The Graphical Language of Symmetric Traced Monoidal Categories

George Kaye

School of Computer Science, University of Birmingham, UK

October 13, 2020

Abstract

This report examines a variant of hypergraphs known as *linear hypergraphs*, with the aim of creating a sound and complete graphical language for symmetric traced monoidal categories (STMCs). We begin by defining the category of linear hypergraphs as a full subcategory of traditional (simple) hypergraphs, in which vertices are the sources and target of exactly one edge each. By defining the operations of composition, symmetry and trace, we show that morphisms equal in an STMC can be interpreted as isomorphic linear hypergraphs (soundness) and that any linear hypergraph has a corresponding unique morphism up to the equations of the STMC (completeness). We use this graphical language to show how we can perform graph rewriting in an STMC using double pushout rewriting, by demonstrating that linear hypergraphs form a partial adhesive category.

1 Introduction

Constructors, architects, and engineers have always enjoyed using blueprints, diagrams, floorplans, schematics, sketches and other kinds of graphical representations of their designs. In many cases, they are more than simply illustrations aiding the understanding of a formal specification, they are the specification itself. By contrast, in mathematics diagrams have not been traditionally considered first-class citizens, although they are often used to help the reader *visualise* a construction or a proof. However, the development of new *formal* diagrammatic languages for a large variety of systems such as quantum communication and computation [7], computational linguistics [8], signal-flow graphs [2], or electronic circuits [1], proved that diagrams can be used not just to aid understanding of proofs, but also to formulate proofs. The graphical formulation has multifaceted advantages, from enabling the use of graph-theoretical techniques to aid reasoning [12] to making the teaching of higher algebraic concepts to younger students less intimidating [10].

These graphical languages build on a common mathematical infrastructure of (usually symmetric and strict) monoidal categories [16], and in particular compact closed categories [18]. One important class of systems that do not fit the compact closed framework are those with a clearly defined notion of *input* and *output*, and in particular digital circuits.

1

A system modelled by a compact closed category has a general notion of *interface port*, and the formalism will allow any two such ports, provided the types match, to be connected. In contrast, in a digital circuit connections may only happen between ports with the same type but opposite input-output polarities. Compact closed categories describe systems with a flexible and refined notion of *causality*, such as quantum systems [20] or games [6]. Electronic circuits, and other artificial systems, have a more rigid notion of causality built-in and require a different kind of categorical setting, namely that of a *symmetric traced monoidal category* (STMC), which satisfies the requirements above. The permissive way in which any two ports in a system modelled by compact closed categories can be connected is replaced by an input-output discipline which directs connectivity, with an explicit construct (the trace) modelling causal feedback loops.

Our primary motivation is to revisit the graphical language of STMCs used to give a formal semantics for digital circuits [11, 12] and prove the following two results, which are *the main contributions of the paper*:

- First, we prove that the graphical language is *sound and complete* for traced monoidal categories, so the diagrammatic proofs in *loc. cit.* are valid.

- Second, we show that the graphical language forms an *partial adhesive category* so that the equations in *loc. cit.* can be expressed as graph rewriting rules without any ambiguity.

We aim to keep our presentation as self-contained and elementary as possible. Some of the concepts involved could be expressed in a more mathematically sophisticated way, and this could have simplified some of the proofs. However, we think that this sophistication can raise the barrier in terms of accessibility for a broader audience, so we prefer a more elementary and more direct approach, even at the cost of some more intricate proofs.

1.1 Structure of the report

The structure of the report is as follows. In §2 we introduce a standard definition of hypergraphs, and then refine this to obtain *linear hypergraphs*, which are motivated by our study of string diagrams. §3 details several hypergraph constructs and operations that will be of use to us. We then use these ingredients in §4 to show that we can represent morphisms in a free PROP (a category of PROducts and Permutations, where objects are natural numbers) as hypergraphs. We take the opposite perspective in §5, to show that we can also recover categorical terms from hypergraphs, showing that we have both soundness and completeness. In §6 we study graph rewriting, a useful application of our graphical language, and in §7 follow with a case study into the axioms related to digital circuits. Finally in §8 we generalise our approach to consider terms from any STMC, not just PROPs. Details of many of the proofs throughout this report can be found in the appendices.

1.2 Notation

For two sets X and Y, let $X+Y$ be their disjoint union and $X-Y$ be the relative complement of Y in X. Let $|X|$ be the cardinality of a set X. We write $[n] \subset \mathbb{N}$ as the subset of the natural numbers containing $0, 1, \cdots, n-1$. Observe that we therefore write $[|X|]$ for the

subset of the natural numbers of equal cardinality to set X. Let $(X, \leq^X)$ be a totally ordered set, where usually we will just write the carrier X. We use π_x as a 'projection' function to denote the xth element of a totally ordered set. When considering two totally ordered sets X and Y with total orders $\leq^X$ and $\leq^Y$, we write as $\leq^X \oslash \leq^Y$ a total order on their disjoint union in which all elements of X are less than elements in Y, and the original orders on X and Y are preserved.

Let $X^\star$ be the free monoid on a set X and $f^\star : X^\star \to Y^\star$ the pointwise application of a function $f : X \to Y$. For a function $f : X \to Y$, we denote by $f^\star : X^\star \to Y^\star$ its pointwise application to each element in its input.

