.

After setting up the framework’s methodological foundation, we must take up the characterization of the physically meaningful objects as a subset of $\mathbf{L}(x)$ for each $x \in \mathbf{Types}_{\mathbb{A}}$. In this work, we addressed the aspect of complete positivity. Thanks to

the parallel product operation, the K family can be constructed using a straightforward inductive formulation of the notion of “completely K -preserving map,” which generalizes the notion of complete positivity. Despite the generalization, it remains the case that $K(x)$ is a convex cone, and is furthermore isomorphic to a cone of positive semidefinite operators, thus sharing its many nice properties. Consequently, many of the same convex optimization techniques that can be used to solve problems concerning states and channels [55, 56] are applicable to these higher-order maps as well. The construction of the K family also represents a first test of the suitability of the parallel product operation to occupy a basic role in the mathematics of higher-order quantum theory.

It remains for future work to complete the full characterization of the physically meaningful objects, which should form a subset of $K(x)$ for each type x . In keeping with the precedent set by previous works, [39, 41], the next step would be to develop an appropriate definition of *deterministic* map for all types. The full higher-order definition would generalize states, channels, and superchannels, which are the deterministic maps in the familiar low-order cases. One approach, similar in spirit to [39] and [41], would be to define a general deterministic transformation inductively, by the requirement to send deterministic input maps to deterministic output maps, potentially in a complete sense with respect to our parallel product.

Once we have a suitable characterization of the physically meaningful maps, we intend to use our framework to study quantum processes that exhibit indefinite causal order [27, 33–36]. Such processes should be included by a permissive characterization of physically meaningful maps. We aim to investigate whether processes with indefinite causal order are associated with any special signature within our framework. Another direction we intend to pursue is the extension of quantum information techniques to higher-order transformations, such as the generalization of the notion of channel entropy [18, 57–60]. This problem could be approached one level at a time, first passing from channels to superchannels, but if an adequate inductive characterization can be found then it could be solved at once for all orders. Another potential area of application, and one of the main motivations for this work, is quantum resource theories [10, 11]. In extant work, the resources studied are either static (states) or dynamical (channels), but it has been shown that higher-order processes like the SWITCH can also convey a quantum advantage in information processing [28–32, 37]. To expand the scope to include all higher-order processes would be to open a broad new domain for the general theory of quantum resources.

Acknowledgments

E. Z. acknowledges support from the Eric Milner’s Graduate Scholarship, the Alberta Graduate Excellence Scholarship (AGES), and the Eyes High International Doctoral Scholarship. C. M. S. acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Discovery Grant “The power of quantum resources” RGPIN-2022-03025 and the Discovery Launch Supplement DGEER-2022-00119.

References

- [1] Knight, P., Walmsley, I.: UK national quantum technology programme. *Quantum Sci. Technol.* **4**(4), 040502 (2019) <https://doi.org/10.1088/2058-9565/ab4346>
- [2] Raymer, M.G., Monroe, C.: The US National Quantum Initiative. *Quantum Sci. Technol.* **4**(2), 020504 (2019) <https://doi.org/10.1088/2058-9565/ab0441>
- [3] Sussman, B., Corkum, P., Blais, A., Cory, D., Damascelli, A.: Quantum Canada. *Quantum Sci. Technol.* **4**(2), 020503 (2019) <https://doi.org/10.1088/2058-9565/ab029d>
- [4] Koch, C.P., Boscain, U., Calarco, T., Dirr, G., Filipp, S., Glaser, S.J., Kosloff, R., Montangero, S., Schulte-Herbrüggen, T., Sugny, D., Wilhelm, F.K.: Quantum optimal control in quantum technologies. Strategic report on current status, visions and goals for research in Europe. *EPJ Quantum Technol.* **9**(1), 19 (2022) <https://doi.org/10.1140/epjqt/s40507-022-00138-x>
- [5] Dowling, J.P., Milburn, G.J.: Quantum technology: The second quantum revolution. *Philos. Trans. R. Soc. A*, 1655–1674 (2003) <https://doi.org/10.1098/rsta.2003.1227>
- [6] Degen, C.L.: Quantum sensing. *Rev. Mod. Phys.* **89**(3), 035002 (2017) <https://doi.org/10.1103/RevModPhys.89.035002>
- [7] Aaronson, S., Childs, A.M., Farhi, E., Harrow, A.W., Sanders, B.C.: Future of Quantum Computing. arXiv:2506.19232 [quant-ph] (2025). <https://doi.org/10.48550/arXiv.2506.19232>
- [8] Sanders, B.C.: The success and failure of quantum computing start-ups. *Nat. Electron.* **8**(1), 5–7 (2025) <https://doi.org/10.1038/s41928-025-01337-x>
- [9] Sidhu, J.S., Joshi, S.K., Gündoğan, M., Brougham, T., Lowndes, D., Mazzarella, L., Krutzik, M., Mohapatra, S., Dequal, D., Vallone, G., Villoresi, P., Ling, A., Jennewein, T., Mohageg, M., Rarity, J.G., Fuentes, I., Pirandola, S., Oi, D.K.L.: Advances in space quantum communications. *IET Quantum Commun.* **2**(4), 182–217 (2021) <https://doi.org/10.1049/qt2.12015>
- [10] Chitambar, E., Gour, G.: Quantum resource theories. *Rev. Mod. Phys.* **91**(2), 025001 (2019) <https://doi.org/10.1103/RevModPhys.91.025001>
- [11] Gour, G.: Quantum Resource Theories. Cambridge University Press, Cambridge (2025). <https://doi.org/10.1017/9781009560870>
- [12] Theurer, T., Killoran, N., Egloff, D., Plenio, M.B.: Resource Theory of Superposition. *Phys. Rev. Lett.* **119**(23), 230401 (2017) <https://doi.org/10.1103/PhysRevLett.119.230401>

