Tensors, !-graphs, and non-commutative quantum structures

Aleks Kissinger
University of Oxford
aleks.kissinger@cs.ox.ac.uk

David Quick
University of Oxford
david.quick@cs.ox.ac.uk

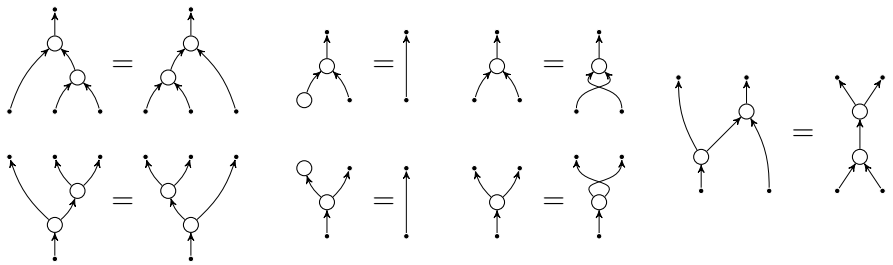
Categorical quantum mechanics (CQM) and the theory of quantum groups rely heavily on the use of structures that have both an algebraic and co-algebraic component, making them well-suited for manipulation using diagrammatic techniques. Diagrams allow us to easily form complex compositions of (co)algebraic structures, and prove their equality via graph rewriting. One of the biggest challenges in going beyond simple rewriting-based proofs is designing a graphical language that is expressive enough to prove interesting properties (e.g. normal form results) about not just single diagrams, but entire families of diagrams. One candidate is the language of *!-graphs*, which consist of graphs with certain subgraphs marked with boxes (called !-boxes) that can be repeated any number of times. New !-graph equations can then be proved using a powerful technique called *!-box induction*. However, previously this technique only applied to commutative (or cocommutative) algebraic structures, severely limiting its applications in some parts of CQM and (especially) quantum groups. In this paper, we fix this shortcoming by offering a new semantics for *non-commutative* !-graphs using an enriched version of Penrose's abstract tensor notation.

1 Introduction

Diagrammatic theories give us a way to study a wide variety of algebraic and coalgebraic structures in monoidal categories. They consist of two parts: a *signature* Σ and a set of *diagram equations* E. The signature consists of a set of objects $\{A,B,\ldots\}$ along with a set of generating morphisms with input and output arities formed from combining objects with $\otimes$ and I. For example, the signature of a Frobenius algebra consists of four morphisms: $(\mu : A\otimes A \to A,\ \eta : I \to A,\ \delta : A \to A\otimes A,\ \varepsilon : A \to I)$, or, written diagrammatically:

$\Sigma =$ 

Then, E is a set of equations between morphisms built from these generators, which we can picture as equations between string diagrams. For example, the theory of commutative Frobenius algebras contains the (co)associativity, (co)unit, (co)commutativity and Frobenius equations:



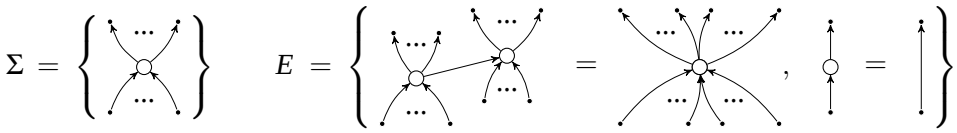
(1)

A *model* of (Σ,E) in a (symmetric, traced, or compact closed) monoidal category $\mathscr{C}$ assigns a morphism to each generator in Σ such that all equations in E hold.

B. Coecke, I. Hasuo & P. Panangaden (Eds.):
Quantum Physics and Logic 2014 (QPL 2014).
EPTCS 172, 2014, pp. 56–67, doi:10.4204/EPTCS.172.5

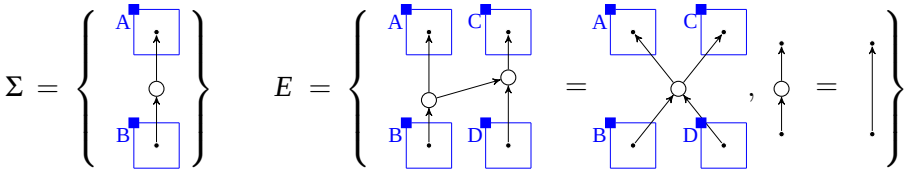
Remark 1.1. Many familiar algebraic constructions arise as special cases of this setup. For instance, any linear 'term-like' algebraic theory (i.e. where free variables occur precisely once on the LHS and RHS of every equation) can be presented this way. Also, if we restrict to equations in E that are directed acyclic, we obtain presentations of PROPs (or coloured PROPs in the multi-sorted case). In that case, models of (Σ, E) in $\mathscr{C}$ are in 1-to-1 correspondence with strong monoidal functors from the presented PROP into $\mathscr{C}$.

This style of algebraic theory works well when generators have fixed, finite arity. However, it is often possible to find a much more elegant presentation of a theory if we allow the arity of our generators to vary. For instance, commutative Frobenius algebras can be alternatively presented using a single variable-arity generator sometimes called a 'spider', along with just two equations.

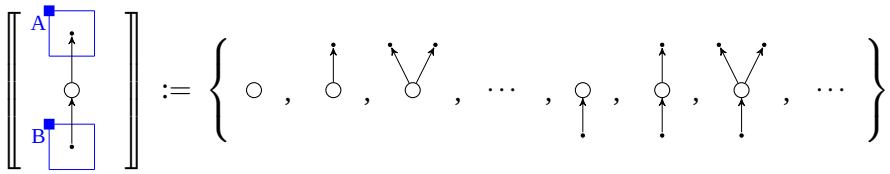


A model of such a theory is no longer just a finite set of morphisms, but rather, a set of *families* of morphisms $f_{j,k} : A^{\otimes j} \to A^{\otimes k}$, indexed by input/output arities, such that the equations in E hold for all possible arities.