When we have two functions $f : X \to Y$ and $g : U \to V$ where X and U are disjoint, we can define a function $f \oplus g : X + U \to Y \cup V$, which acts as f on elements of X and g on elements of U.

For a category $\mathcal{C}$, the hom-set for objects A and B is denoted $\mathcal{C}(A, B)$. Parallel composition $(- \otimes -$, read top-to-bottom) binds tighter than sequential composition $(- \cdot -$, read left-to-right), that is $f \otimes g \cdot h \otimes k$ should be read as $(f \otimes g) \cdot (h \otimes k)$.

2 Hypergraphs

2.1 Simple hypergraphs

We begin by introducing a standard definition of hypergraphs, where edges can connect to an arbitrary number of vertices. Let $\mathbb{A}$ be a countably infinite set of atoms (in the sense of [23]).

Definition 1 (Simple hypergraph). *A simple hypergraph is a tuple $H = (V, E, \mathsf{s}, \mathsf{t})$ where*

- *$V \subset \mathbb{A}$ is a set of vertices*

- *$E \subset \mathbb{A}$ is a set of edges*

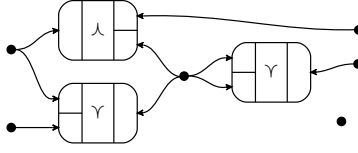- *$\mathsf{s}, \mathsf{t} : E \to V^\star$ are ordered source and target functions*

A *hypergraph signature* is a set of edge labels Σ equipped with functions $\mathsf{dom}, \mathsf{cod} : \Sigma \to \mathbb{N}$. A *labelled simple hypergraph* over signature Σ is a hypergraph $H = (V, E, \mathsf{s}, \mathsf{t})$ equipped with a labelling function $\Lambda : E \to \Sigma$, where for any $e \in E$, if $\Lambda(e) = l$ then $\mathsf{dom}(l) = |\mathsf{s}(e)|$ and $\mathsf{cod}(l) = |\mathsf{t}(e)|$.

Example 2 (Labelled simple hypergraph). *Below is an informal drawing of a labelled simple hypergraph over the signature*

$$\Sigma = \{ \curlywedge : 1 \to 2, \ \curlyvee : 2 \to 1 \}$$

containing a 'fork' and 'join' edge. Vertices are drawn as black dots. Edges are drawn as boxes, with ordered sources on the left and ordered targets on the right.

$$V = \{ v_0, v_1, v_2, v_3, v_4, v_5 \} \qquad E = \{ e_0, e_1, e_2 \}$$

$$\mathsf{s} = \{ e_0 \mapsto [v_0], e_1 \mapsto [v_0, v_1], e_2 \mapsto [v_2, v_2] \} \qquad \mathsf{t} = \{ e_0 \mapsto [v_3, v_2], e_1 \mapsto [v_2], e_2 \mapsto [v_4] \}$$

$$\Lambda = \{ e_0 \mapsto \curlywedge, e_1 \mapsto \curlyvee, e_2 \mapsto \curlywedge \}$$

3

A (labelled) simple hypergraph homomorphism $h : F \to G$ consists of functions h_V and h_E such that the following diagrams commute:

$$
\begin{array}{ccc}
E_F \xrightarrow{\ \mathsf{s}_F\ } V_F^\star & E_F \xrightarrow{\ \mathsf{t}_F\ } V_F^\star & E_F \xrightarrow{\ \Lambda_F\ } \Sigma \\
\Big\downarrow{\scriptstyle h_E} \quad \Big\downarrow{\scriptstyle h_V} & \Big\downarrow{\scriptstyle h_E} \quad \Big\downarrow{\scriptstyle h_V} & \Big\downarrow{\scriptstyle h_E} \quad \Big\downarrow{\scriptstyle \mathsf{id}} \\
E_G \xrightarrow{\ \mathsf{s}_G\ } V_G^\star & E_G \xrightarrow{\ \mathsf{t}_G\ } V_G^\star & E_G \xrightarrow{\ \Lambda_G\ } \Sigma
\end{array}
$$

We say that two simple hypergraph homomorphisms are equal if their components are equal. If h_V and h_E are bijective then F and G are isomorphic, written $F \equiv G$. It is immediate that $\equiv$ is an equivalence relation.

Hypergraph homomorphisms are the morphisms in the category of simple hypergraphs **SHyp**, defined as a functor category [3]. Hypergraph signatures Σ can be seen as simple hypergraphs, with a single vertex v and edges for each label $m \to n$ in the signature, where v appearing m (n respectively) times in its sources (targets respectively). Thus we can define labelled simple hypergraphs as a slice category.

Definition 3 (Category of (labelled) simple hypergraphs)**.** *Let **SHyp** be the functor category* $[\mathbf{X}, \mathbf{Set}]$*, where **X** has objects pairs of natural numbers (m, n) (edges with m sources and n targets) and an extra object* ▼ *(vertices). For each object $x = (m, n)$, there are $m + n$ arrows from x to* ▼*. Let **SHyp**$_\Sigma$ = **SHyp**$/\Sigma$ be the slice category over a hypergraph signature Σ.*

We call a hypergraph homomorphism an *embedding* if its components are injective.