- [13] Streltsov, A., Adesso, G., Plenio, M.B.: *Colloquium: Quantum coherence as a resource*. Rev. Mod. Phys. **89**(4), 041003 (2017) <https://doi.org/10.1103/RevModPhys.89.041003>
- [14] Gottesman, D.: The Heisenberg Representation of Quantum Computers. arXiv quant-ph/9807006 (1998). <https://doi.org/10.48550/arXiv.quant-ph/9807006>
- [15] Veitch, V., Mousavian, S.A.H., Gottesman, D., Emerson, J.: The resource theory of stabilizer quantum computation. New J. Phys. **16**(1), 013009 (2014) <https://doi.org/10.1088/1367-2630/16/1/013009>
- [16] Heyfron, L.E., Campbell, E.T.: An efficient quantum compiler that reduces T count. Quantum Sci. Technol. **4**(1), 015004 (2018) <https://doi.org/10.1088/2058-9565/aad604>
- [17] Chiribella, G., D’Ariano, G.M., Perinotti, P.: Transforming quantum operations: Quantum supermaps. Europhys. Lett. **83**(3), 30004 (2008) <https://doi.org/10.1209/0295-5075/83/30004>
- [18] Gour, G.: Comparison of Quantum Channels by Superchannels. IEEE Trans. Inf. Theory **65**(9), 5880–5904 (2019) <https://doi.org/10.1109/TIT.2019.2907989>
- [19] Gour, G., Scandolo, C.M.: Dynamical Resources. arXiv:2101.01552 [quant-ph] (2020). <https://doi.org/10.48550/arXiv.2101.01552>
- [20] Gour, G., Winter, A.: How to Quantify a Dynamical Quantum Resource. Phys. Rev. Lett. **123**(15), 150401 (2019) <https://doi.org/10.1103/PhysRevLett.123.150401>
- [21] Liu, Z.-W., Winter, A.: Resource Theories of Quantum Channels and the Universal Role of Resource Erasure. arXiv:1904.04201 [quant-ph] (2019). <https://doi.org/10.48550/arXiv.1904.04201>
- [22] Liu, Y., Yuan, X.: Operational resource theory of quantum channels. Phys. Rev. Res. **2**(1), 012035 (2020) <https://doi.org/10.1103/PhysRevResearch.2.012035>
- [23] Yuan, X., Zeng, P., Gao, M., Zhao, Q.: One-Shot Dynamical Resource Theory. arXiv:2012.02781 [quant-ph] (2020). <https://doi.org/10.48550/arXiv.2012.02781>
- [24] Bennett, C.H., Brassard, G., Crépeau, C., Jozsa, R., Peres, A., Wootters, W.K.: Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. Phys. Rev. Lett. **70**(13), 1895–1899 (1993) <https://doi.org/10.1103/PhysRevLett.70.1895>
- [25] Devetak, I., Winter, A.: Distilling Common Randomness From Bipartite Quantum States. IEEE Trans. Inf. Theory **50**, 3183–3196 (2004) <https://doi.org/10.1109/TIT.2004.838115>