Comparing this to the equations at the beginning of this section, we seem to have lost some formality. That is, the 'concrete' diagrammatic identities above can be formalised in such a way that proofs can be performed (and even machine-checked) via a suitable notion of diagram rewriting, as formalised in [2]. One might be tempted to think that this level of rigour is lost when we describe equations in a mathematical meta-language, making use of ellipses, for example, to represent repetition. However, in [1], the authors introduced *!-boxes* (pronounced 'bang-boxes') as a method for reasoning about graphs with repeated structure. As !-box rules, the previously informal rules can be formalised as:



Intuitively, marking a subgraph with a !-box means that subgraph (along with edges in/out of it) can be repeated any number of times to obtain an *instance* of the graph. Thus we interpret a graph with !-boxes as a set of all its instances.



Similarly, for rules with !-boxes, matched pairs of !-boxes can be repeated in the LHS and RHS to obtain instances of that rule. Thus, for our example of the commutative Frobenius algebra, we have reduced our theory of 7 equations to just 2.

!-boxes were given a formal semantics in [7], making use of adhesive categories [8]. They also come with a simple and powerful induction principle introduced by one of the authors in [5] and proven correct in [10]. But there's a catch: note how we were careful to say that *commutative* Frobenius algebras

have an elegant presentation as above. A major drawback of the existing !-box notation is that it is only unambiguous if all of the nodes in the diagram are invariant under permuting inputs/outputs. This is severely limiting in two ways. The first and most obvious limitation is that we are forced to consider only commutative algebraic structures. The second, more subtle limitation is that we have no freedom to *definitionally* extend our theory, i.e. introduce new nodes defined as diagrams of other nodes, without making implicit assumptions about those diagrams (namely, that they are symmetric on inputs/outputs).

In order to overcome these shortcomings, we extend the !-graph notation with some extra information about how newly-created edges should be ordered when a !-box is expanded. This turns out to be fairly straightforward as soon as one shifts from a graph-based semantics for diagrams, as employed in [2], to a *tensor-based* semantics, where morphisms in the free compact closed category are represented using a version of Penrose's abstract tensor notation [11]. This approach, recently formalised in [6], has the property that non-commutativity comes 'for free', where the edges connected to a single element are represented as a list of edge names. Contrast this with the graph-based semantics for string diagrams or Joyal and Street's geometric construction [3], where one needs to add some extra structure (e.g. a total ordering or typing on adjacent edges) to break symmetries.

So, without further ado, we introduce tensor expressions for compact closed categories and extend them to accommodate !-boxes.

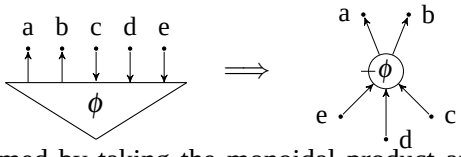
2 Tensors

Assume we are working in a compact closed category $\mathscr{C}$ freely generated by a set of objects $X,Y,Z,\ldots$ and morphisms of the form $\phi : I \to X_1 \otimes \ldots \otimes X_n$, i.e. morphisms with only non-trivial outputs. Since $\mathscr{C}$ is compact closed, this yields no loss of generality, since we represent an input of type A as an output of type A^*. For simplicity, we'll assume every 'input' is of fixed type X^* and every 'output' is of type X.

Since we want to distinguish inputs/outputs we label them using lower case letters. They will have a hat to illustrate being an 'output': $\{\hat{a},\hat{b},\ldots\}$, or a check to illustrate being an 'input': $\{\check{a},\check{b},\ldots\}$. Translating a morphism ϕ into tensor notation yields:

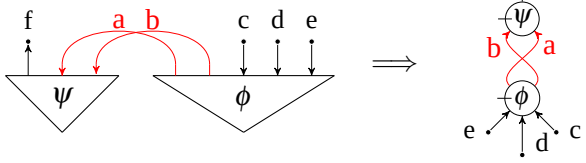
$$\phi : I \to X \otimes X \otimes X^* \otimes X^* \otimes X^* \qquad \Longrightarrow \qquad \phi_{\hat{a}\hat{b}\check{c}\check{d}\check{e}}$$

We introduce a special graphical notation for morphisms with only outputs. We write them as circles with a tick, taking the convention that inputs/outputs are ordered clockwise from the tick.



Writing two tensors side-by-side yields a new tensor formed by taking the monoidal product and 'contracting' any repeated names using the compact structure on X.

$$\psi_{\hat{f}\check{a}\check{b}}\phi_{\hat{a}\hat{b}\check{c}\check{d}\check{e}} \quad := \quad$$



(2)

We say repeated edge names (e.g. a and b above) are *bound* in a tensor expression, and all other edge names are *free*. In the graph we have labelled the bound edges, though this is purely for demonstrating which edges are bound. The names of bound edges can be changed at will, provided they are replaced with new, fresh names. Hence $\psi_{\hat{f}\check{a}\check{b}}\phi_{\hat{a}\hat{b}\check{c}\check{d}\check{e}}$ and $\psi_{\hat{f}\check{x}\check{y}}\phi_{\hat{x}\hat{y}\check{c}\check{d}\check{e}}$ represent the same graph. As a result, we typically will not write down bound names in the graphical notation.