Lemma 4 (Simple hypergraph monomorphisms)**.** *A morphism in **SHyp**$_\Sigma$ is a monomorphism iff its components are injective.*

Proof. For a morphism $m : F \to G$ to be mono, for any two morphisms $p, q : H \to F$ (for any other hypergraph H), if $m \circ p = m \circ q$ then $p = q$. First we show that if m is an embedding it must be mono. If we consider each equivalence map of m separately, this means that we must show that $m_V \circ p_V = m_V \circ q_V$ implies $p_V = q_V$ (and the same for m_E). But we have assumed that m_T is injective, so the antecedent reduces to $g_T = h_T$. So m is mono.

Conversely, if m is not an embedding, it cannot be a monomorphism. A morphism that is not an embedding maps multiple vertices or edges into one. Therefore for a morphism $m : F \to G$ that maps vertices v_1 and v_2 in F to v in G, there exist two morphisms $p, q : G \to F$ (in p, $v \mapsto v_1$ and in q, $v \mapsto v_2$), and similar for morphisms that map multiple edges to one. Therefore there exist p, q such that $p \neq q$, so m is not mono. $\qquad\square$

4

2.2 Linear hypergraphs

In simple hypergraphs, vertices can connect to an arbitrary number of edges. However, to make wires in string diagrams split or join, an additional Frobenius structure must be imposed. This structure works particularly well in the framework of compact closed categories, but this is a structure which we aim to avoid. Therefore we must restrict simple hypergraphs so that each target of an edge only 'connects' to exactly one source.

Another limitation of simple hypergraphs is that they have no interfaces. These are usually added using ordered cospans [3], but we prefer to build the interfaces directly into the structure of the hypergraph. This way, it is straightforward to require that each target of an edge connects to only one other source.

To solve these problems, we define a variant on simple hypergraphs called *linear hypergraphs*. We add an extra 'edge' $\blacklozenge \in 1$ to denote the interface of the hypergraph, and we require each vertex to have exactly one 'left' and one 'right' connection.

Definition 5 (Linear hypergraph). *A linear hypergraph is a tuple*

$$H = ((T, \leq^T), (S, \leq^S), E, \lambda, \rho, \kappa)$$

where

- *$T, S \subset \mathbb{A}$ are two finite disjoint totally ordered sets with equal cardinality, of target and source vertices respectively*

- *$E \subset \mathbb{A}$ is a finite set of edges*

- *$\lambda : T \to E + 1$ and $\rho : S \to E + 1$ are left and right functions that assign target, respectively source, vertices to edges*

- *$\kappa : T \to S$ is a connections bijection linking target and source vertices*

Target vertices v such that $\lambda(v) = \mathsf{inr}(\blacklozenge)$ form the set of *inputs* written $T^{\blacklozenge}$. Source vertices v such that $\rho(v) = \mathsf{inr}(\blacklozenge)$ form the set of *outputs* written $S^{\blacklozenge}$. For a hypergraph H with m inputs and n outputs, we write it as $H : m \to n$, where $m \to n$ is the *type* of the hypergraph.

We can also derive functions $\mathsf{s} : E \to S^\star$ and $\mathsf{t} : E \to T^\star$ as the inverse image of ρ and λ respectively, representing the *sources* and *targets* of an edge. As with simple hypergraphs, we can define *labelled linear hypergraphs* over a signature Σ with labelling function Λ as a tuple $H = ((T, \leq^T), (S, \leq^S), E, \lambda, \rho, \kappa, \Lambda)$.

Example 6 (Labelled linear hypergraph). *Below is an example of a labelled linear hypergraph over the signature Σ defined in Example 2.*

$$H = ((T, \leq^T), (S, \leq^S), E, \lambda, \rho, \kappa, \Lambda)$$
$$T = \{v_0, v_1, v_2, v_3, v_4\} \qquad S = \{v_5, v_6, v_7, v_8, v_9\} \qquad E = \{e_0, e_1\}$$
$$\lambda = \{v_0 \mapsto \blacklozenge, v_1 \mapsto \blacklozenge, v_2 \mapsto e_0, v_3 \mapsto e_0, v_4 \mapsto v_4\}$$
$$\rho = \{v_5 \mapsto e_0, v_6 \mapsto e_1, v_7 \mapsto e_1, v_8 \mapsto \blacklozenge, v_9 \mapsto \blacklozenge\}$$
$$\kappa = \{v_0 \mapsto v_7, v_1 \mapsto v_5, v_2 \mapsto v_8, v_3 \mapsto v_6, v_4 \mapsto v_0\}$$
$$\Lambda = \{e_0 \mapsto \curlywedge, e_1 \mapsto \curlyvee\}$$

5

Below are two ways of drawing a labelled linear hypergraph over Σ that will be used throughout this document. On the left is a 'formal' notation where source (target respectively) vertices are drawn as black dots on the left (right respectively) of the diagram, with arrows connecting them to the appropriate edge in the stack in the middle. Connections are represented by the arrow on the far right. We represent the inputs (outputs respectively) of the term as incident on a grey edge labelled α (ω respectively), to distinguish them from the actual edges.