- [26] Devetak, I., Harrow, A.W., Winter, A.J.: A Resource Framework for Quantum Shannon Theory. *IEEE Trans. Inf. Theory* **54**(10), 4587–4618 (2008) <https://doi.org/10.1109/TIT.2008.928980>
- [27] Chiribella, G., D’Ariano, G.M., Perinotti, P., Valiron, B.: Quantum computations without definite causal structure. *Phys. Rev. A* **88**(2), 022318 (2013) <https://doi.org/10.1103/PhysRevA.88.022318>
- [28] Colnaghi, T., D’Ariano, G.M., Facchini, S., Perinotti, P.: Quantum computation with programmable connections between gates. *Phys. Lett. A* **376**(45), 2940–2943 (2012) <https://doi.org/10.1016/j.physleta.2012.08.028>
- [29] Chiribella, G.: Perfect discrimination of no-signalling channels via quantum superposition of causal structures. *Phys. Rev. A* **86**(4), 040301 (2012) <https://doi.org/10.1103/PhysRevA.86.040301>
- [30] Chiribella, G., Kristjánsson, H.: Quantum Shannon theory with superpositions of trajectories. *Proc. R. Soc. A* **475**(2225), 20180903 (2019) <https://doi.org/10.1098/rspa.2018.0903>
- [31] Kristjánsson, H., Chiribella, G., Salek, S., Ebler, D., Wilson, M.: Resource theories of communication. *New J. Phys.* **22**(7), 073014 (2020) <https://doi.org/10.1088/1367-2630/ab8ef7>
- [32] Kristjánsson, H.: A second-quantised Shannon theory. PhD thesis, University of Oxford (2022)
- [33] Oreshkov, O., Costa, F., Brukner, Č.: Quantum correlations with no causal order. *Nat. Commun.* **3**(1), 1092 (2012) <https://doi.org/10.1038/ncomms2076>
- [34] Oreshkov, O., Giarmatzi, C.: Causal and causally separable processes. *New J. Phys.* **18**(9), 093020 (2016) <https://doi.org/10.1088/1367-2630/18/9/093020>
- [35] Baumeler, Ä., Wolf, S.: The space of logically consistent classical processes without causal order. *New J. Phys.* **18**(1), 013036 (2016) <https://doi.org/10.1088/1367-2630/18/1/013036>
- [36] Wechs, J., Dourdent, H., Abbott, A.A., Branciard, C.: Quantum Circuits with Classical Versus Quantum Control of Causal Order. *Phys. Rev. X Quantum* **2**(3), 030335 (2021) <https://doi.org/10.1103/PRXQuantum.2.030335>
- [37] Zhao, X., Yang, Y., Chiribella, G.: Quantum Metrology with Indefinite Causal Order. *Phys. Rev. Lett.* **124**(19), 190503 (2020) <https://doi.org/10.1103/PhysRevLett.124.190503>
- [38] Perinotti, P.: Causal Structures and the Classification of Higher Order Quantum Computations. In: Renner, R., Stupar, S. (eds.) *Time in Physics*, pp. 103–127.