Definition 2.1. The set of *tensor expressions* for a signature $\mathscr{S}$ consists of (i) the trivial tensor 1, (ii) the identity tensor $1_{\hat{a}\check{b}}$, (iii) atomic tensors $\psi_{\hat{a}\check{b}\ldots}$ with the appropriate names for each $\psi \in \mathscr{S}$, (iv) GH for G,H tensor expressions, and (v) G' obtained by changing some of the names of a tensor expression G—subject to the condition that $\hat{a}$ and $\check{a}$ occur at most once for each name a.

Definition 2.2. Two tensor expressions G,G' are equivalent, written $G \equiv G'$ if G can be made into G' by replacing bound names or by applying one or more of the following identities:

$$(GH)K \equiv G(HK) \qquad GH \equiv HG \qquad G1 \equiv G$$

$$G1_{\hat{b}\check{a}} \equiv G[\check{b} \mapsto \check{a}] \qquad H1_{\hat{a}\check{b}} \equiv H[\hat{b} \mapsto \hat{a}]$$

Assume for the last two identities that $\check{b}$ and $\hat{b}$ are free in G and H, respectively. An $\equiv$-equivalence class of tensor expressions is called a *tensor*.

Note that we use $\equiv$ for syntactic equivalence of tensor expressions (and later !-tensor expressions). We reserve the normal equals sign for equality by the rules of a given theory. As such, we always assume $(G \equiv H) \implies (G = H)$, but not the converse.

Tensors are related to morphisms in the free compact closed category as follows. Suppose we fix a set of *canonical names* $\{\check{x}_1,\check{x}_2,\ldots\}$ and $\{\hat{x}_1,\hat{x}_2,\ldots\}$. A tensor G is said to be *canonically named* if for some N it has as a free name precisely one of $\check{x}_i$ or $\hat{x}_i$ for $1 \leq i \leq N$.

Theorem 2.3. *Canonically-named tensors are in 1-to-1 correspondence to morphisms in the free compact closed category generated by a signature $\mathscr{S}$.*

Proof. First note that adding 'hats' and 'checks' to edge names is essentially applying the Int construction (c.f. [4]) to free traced symmetric monoidal category, in the tensorial presentation given in [6]. The free compact closure of the free traced monoidal category then satisfies the appropriate universal property to make it the free compact closed category. □

To summarise, we can interpret a tensor in a compact closed category as follows. First, we swap its free names for 'canonical names' (or otherwise order the outputs somehow), then interpret each atomic expression as a morphism (or one of a family of morphisms, parametrised by its arity). Finally, we construct the composed morphism by composing each of the components and contracting repeated edge names, as in (2).

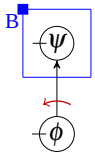
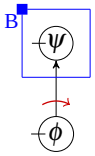
Alternatively, one can study models in an existing abstract tensor system (in the sense of Penrose), in which case interpretation is trivial. These two points of view (categorical vs. ATS) are roughly equivalent, as was shown in [6].

3 Adding !-boxes to tensor expressions

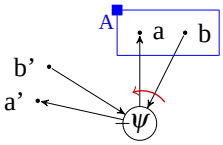
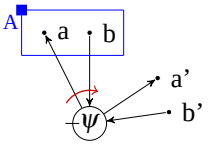
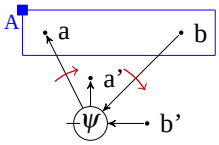
We now extend the existing tensor notation with !-boxes. Graphically !-boxes are blue boxes surrounding a subgraph, labelled with a name $(A,B,\ldots)$. We can denote this with square brackets around a subterm in a tensor expression, labelled with a superscript. Intuitively a !-box represents a portion of the graph that can be copied multiple times. For this to be well-defined in the non-commutative case we need to clarify where each new copy of the subgraph gets attached to surrounding nodes.

This is done by assigning an expansion direction (clockwise vs anticlockwise) to any group of edges from a node to a !-box. We draw these as arrows over edge groups in our !-graphs and for our tensors we

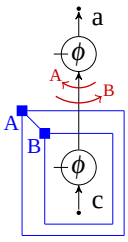
denote clockwise edge groups as $[\ldots\rangle^A$ and anticlockwise edge groups as $\langle\ldots]^A$. For example:

$\phi_{\langle \hat{a}]^B}\left[\psi_{\check{a}}\right]^B :=$  vs. $\phi_{[\hat{a}\rangle^B}\left[\psi_{\check{a}}\right]^B :=$ 

In the next section, we will see how the arrows clarify not only which direction edges should expand, but also whether they should expand in groups or individually. For example, the following notation gives anti-clockwise expansion of $\hat{a}\check{b}$ as a group, clockwise expansion of $\hat{a}\check{b}$ as a group, and clockwise expansion of $\hat{a}$ and $\check{b}$ as individual edges, respectively:

$\psi_{\hat{a}'\check{b}'\langle \hat{a}\check{b}]^A}\left[\right]^A$  vs. $\psi_{[\hat{a}\check{b}\rangle^A\hat{a}'\check{b}'}\left[\right]^A$  vs. $\psi_{[\hat{a}\rangle^A\hat{a}'[\check{b}\rangle^A\check{b}'}\left[\right]^A$ 

It is also possible for !-boxes to be nested inside other !-boxes. This means expansion of the parent box makes a new copy of the child with a new !-box name. Edge groups can correspondingly be nested if the edges enter more than one box. In the diagram to the left we have the !-graph with !-tensor expression: $\phi_{\hat{a}[\langle\check{b}]^B\rangle^A}\left[\left[\phi_{\hat{b}\check{c}}\right]^B\right]^A$. We have labelled which arrow corresponds to which !-box. This is not necessary if we adopt the convention that a parent box's arrow must be drawn closer to the node than it's child box's arrow. Note that the labels inside nodes are to assign a type to the node as apposed to naming the node. This means since we often have multiple nodes with the same type, we will have nodes with the same label.