On the right is a more intuitive, informal representation, where connected target and source vertices are drawn as a single black dot. The orders on the vertices dictate the position of each arrow incident on an edge. Again, vertices incident on the interface are shown as connecting to grey α and ω edges.
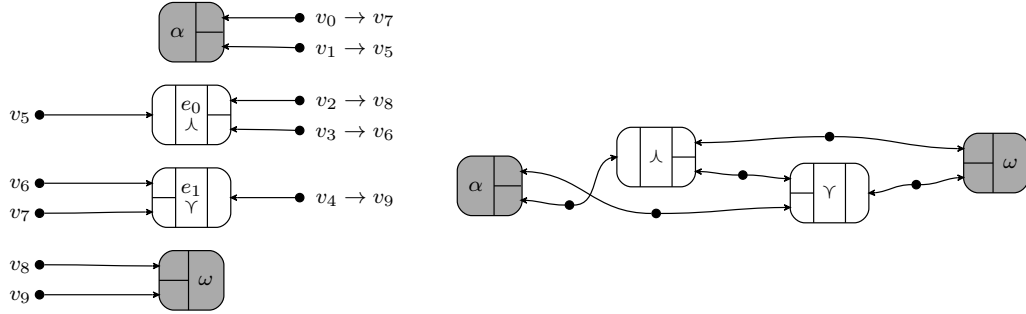
A labelled linear hypergraph homomorphism $h : F \to G$ consists of functions h_T, h_S and h_E such that the following diagrams commute:

$$
\begin{array}{ccc}
E_F \xrightarrow{\ s_F\ } S_F^\star & E_F \xrightarrow{\ t_F\ } T_F^\star & T_F \xrightarrow{\ \kappa_F\ } S_F & E_F \xrightarrow{\ \Lambda_F\ } \Sigma \\
\downarrow{h_E} \quad \downarrow{h_S^\star} & \downarrow{h_E} \quad \downarrow{h_T^\star} & \downarrow{h_T} \quad \downarrow{h_S} & \downarrow{h_E} \quad \downarrow{\mathsf{id}} \\
E_G \xrightarrow{\ s_G\ } S_G^\star & E_G \xrightarrow{\ t_G\ } T_G^\star & T_G \xrightarrow{\ \kappa_G\ } S_G & E_G \xrightarrow{\ \Lambda_G\ } \Sigma
\end{array}
$$

We say that two linear hypergraph homomorphisms are equal if their three components are equal. If h_T, h_S and h_E are bijective and the following diagrams also commute:

$$
\begin{array}{cc}
1 \xrightarrow{\ t_F\ } T_F^\star & 1 \xrightarrow{\ s_F\ } S_F^\star \\
\downarrow{\mathsf{id}} \quad \downarrow{h_T^\star} & \downarrow{\mathsf{id}} \quad \downarrow{h_S^\star} \\
1 \xrightarrow{\ t_G\ } T_G^\star & 1 \xrightarrow{\ s_G\ } S_G^\star
\end{array}
$$

then F and G are *isomorphic* $F \equiv G$. Once more, it is immediate that $\equiv$ is an equivalence relation.

Labelled linear hypergraphs form a category $\mathbf{LHyp}_\Sigma$ with objects the hypergraphs over signature Σ and morphisms the hypergraph homomorphisms. As with the simple variant, we call a homomorphism an embedding if its components are injective.

Lemma 7 (Linear hypergraph monomorphisms)**.** *A morphism in $\mathbf{LHyp}_\Sigma$ is mono iff it is an embedding.*

Proof. As with simple hypergraphs (Lemma 4). $\qquad \square$

Lemma 8. *For any linear hypergraph homomorphism* $h : F \to G$, h_T *is injective iff* h_S *is injective.*

Proof. For ($\Rightarrow$), assume that h_S is injective and h_T is not injective. Then there exist $v_1, v_2 \in F$ and $v_3 \in G$ such that $h_T(v_1) = h_T(v_2) = v_3$. Since h_S is injective and κ_F is bijective, $h_S \circ \kappa_F(v_1) \neq h_S \circ \kappa_F(v_2)$. But by the homomorphism conditions $\kappa_G(v_3) = h_S \circ \kappa_F(v_1) = h_S \circ \kappa_F(v_2)$, a contradiction. Therefore if h_S is injective then h_T is too. The reverse also holds since κ is bijective. $\qquad \square$

It is clear that linear hypergraphs are merely simple hypergraphs with some restrictions, so they form a subcategory.

Definition 9 (Inclusion functor)**.** *For any linear hypergraph F we can represent it as a simple hypergraph with the inclusion functor $I : \mathbf{LHyp}_\Sigma \to \mathbf{SHyp}_\Sigma$, with its effect defined as*

$$I(f) = (S, E, \mathsf{s}, \kappa^\star \circ \mathsf{t}).$$

Lemma 10 (Equivalence of homomorphisms)**.** *Under the inclusion functor I, the notion of a linear hypergraph homomorphism is equivalent to that of a simple hypergraph homomorphism.*

Proof. For the sources condition, S is identified with V and E is unchanged so the condition immediately holds. The labelling condition is unchanged. We can derive the simple targets condition from the combination of the linear targets and connections conditions.

$$
\begin{array}{ccccc}
E_F & \xrightarrow{\;\mathsf{t}_F\;} & T_F^\star & \xrightarrow{\;\kappa_F^\star\;} & S_F^\star \\
\downarrow{\scriptstyle f_E} & & \downarrow{\scriptstyle h_T^\star} & & \downarrow{\scriptstyle h_S^\star} \\
E_G & \xrightarrow{\;\mathsf{t}_G\;} & T_G^\star & \xrightarrow{\;\kappa_G^\star\;} & S_G^\star
\end{array}
$$

$\qquad \square$

Corollary 11. $\mathbf{LHyp}_\Sigma$ *is a full subcategory of* $\mathbf{SHyp}_\Sigma$.