- Birkhäuser, Cham (2017). https://doi.org/10.1007/978-3-319-68655-4_7
- [39] Bisio, A., Perinotti, P.: Theoretical framework for higher-order quantum theory. *Proc. R. Soc. A* **475**(2225), 20180706 (2019) <https://doi.org/10.1098/rspa.2018.0706>
 - [40] Apadula, L., Bisio, A., Perinotti, P.: No-signalling constrains quantum computation with indefinite causal structure. *Quantum* **8**, 1241 (2024) <https://doi.org/10.22331/q-2024-02-05-1241> [2202.10214](https://arxiv.org/abs/2202.10214)
 - [41] Kissinger, A., Uijlen, S.: A categorical semantics for causal structure. *Log. Methods Comput. Sci.* **15**(3), 1–48 (2019) [https://doi.org/10.23638/LMCS-15\(3:15\)2019](https://doi.org/10.23638/LMCS-15(3:15)2019)
 - [42] Simmons, W., Kissinger, A.: Higher-Order Causal Theories Are Models of BV-Logic. In: 47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022), pp. 80–18014. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Vienna (2022). <https://doi.org/10.4230/LIPIcs.MFCS.2022.80>
 - [43] Simmons, W., Kissinger, A.: A complete logic for causal consistency. *arXiv:2403.09297 [cs.LO]* (2024). <https://doi.org/10.48550/arXiv.2403.09297>
 - [44] Hefford, J., Wilson, M.: A profunctorial semantics for quantum supermaps. In: *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science. LICS '24*, p. 43. Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3661814.3662123> . <https://doi.org/10.1145/3661814.3662123>
 - [45] Wilson, M.: *Compositional frameworks for supermaps and causality*. PhD thesis, University of Oxford (2023)
 - [46] Wilson, M., Chiribella, G.: Free Polycategories for Unitary Supermaps of Arbitrary Dimension. *arXiv:2207.09180 [quant-ph]* (2022). <https://doi.org/10.48550/arXiv.2207.09180>
 - [47] Wilson, M., Chiribella, G.: Causality in higher order process theories. In: Heunen, C., Backens, M. (eds.) *Proceedings 18th International Conference on Quantum Physics and Logic, Gdansk, Poland, and online, 7-11 June 2021. Electronic Proceedings in Theoretical Computer Science*, vol. 343, pp. 265–300 (2021). <https://doi.org/10.4204/EPTCS.343.12>
 - [48] Wilson, M., Chiribella, G., Kissinger, A.: Quantum Supermaps are Characterized by Locality. *arXiv:2205.09844 [quant-ph]* (2022). <https://doi.org/10.48550/arXiv.2205.09844>
 - [49] Zanoni, E., Scandolo, C.M.: Choi-defined resource theories. *Phys. Rev. A* **111**(6), 062407 (2025) <https://doi.org/10.1103/PhysRevA.111.062407>

- [50] Burniston, J., Grabowecky, M., Scandolo, C.M., Chiribella, G., Gour, G.: Necessary and sufficient conditions on measurements of quantum channels. *Proc. R. Soc. A* **476**(2236), 20190832 (2020) <https://doi.org/10.1098/rspa.2019.0832>
- [51] Avigad, J.: *Mathematical Logic and Computation*. Cambridge University Press, Cambridge (2022). <https://doi.org/10.1017/9781108778756>
- [52] Steakley, S., Zanoni, E., Scandolo, C.M.: *Operational Higher-Order Quantum Theory from Types*. Forthcoming
- [53] Steakley, S.: *A type-based framework for higher-order quantum theory*. Master's thesis, University of Calgary. Forthcoming
- [54] Zanoni, E.: *Maps and Higher-Order Maps for the Manipulation of Quantum Resources*. PhD thesis, University of Calgary (2025)
- [55] Skrzypczyk, P., Cavalcanti, D.: *Semidefinite Programming in Quantum Information Science*. IOP Publishing, Bristol (2023)
- [56] Girard, M.W., Gour, G., Friedland, S.: On convex optimization problems in quantum information theory. *J. Phys. A* **47**(50), 505302 (2014) <https://doi.org/10.1088/1751-8113/47/50/505302>
- [57] Gour, G., Wilde, M.M.: Entropy of a quantum channel. *Phys. Rev. Res.* **3**(2), 023096 (2021) <https://doi.org/10.1103/PhysRevResearch.3.023096>
- [58] Gour, G., Kim, D., Nateeboon, T., Shemesh, G., Yoeli, G.: Inevitable negativity: Additivity commands negative quantum channel entropy. *Phys. Rev. A* **111**(5), 052424 (2025) <https://doi.org/10.1103/PhysRevA.111.052424>
- [59] Yuan, X.: Hypothesis testing and entropies of quantum channels. *Phys. Rev. A* **99**(3), 032317 (2019) <https://doi.org/10.1103/PhysRevA.99.032317>
- [60] Chu, Y., Huang, F., Li, M.-X., Zheng, Z.-J.: An entropy function of a quantum channel. *Quantum. Inf. Process.* **22**(1), 27 (2022) <https://doi.org/10.1007/s11128-022-03778-1>