We can now imagine more general generators allowing arbitrary arrangements of input and output edges. Any such node, say of type ϕ, then needs to be assigned a morphism in our category for each possible arrangement of edges. We represent an arrangment as a word over $\{\wedge,\vee\}$ where $\wedge$ represents outputs and $\vee$ represents inputs. For example the node  has edge arrangment $\wedge\vee\wedge\wedge$ and needs to be assigned a morphism $f : I \to X \otimes X^* \otimes X \otimes X$. Hence we need $\phi : \{\wedge,\vee\}^* \to \mathrm{Mor}(\mathscr{C})$ to model the node type ϕ.

!-tensors replace lists of edges on individual morphisms with *edgeterms* of which we now give a recursive definition.

Definition 3.1. Fix a disjoint, infinite sets $\mathscr{E}$ and $\mathscr{B}$ of edge names and !-box names, respectively. We denote the set of *directed edges* as $\overline{\mathscr{E}} := \{\check{a},\hat{a} : a \in \mathscr{E}\}$. The set of *edgeterms* $\mathscr{T}_e$ is defined recursively as follows:

- $\varepsilon \in \mathscr{T}_e$ (i.e empty)
- $\check{a},\hat{a} \in \mathscr{T}_e$ $a \in \mathscr{E}$
- $[e\rangle^A, \langle e]^A \in \mathscr{T}_e$ $e \in \mathscr{T}_e$, $A \in \mathscr{B}$
- $ef \in \mathscr{T}_e$ $e, f \in \mathscr{T}_e$

Two edgeterms are equivalent if one can be transformed into the other by:

$$\varepsilon e \equiv e \equiv e\varepsilon \qquad e(fg) \equiv (ef)g \qquad [\varepsilon\rangle^A \equiv \varepsilon \equiv \langle\varepsilon]^A$$

Since the well-formedness conditions for !-tensor expressions are a bit more complicated than for tensor expressions, we first define the set of all !-pretensor expressions, including those that may be ill-formed.

Definition 3.2. The set of all !-pretensor expressions $\mathscr{T}'_\Sigma$ for a signature Σ is defined recursively as:

$$\begin{array}{ll} \bullet\ 1, 1_{\hat{a}\check{b}} \in \mathscr{T}'_\Sigma & a, b \in \mathscr{E} \\ \bullet\ \phi_e \in \mathscr{T}'_\Sigma & e \in \mathscr{T}_e, \phi \in \Sigma \\ \bullet\ \big[G\big]^A \in \mathscr{T}'_\Sigma & G \in \mathscr{T}'_\Sigma,\ A \in \mathscr{B} \\ \bullet\ GH \in \mathscr{T}'_\Sigma & G, H \in \mathscr{T}'_\Sigma \end{array}$$

We introduce the notion of a *context*, which lists the !-boxes in which a certain edge name occurs, from the inside-out. These come in two flavours, *edge contexts* and *node contexts*.

Definition 3.3. Given a directed edge $a \in \overline{\mathscr{E}}$ in a !-tensor G nested as $\big[\big[\phi_{\ldots\langle\langle a\rangle^{E_1}\ldots\rangle^{E_n}\ldots}\big]^{N_1}\ldots\big]^{N_m}$.

We define the *edge context*, *node context*, and *context* of a respectively as:

$$\begin{array}{rll} \mathrm{ectx}_G(a) := [E_1, \ldots, E_n] & & \text{(edge context)} \\ \mathrm{nctx}_G(a) := [N_1, \ldots, N_m] & & \text{(node context)} \\ \mathrm{ctx}_G(a) := \mathrm{ectx}_G(a).\mathrm{nctx}_G(a) & & \text{(context)} \end{array}$$

That is, $\mathrm{ectx}_G(a)$ lists the !-boxes containing a that occur as part of a's edgeterm, and $\mathrm{nctx}_G(a)$ lists the rest.

Finally, a !-tensor expression is a !-pretensor expression where !-box/edge names must be suitably unique and occur in compatible contexts.

Definition 3.4. A !-tensor expression is a !-pretensor expression satisfying the following conditions:

F1. $\check{a}$ and $\hat{a}$ occur at most once for each edge name a

F2. $\big[\ldots\big]^A$ must occur at most once for each !-box name A

C1. $\mathrm{ectx}_G(a) \cap \mathrm{nctx}_G(a) = \varnothing$ for all edges $a \in \mathscr{E}$ in G

C2. If $\mathrm{ectx}_G(a) = [B_1, \ldots, B_n]$ then all $B_i \in \mathrm{Boxes}(G)$ and $B_1 \prec_G B_2 \prec_G \ldots \prec_G B_n$

C3. For all bound pairs $\check{a}, \hat{a}$ of edge names in G, there exist lists es, bs of !-box names such that:

$$es.\mathrm{nctx}_G(\check{a}) = \mathrm{ectx}_G(\hat{a}).bs \quad \text{and} \quad es.\mathrm{nctx}_G(\hat{a}) = \mathrm{ectx}_G(\check{a}).bs$$

where $A \prec_G B$ means that the !-box A is nested inside B in G (without other boxes nested between). We write $\mathscr{T}_\Sigma$ for the set of all !-tensor expressions for a signature Σ.