3 Constructs and operations

This section will detail some essential operations that we can perform with our hypergraphs, which will help us obtain our coherence result. Examples throughout the section will be performed using three hypergraphs $F : 1 \to 1$, $G : 1 \to 2$ and $H : 2 \to 2$, each containing one edge labelled with the corresponding lower case letter.
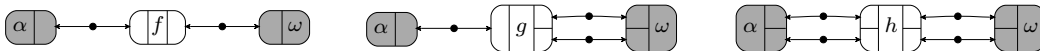
Recall that a function must be *everywhere-defined*, that is to say for any function $f : X \to Y$, for all $x \in S$, there exists some $y \in Y$ such that $f(x) = y$. We call a linear hypergraph $F = ((T, \leq^T), (S, \leq^S), E, \lambda, \rho, \kappa, \Lambda)$ *well-formed* if λ, ρ, κ and Λ are everywhere-defined, κ is bijective, and the labelling condition is satisfied.

3.1 Equivariance

When performing operations on hypergraphs, we often require that their vertices and edges are disjoint. However this is not necessarily always the case, for example composing a hypergraph with itself. Fortunately, since the sets T, S and E are subsets of the countably infinite set of atoms $\mathbb{A}$, we can simply *rename* the problematic edges or vertices [23].

Definition 12 (Action). *For any permutation $\tau : \mathbb{A} \to \mathbb{A}$, an action $\tau \bullet -$ acts as follows:*

Element *For any elements $x \in \mathbb{A}, y \notin \mathbb{A}, \tau \bullet x = \tau(x)$ and $\tau \bullet y = y$.*

Set *For any set X, $\tau \bullet X = \{\tau \bullet x \mid x \in X\}$.*

Total order *For any total order $\leq^X$, $\tau \bullet X_{\leq} = \{(x, y) \mid x, y \in X, \tau^{-1} \bullet x < \tau^{-1} \bullet y\}$.*

Function *For any function $f : X \to Y, (\tau \bullet f)(v) = \tau \bullet f(\tau^{-1} \bullet v)$.*

Definition 13 (Renaming). *For any linear hypergraph*

$$F = ((T, \leq^T), (S, \leq^S), E, \lambda, \rho, \kappa, \Lambda)$$

and for any bijection $\tau : \mathbb{A} \to \mathbb{A}$ we can apply τ to F to rename it:

$$\tau \bullet F = ((\tau \bullet T, \tau \bullet \leq^T), (\tau \bullet S, \tau \bullet \leq^S), \pi \bullet E, \tau \bullet \lambda, \tau \bullet \rho, \tau \bullet \kappa, \tau \bullet \Lambda).$$

Proposition 14 (Equivariance of hypergraphs). *For any linear hypergraph $F : m \to n$ and permutation $\tau : \mathbb{A} \to \mathbb{A}, \tau \bullet F \equiv F$.*

Proof.

$$h_T(v) = \tau^{-1} \bullet v \qquad h_S(v) = \tau^{-1} \bullet v \qquad h_E(e) = \tau^{-1} \bullet e$$

$\square$

Therefore the definition of F is equivariant under name permutations, so we are justified in renaming vertices and edges 'on the fly'. From here on, when working with linear hypergraphs we will quotient by equivariance.

3.2 Sequential composition

To compose hypergraphs sequentially, we 'redirect' any vertices that connected to the output of the first hypergraph to connect to those originally connected to the input of the second hypergraph. Graphically, we juxtapose the hypergraphs horizontally:

$$F \cdot G : 1 \to 2$$
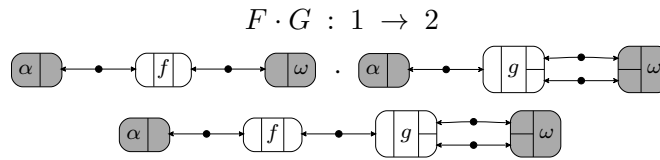
Definition 15 (Sequential composition). *For any two linear hypergraphs $F : m \to n$ and $G : n \to p$ over a signature Σ:*

$$F : m \to n = ((T_F, \leq_F^T), (S_F, \leq_F^S), E_F, \lambda_F, \rho_F, \kappa_F, \Lambda_F)$$
$$G : n \to p = ((T_G, \leq_G^T), (S_G, \leq_G^S), E_G, \lambda_G, \rho_G, \kappa_G, \Lambda_G)$$

with functions s_F, t_F and s_G, t_G respectively, we can define their sequential composition

$$H : m \to p = F \cdot G = ((T_H, \leq_H^T), (S_H, \leq_H^S), E_H, \lambda_H, \rho_H, \kappa_H, \Lambda_H).$$

We delete the outputs of F and the inputs of G. The new set of edges is simply the disjoint union of the edges in F and G.

$$T_H = T_F + (T_G - T_G^{\blacklozenge}) \qquad \leq_H^T = \leq_F^T \oslash \leq_G^T$$

$$S_H = (S_F - S_G^{\blacklozenge}) + S_G \qquad \leq_H^S = \leq_F^S \oslash \leq_G^S$$

$$E_H = E_F + E_G$$

We combine the left and right functions.