The freshness conditions F1 and F2 ensure that we have not used the same name for more than one edge/box. If a node is in !-box B then any edges attached to it are already in B so it wouldn't make sense to have B in both the $\mathrm{ectx}(a)$ and $\mathrm{nctx}(a)$ for $a \in \overline{\mathscr{E}}$, this is enforced by C1. C2 ensures that edge contexts are compatible with the !-boxes in the rest of the !-tensor. For example $\phi_{[[\check{a}\rangle^A\rangle^B}$ requires A to be nested in

B so does not result in a valid !-tensor when composed with e.g. $\left[\left[\psi_{\check{b}}\right]^B \xi_{\langle \hat{b}]^B}\right]^A$. C3 ensures that edges into !-boxes from the outside are decorated correctly by their edge terms. For instance, this is allowed: $\psi_{[\hat{a}\rangle^A}\left[\phi_{\check{a}}\right]^A$ but this is not: $\psi_{\hat{a}}\left[\phi_{\check{a}}\right]^A$. The freedom to pick bs, es allows bound pairs of edges to share some common context, e.g.: $\left[\psi_{\hat{a}}\phi_{\check{a}}\right]^A$ (both nodes are inside A) or $\psi_{[\hat{a}\rangle^A}\phi_{\langle \check{a}]^A}\left[\right]^A$ (only the edge is inside A). In the second example, A occurs in an edge term, so C2 requires the presence of $\left[\ldots\right]^A$ somewhere in the !-tensor, hence we append the 'empty' !-box $\left[\right]^A$.

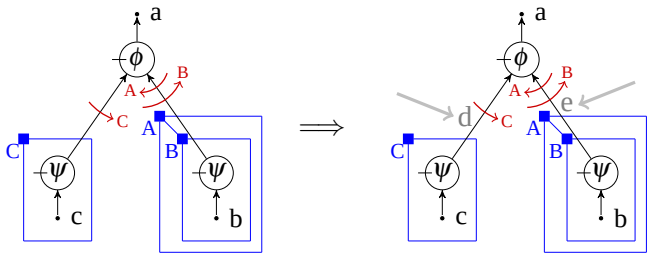
In this paper when we write a composition GH, unless stated otherwise, we will assume this forms a well defined !-tensor.

Naturally, we say two !-tensor expressions are equivalent, written $G \equiv H$, if one can be obtained from the other by using the usual tensor equivalences from Definition 2.2 or by using the edgeterm equivalences from Definition 3.1.

We call the graphical notation for !-tensors the *non-commutative !-graph notation*, or simply (non-commutative) !-graphs.

Theorem 3.5. *Any !-tensor can be represented unambiguously using non-commutative !-graph notation.*

Proof. We show this by providing a general procedure for interpreting a !-graph as a !-tensor expression, and vice-versa. For the sake of clarity, we demonstrate each step on a worked example. Given a non-commutative !-graph, we wish to obtain a unique equivalence class of !-tensor expressions under $\equiv$. Begin by choosing fresh names to write on all the interior edges.



Then, write the !-boxes with nesting as depicted in the diagram:

$$\ldots\left[\ldots\right]^C\left[\ldots\left[\ldots\right]^B\right]^A$$

Write each node in the diagram on the location it occurs (w.r.t. !-boxes):

$$\phi_{\ldots}\left[\psi_{\ldots}\right]^C\left[\left[\psi_{\ldots}\right]^B\right]^A$$

Finally, add the edges of each node, reading clockwise from the tick. Edges occurring under a clockwise arrow marked A should be enclosed in $[\ldots\rangle^A$, and edges under an anti-clockwise arrow should be enclosed in $\langle\ldots]^A$, where the outermost groups are the ones closest to the node in the picture.

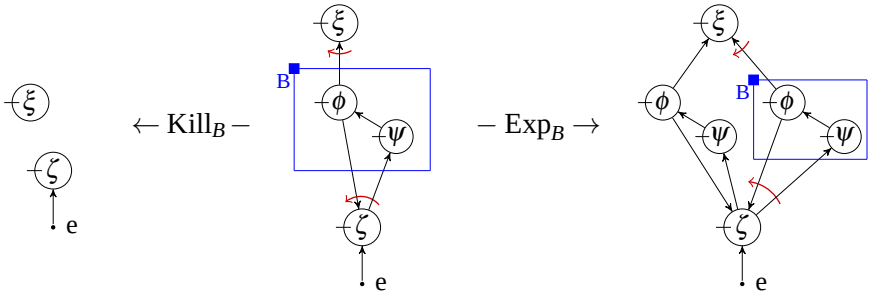
$$\phi_{\hat{a}[\langle \check{e}]^B\rangle^A\langle \check{d}]^C}\left[\psi_{\hat{d}\check{c}}\right]^C\left[\left[\psi_{\hat{e}\check{b}}\right]^B\right]^A$$

The only choices we made in this process were the choice of interior edge names and the order in which to write the individual tensors. However, up to $\equiv$, these are irrelevant. To show that any !-tensor can be represented this way, we simply run the above procedure in reverse. □

Because of this theorem, we use the terms !-tensor and !-graph interchangeably, depending on whether we wish to refer to the syntactic vs. graphical notation.

4 Instantiating tensor expressions with !-boxes

The following diagram demonstrates two !-box operations we can apply to a graph: killing a !-box is the operation deleting the box B and all contents (including edges to/from B), and expanding is the operation creating a new concrete instance of the subgraph inside B (attached appropriately). We can represent the original graph in this diagram with the tensor expression $\left[\phi_{\hat{a}\check{c}\hat{b}}\psi_{\hat{c}\check{d}}\right]^B \xi_{[\check{a}\rangle^B}\zeta_{\langle\check{b}\hat{d}]^B\check{e}}$.



We can define both of these operations formally. Since expansion involves copying various edge/!-box names, we need a means of obtaining fresh names. Let $\mathrm{Edges}(G) \subset \mathscr{E}$ and $\mathrm{Boxes}(G) \subset \mathscr{B}$ be the edge names and !-box names occurring in a !-tensor G, respectively.

Definition 4.1. A *freshness function* for a !-tensor G is a pair of bijections $\mathbf{fr} : \mathscr{E} \to \mathscr{E}$ and $\mathbf{fr} : \mathscr{B} \to \mathscr{B}$ such that

$$\mathrm{Edges}(G) \cap \mathbf{fr}(\mathrm{Edges}(G)) = \varnothing \quad \text{and} \quad \mathrm{Boxes}(G) \cap \mathbf{fr}(\mathrm{Boxes}(G)) = \varnothing$$

For !-tensor expressions G or edgeterms e, we will write $\mathbf{fr}(G)$ or $\mathbf{fr}(e)$ to designate the new expression with names substituted according to the given bijections.

Definition 4.2. We define $\mathrm{Op}_B \in \{\mathrm{Exp}_B, \mathrm{Kill}_B\}$ recursively over !-tensor expressions. For most cases, both operations act trivially:

$$\begin{aligned} \mathrm{Op}_B(GH) &:= \mathrm{Op}_B(G)\,\mathrm{Op}_B(H) & \mathrm{Op}_B(ef) &:= \mathrm{Op}_B(e)\,\mathrm{Op}_B(f) \\ \mathrm{Op}_B(\left[G\right]^A) &:= \left[\mathrm{Op}_B(G)\right]^A & \mathrm{Op}_B([e\rangle^A) &:= [\mathrm{Op}_B(e)\rangle^A \\ \mathrm{Op}_B(\phi_e) &:= \phi_{\mathrm{Op}_B(e)} & \mathrm{Op}_B(\langle e]^A) &:= \langle \mathrm{Op}_B(e)]^A \\ \mathrm{Op}_B(x) &:= x \end{aligned}$$

where $A \neq B$ and $x \in \{1, 1_{\hat{a}\check{b}}, \check{a}, \hat{a}, \varepsilon\}$. Then, for the final three cases:

$$\begin{aligned} \mathrm{Exp}_B(\left[G\right]^B) &:= \left[G\right]^B \mathbf{fr}(G) & \mathrm{Kill}_B(\left[G\right]^B) &:= 1 \\ \mathrm{Exp}_B([e\rangle^B) &:= [e\rangle^B \mathbf{fr}(e) & \mathrm{Kill}_B([e\rangle^B) &:= \varepsilon \\ \mathrm{Exp}_B(\langle e]^B) &:= \mathbf{fr}(e)\langle e]^B & \mathrm{Kill}_B(\langle e]^B) &:= \varepsilon \end{aligned}$$

Note that $\mathrm{Exp}_B(G)$ implicitly takes a freshness function as input. If we wish to make this explicit, we will write $\mathrm{Exp}_{B,\mathbf{fr}}$. The above operations can be lifted from !-tensor expressions to !-tensors, i.e. $\equiv$-classes of expressions, because of the following theorem.

Theorem 4.3. *Let* $\mathbf{fr}$ *be a freshness function for two !-tensor expressions* G, H*. Then* $G \equiv H$ *implies* $\mathrm{Exp}_{B,\mathbf{fr}}(G) \equiv \mathrm{Exp}_{B,\mathbf{fr}}(H)$ *and* $\mathrm{Kill}_B(G) \equiv \mathrm{Kill}_B(H)$.

Proof. (Sketch) This can be shown by induction over the structure of !-tensor expressions. It is crucial that we use the *same* freshness function **fr** for the expansions of G and H, otherwise G and H could end up with distinct free edges or !-boxes. □

These two !-box operations give us a means to define the set of all (concrete) tensors that a single !-tensor represents.

Definition 4.4. A tensor G' is a *concrete instance* of a !-tensor G if it is obtained from G by repeatedly applying the two !-box operations Exp and Kill until G' contains no !-boxes. This sequence of operations is called the *instantiation* of G'. We write $[\![G]\!]$ for the set of all concrete instances of G.

When we fix a model in some category $\mathscr{C}$, concrete tensors can then be interpreted as morphisms in $\mathscr{C}$, just as before. We therefore interpret each !-tensor expression as a family of morphisms in $\mathscr{C}$, namely the interpretations of each of its concrete instances.

5 Reasoning with !-boxes

The real power of !-boxes comes from the ability to do equational reasoning using infinite families of rules. Just as it makes sense to instantiate a single !-tensor, it makes sense to instantiate an *equation* $G = H$ between two !-tensors, provided they have compatible boundaries.

Definition 5.1. A *!-tensor equation* '$G = H$' consists of a pair of !-tensors (G,H) that have *compatible boundaries*. That is, they have identical free edge names and !-boxes, $A \prec_G B \Leftrightarrow A \prec_H B$ for all !-boxes in G and H, and $\mathrm{ctx}_G(a) = \mathrm{ctx}_H(a)$ for all free edge names.

Intuitively, we require that the LHS and RHS of a !-tensor equation have the same interface to attach to other graphs (same free variables and same box structure). These consistency conditions guarantee that (i) applying !-box operations to valid equations yields valid equations, and (ii) when G occurs as a sub-expression of some other !-tensor K, it can be substituted for H to yield another valid !-tensor K'.