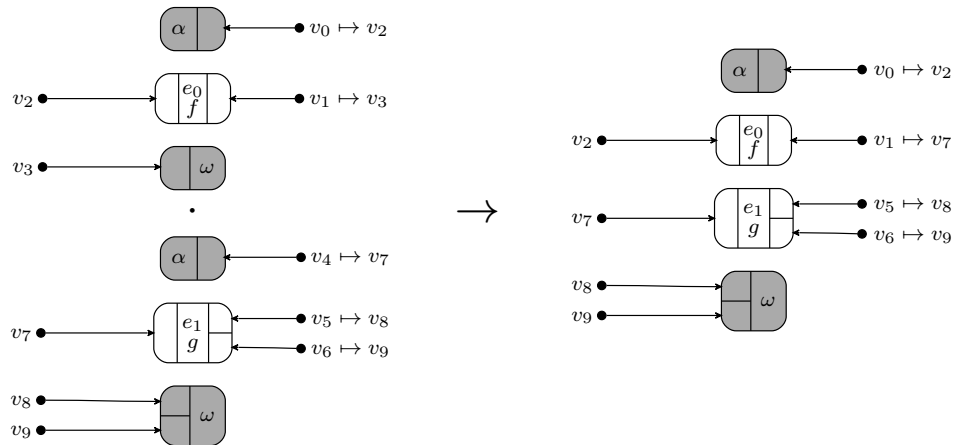$$\lambda_H(v) = (\lambda_F \oplus \lambda_G) \qquad \rho_H(v) = (\rho_F \oplus \rho_G)$$

The connections function maps a vertex connected to the xth output of F to the vertex connected to the xth input of G.

$$\kappa_H(v) = \begin{cases} \kappa_G(\pi_x(T_G^{\blacklozenge})) & \text{if } \kappa_F(v) = \pi_x(S_F^{\blacklozenge}) \\ (\kappa_F \oplus \kappa_G)(v) & \text{otherwise} \end{cases}$$

Finally we take the union of the labelling function.

$$\Lambda_H = \Lambda_F \oplus \Lambda_G$$

This can be drawn formally as follows:

Proposition 16 (Well-formedness of sequential composition)**.** *For any two linear hypergraphs $F : m \to n$ and $G : n \to p$, $F \cdot G$ is a well-formed hypergraph.*

The unit of sequential composition is the identity hypergraph.

Definition 17 (Identity hypergraph)**.** *An identity hypergraph* $\mathsf{id}_n : n \to n$ *over a signature Σ is defined as*

$$n = ((\mathsf{A}, \leq^{\mathsf{A}}), (\mathsf{B}, \leq^{\mathsf{B}}), \emptyset, \lambda, \rho, \kappa, \emptyset)$$

where $\mathsf{A}, \mathsf{B} \subset \mathbb{A}$ *are disjoint,* $|\mathsf{A}| = |\mathsf{B}| = n$ *and*

$$\lambda(v) = \blacklozenge \qquad \rho(v) = \blacklozenge \qquad \kappa(\pi_x(\mathsf{A})) = \pi_x(\mathsf{B}).$$

We usually write id_n simply as n. The identity can be represented graphically as a hypergraph where all vertices are incident on both the input and the output. Below are examples for $n = 1$ and $n = 2$:

3.3 Parallel composition

To compose hypergraphs in parallel, we simply combine their input and output interfaces and leave everything else untouched. Graphically, we can represent this by juxtaposing them vertically:

$$F \otimes G : 1 + 1 \to 1 + 2$$
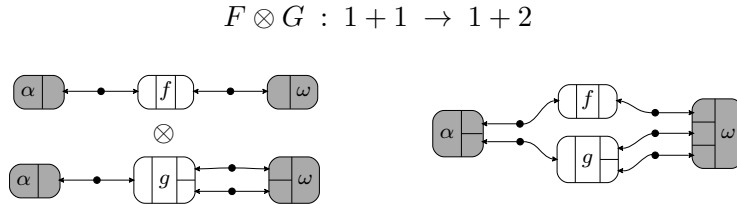
Definition 18 (Parallel composition)**.** *For any two linear hypergraphs $F : m \to n$ and $G : p \to q$ over a signature Σ:*

$$F = ((T_F, \leq_F^T), (S_F, \leq_F^S), E_F, \lambda_F, \rho_F, \kappa_F, \Lambda_F)$$
$$G = ((T_G, \leq_G^T), (S_G, \leq_G^S), E_G, \lambda_G, \rho_G, \kappa_G, \Lambda_G)$$

with source and target functions $\mathsf{s}_F, \mathsf{t}_F$ and $\mathsf{s}_G, \mathsf{t}_G$ respectively, we can define their parallel composition

$$H : m + p \to n + q = F \otimes G = ((T_H, \leq_H^T), (S_H, \leq_H^S), E_H, \lambda_H, \rho_H, \kappa_H, \Lambda_H).$$

Unlike in sequential composition, no vertices need to be deleted in parallel composition. Once again, the new set of edges is the disjoint union of edges from f and g.

$$T_H = T_F + T_G \qquad \leq_H^T \, = \, \leq_F^T \oslash \leq_G^T$$

$$S_H = S_F + S_G \qquad \leq_H^S \, = \, \leq_F^S \oslash \leq_G^S$$

$$E_H = E_F + E_G$$

We again combine the left and right functions.

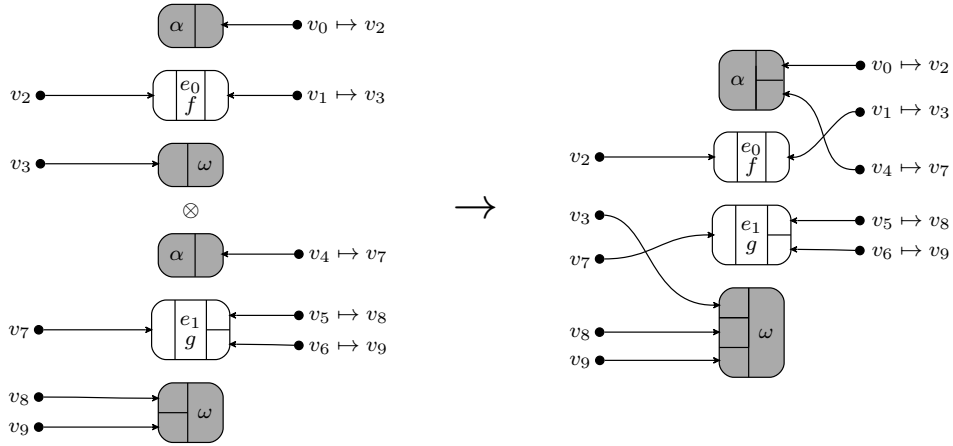$$\lambda_H(v) = (\lambda_F \oplus \lambda_G) \qquad \rho_H(v) = (\rho_F \oplus \rho_G)$$

All connections are preserved from F and G, since we have not deleted any vertices.

$$\kappa_H(v) = \kappa_F \oplus \kappa_G$$

The labelling is also preserved from F and G.

$$\Lambda_H = \Lambda_F \oplus \Lambda_G$$

The formal graphical representation can be seen below.

Proposition 19 (Well-formedness of parallel composition)**.** *For any two linear hypergraphs $F : m \to n$ and $G : p \to q$, $F \otimes G$ is a well-formed linear hypergraph.*

The unit of parallel composition is the empty hypergraph (an identity hypergraph on 0).

Definition 20 (Empty hypergraph)**.** *The empty hypergraph $0 : 0 \to 0$ over a signature Σ is defined as*

$$0 = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset).$$

The empty hypergraph is simply a hypergraph with no vertices or regular edges, just the input and output edge with empty interfaces. Graphically it can be represented by the two empty interfaces.

$- \otimes -$ is a bifunctor, so there may be multiple orders in which we can perform sequential or parallel composition that still result in the same hypergraph.
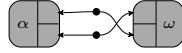
Proposition 21 (Bifunctoriality)**.** *For any $m, n \in \mathbb{N}$, $m \otimes n \equiv m + n$ and for any $F : m \to n$, $G : r \to s$, $H : n \to p$, $K : s \to t$,*

$$F \otimes G \cdot H \otimes K \equiv (F \cdot H) \otimes (G \cdot K).$$
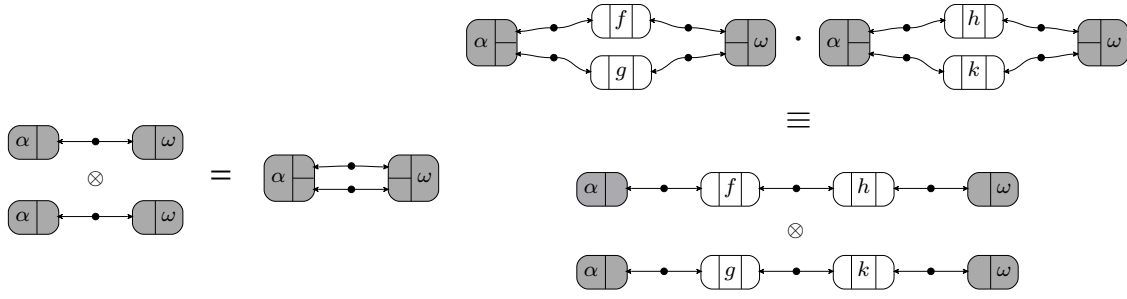
3.4 Symmetry

To swap wires in hypergraphs, we require a new construct, named the *swap* hypergraph. This hypergraph swaps over two wires.

Definition 22 (Swap hypergraph)**.** *The swap hypergraph for two wires* $\times_{1,1}$ *is defined as*

$$\times_{1,1} = ((\{\mathsf{a}, \mathsf{b}\}, \mathsf{a} < \mathsf{b}), (\{\mathsf{c}, \mathsf{d}\}, \mathsf{c} < \mathsf{d}), \emptyset, \lambda, \rho, \kappa, \emptyset)$$

where $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d} \in \mathbb{A}$ *and*

$$\lambda(v) = \blacklozenge \qquad \rho(v) = \blacklozenge \qquad \kappa(\mathsf{a}) = \mathsf{d} \qquad \kappa(\mathsf{b}) = \mathsf{c}$$

By composing multiple copies of the swap hypergraph in sequence and parallel we can build up constructs in which we swap many wires.