Theorem 5.2. *Let* **fr** *be a freshness function for !-tensors G,H. Then, if $G = H$ is a !-tensor equation, then so too are:*

$$\begin{aligned} \mathrm{Kill}_B(G = H) &:= (\mathrm{Kill}_B(G) = \mathrm{Kill}_B(H)) \\ \mathrm{Exp}_B(G = H) &:= (\mathrm{Exp}_{B,\mathbf{fr}}(G) = \mathrm{Exp}_{B,\mathbf{fr}}(H)) \end{aligned}$$

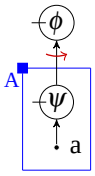
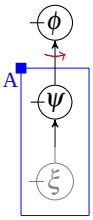
Proof. (Sketch) It is straightforward to show that killing/expanding B affects the free variables and the !-boxes in the same way on the LHS/RHS. To check the contexts, split a single free edge name into 3 cases, depending on whether the !-box B occurs in the node-context of a, the edge context of a, or neither. In all cases, a and/or $\mathbf{fr}(a)$ will have identical contexts on the LHS/RHS. □

As in the case of !-tensors, we can define $[\![G = H]\!]$ to be the set of all concrete rules derivable from $G = H$ using the !-box operations. A valid model of a graphical theory is then one where all of the equations in $[\![G = H]\!]$ hold for each equation $G = H$. Proving that a rule holds for *all* of its instances could be a daunting task in general, however in many cases a technique called *!-box induction*—which we will meet shortly—comes to the rescue.

We obtain a notion of substitution of sub-expressions constructively, via inference rules. The first few should look familiar as congruence- and substitution-like rules for !-tensors.

$$\frac{G = H}{GK = HK}\ (\text{Prod}) \qquad \frac{G = H}{\left[G\right]^A = \left[H\right]^A}\ (\text{Box}) \qquad \frac{G = H}{G[a \to b] = H[a \to b]}\ (\text{Rename})$$

Where $G[a \to b]$ and $H[a \to b]$ are G and H with the free edge/!-box name a replaced by b. We require that K and A are chosen such that GK, HK, $\big[G\big]^A$, and $\big[H\big]^A$ are well-defined. These rules provide the conditions under which some equation $G = H$ can be unified, given some context, with a bigger equation $G' = H'$. The final inference rule (Weaken) is less intuitive from the point of view of terms, and is best understood graphically. Consider the following embedding of !-graphs:

$\big[\psi_{\check{a}\hat{b}}\big]^A \phi_{\langle \check{b} \rangle^A} :=$  $\hookrightarrow$  $=: \big[\psi_{\check{a}\hat{b}}\xi_{\hat{a}}\big]^A \phi_{\langle \check{b} \rangle^A}$

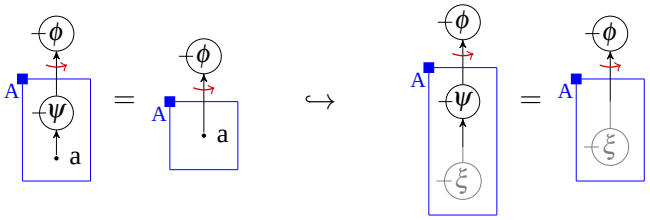
The LHS does not embed as a sub-term of the RHS, because the !-box A contains more stuff on the RHS. However, semantically, this is perfectly fine, as all of the concrete instances of the LHS will have (uniquely-determined) embeddings into all of the concrete instances of the RHS. So, we also need a rule that allows us to 'weaken' !-boxes by adding more nodes to them.

$$\frac{G = G'}{\mathrm{W}_{A \mapsto K}(G) = \mathrm{W}_{A \mapsto K}(G')} \text{ (Weaken)}$$

Where $\mathrm{W}_{A \mapsto K}(G)$ is defined recursively as:

$$\begin{aligned}
\mathrm{W}_{A \mapsto K}(\big[G\big]^A) &:= \big[GK\big]^A \\
\mathrm{W}_{A \mapsto K}(\big[G\big]^B) &:= \big[\mathrm{W}_{A \mapsto K}(G)\big]^B && \text{if } A \neq B \\
\mathrm{W}_{A \mapsto K}(GH) &:= \mathrm{W}_{A \mapsto K}(G)\,\mathrm{W}_{A \mapsto K}(H) \\
\mathrm{W}_{A \mapsto K}(x) &:= x && x \in \{1, 1_{\hat{a}\check{b}}, \phi_e\}
\end{aligned}$$

These four rules give us everything we need to apply equations on !-tensors to obtain new equations. For example, the equation on the left below can be applied to delete all of the ψ-nodes occurring as input to a ϕ. An example application of this rule is shown on the right.



We can also add inference rules for each of our !-box operations i.e.

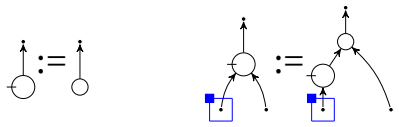
$$\frac{G = H}{\mathrm{Exp}_B(G = H)} \text{ (Exp)} \qquad \frac{G = H}{\mathrm{Kill}_B(G = H)} \text{ (Kill)}$$

Perhaps most interestingly, we can introduce new !-boxes, where previously there were none, via *!-box induction*.

$$\frac{\mathrm{Kill}_A(G = H) \qquad G = H \Rightarrow \mathrm{Exp}_A(G = H)}{G = H} \text{ (Induction)}$$

As mentioned in Section 1, non-commutative nodes give us the ability to make recursive definitions of variable-arity generators in terms of fixed-arity generators of our theory. This induction principle in turn gives us the means to lift rules about fixed arity generators up to more powerful !-tensor rules.

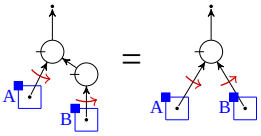
We conclude by showing a simple example. Suppose we take the theory of a monoid, i.e. the pair of generators $\left(\ \ ,\ \ \right)$ satisfying the commutativity and unit laws from (1).Then we can recursively define, as a new generator, an n-fold tree of multiplications.



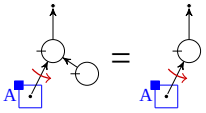
(3)

Remark 5.3. Note how non-commutative !-boxes make such recursive definitions possible in the first place, without assuming *a priori* that the family of graphs generated by the definition are symmetric on their inputs/outputs. This need not be true, even in the case where all of the concrete generators are commutative. This limitation in the case of commutative !-boxes was highlighted in [10], where only a partial proof of the spider theorem for commutative Frobenius algebras could be done using (commutative) !-box induction.

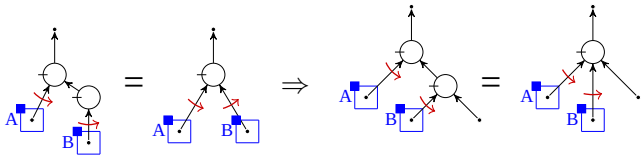
The first property we would like to prove about such trees is that adjacent trees merge to form bigger trees. As a !-box rule, it looks like this:



We can then hit this rule with the induction on B to break it into cases:

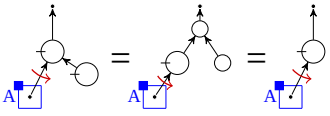


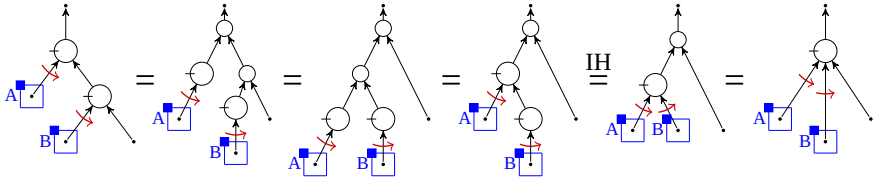
(base)



(step)

...each of which have simple rewriting proofs:





One caveat is that when we apply the induction hypothesis in step 4, the !-box B must be 'fixed' (i.e. we're not allowed to do any instantiation of B via Exp, Kill, etc.). This is because B occurs free on both sides of the implication $G = H \Rightarrow \mathrm{Exp}_B(G = H)$. See [10] for details.

This style of proof is the main workhorse of soundness proofs of rules like the merging rule (a.k.a. 'spider rule') for commutative Frobenius algebras described in Section 1, and can be extended to the non-commutative case, proving a similar rule for e.g. *symmetric* Frobenius algebras, giving a purely diagrammatic characterisation of the normal forms described in [9].

References

[1] Lucas Dixon & Ross Duncan (2009): *Graphical Reasoning in Compact Closed Categories for Quantum Computation*. *AMAI* 56(1), p. 20, doi:10.1017/S0305004100074338.

[2] Lucas Dixon & Aleks Kissinger (2013): *Open-graphs and monoidal theories*. *Mathematical Structures in Computer Science* 23, pp. 308–359, doi:10.1017/S0960129512000138. arXiv:1007.3794v1 [cs.LO].

[3] Andre Joyal & Ross Street (1991): *The geometry of tensor calculus I*. *Advances in Mathematics* 88, pp. 55–113, doi:10.1016/0001-8708(91)90003-P.

[4] Andre Joyal, Ross Street & Dominic Verity (1996): *Traced Monoidal Categories*. *Math. Proc. Camb. Phil. Soc.* 119(3), pp. 447–468, doi:10.1017/S0305004100074338.

[5] Aleks Kissinger (2011): *Pictures of Processes: Automated Graph Rewriting for Monoidal Categories and Applications to Quantum Computing*. Ph.D. thesis, University of Oxford. arXiv:1203.0202 [math.CT].

[6] Aleks Kissinger (2014): *Abstract Tensor Systems as Monoidal Categories*. In C Casadio, B Coecke, M Moortgat & P Scott, editors: *Categories and Types in Logic, Language, and Physics: Festschrift on the occasion of Jim Lambek's 90th birthday, Lecture Notes in Computer Science* 8222, Springer, doi:10.1007/978-3-642-54789-8_13. arXiv:1308.3586 [math.CT].

[7] Aleks Kissinger, Alex Merry & Matvey Soloviev (2012): *Pattern Graph Rewrite Systems*. In: *Proceedings of DCM 2012, EPTCS* 143, doi:10.4204/EPTCS.143.5. arXiv:1204.6695 [math.CT].

[8] Stephen Lack & Pawel Sobocinski (2005): *Adhesive and quasiadhesive categories*. *Theoretical Informatics and Applications* 39(2), pp. 522–546, doi:10.1051/ita:2005028.

[9] Aaron D. Lauda & Hendryk Pfeiffer (2008): *Open-closed strings: Two-dimensional extended TQFTs and Frobenius algebras*. *Topology Appl.* 155(7), pp. 623–666, doi:10.1016/j.topol.2007.11.005.

[10] Alexander Merry (2014): *Reasoning with !-Graphs*. Ph.D. thesis, University of Oxford.

[11] R. Penrose (1971): *Applications of negative dimensional tensors*. In: *Combinatorial Mathematics and its Applications*, Academic Press, pp. 221–244.