Definition 23 (Composite swap)**.** *For any* $m, n \in \mathbb{N}$*, we can define a composite swap hypergraph*

$$\times_{m,n} \ : \ m + n \ \to \ n + m$$

as follows:

$$\times_{0,n} = n$$

$$\times_{m,0} = m$$

$$\times_{1,1} = \times_{1,1}$$

$$\times_{m+1,1} = m \otimes \times_{1,1} \cdot \times_{m,1} \otimes 1$$

12
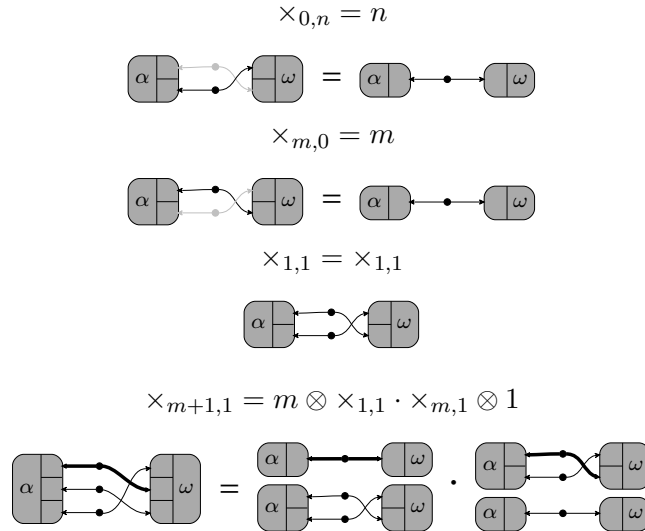
$$\times_{1,n+1} = \times_{1,n} \otimes 1 \cdot n \otimes \times_{1,1}$$

$$\times_{m+1,n+1} = 1 \otimes \times_{1,n} \otimes 1 \cdot \times_{m,n} \otimes \times_{1,1} \cdot n \otimes \times_{m,1} \otimes 1$$

Sometimes it may be preferential to represent composite swaps in a non-inductive way, and instead think in terms of swapping the sets in the input and output interfaces.

Lemma 24 (Alternate swap)**.** *For any $m, n \in \mathbb{N}$, any composite swap hypergraph $\times_{m,n}$ can be written in the form*

$$\times_{m,n} = ((\mathsf{A} + \mathsf{B}, \leq^{\mathsf{A}} \oslash \leq^{\mathsf{B}}), (\mathsf{C} + \mathsf{D}, \leq^{\mathsf{C}} \oslash \leq^{\mathsf{D}}), \emptyset, \lambda, \rho, \kappa, \emptyset)$$

where $\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D} \subset \mathcal{A}$ are disjoint sets, $|\mathsf{A}| = |\mathsf{D}| = m$, $|\mathsf{B}| = |\mathsf{C}| = n$ and

$$\lambda(v) = \blacklozenge \qquad \rho(v) = \blacklozenge \qquad \kappa(\pi_x(\mathsf{A})) = \kappa(\pi_x(\mathsf{D})) \qquad \kappa(\pi_x(\mathsf{B})) = \kappa(\pi_x(\mathsf{C}))$$

Composite swap hypergraphs are *natural*: we can 'push through' hypergraphs composed on the left or the right.

Proposition 25 (Naturality of swap)**.** *For any $m, n, p, q \in \mathbb{N}$, and hypergraphs $F : m \to n$ and $G : p \to q$,*

$$f \otimes g \cdot \times_{n,q} \equiv \times_{m,p} \cdot g \otimes f.$$

3.5 Trace

We can *trace* a hypergraph by connecting some of its outputs to its inputs. Graphically a trace of x wires is represented by bending the first x output and input wires so they join up. The area being traced can be represented by a shaded area.

$$\mathrm{Tr}^1(H) : 1 \to 1$$

Trace is defined recursively. A trace of 0 wires is equal to the original hypergraph.

13

Traces of multiple wires can then be represented by repeatedly tracing with one wire until the desired number has been reached.
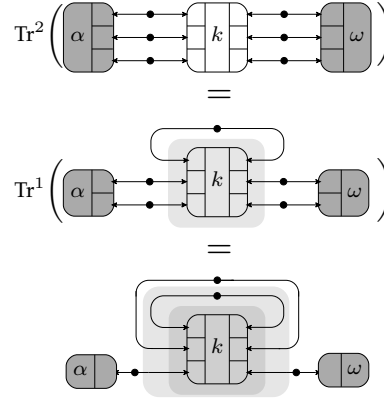
Definition 26 (Trace). *For any linear hypergraph $F : x + m \rightarrow x + n$ over a signature Σ:*

$$F = ((T_F, \leq_F^T), (S_F, \leq_F^S), E_F, \lambda_F, \rho_F, \kappa_F, \Lambda_F)$$

with source and target functions $\mathsf{s}_F, \mathsf{t}_F$ and $\mathsf{s}_G, \mathsf{t}_G$ respectively, we can recursively define its trace of x wires as $Tr^x(f) : m \rightarrow n$ as

$$Tr^0(F) = F$$

$$Tr^{x+1}(F) = Tr^1(Tr^x(F)) \text{ for } x > 0$$

with the base case

$$H = Tr^1(F) = ((T_H, \leq_H^T), (S_H, \leq_H^S), E_H, \lambda_H, \rho_H, \kappa_H, \Lambda_H)$$

defined as follows:

We delete the first input and output vertex, as the wires connecting to these ports have been 'bent around'. The edges are unchanged from the original hypergraph.

$$T_H = T_F - \{\pi_0(T_F^{\blacklozenge})\} \qquad \leq_H^T = \leq_F^T$$

$$S_H = S_F - \{\pi_0(S_F^{\blacklozenge})\} \qquad \leq_H^S = \leq_F^S$$

$$E_H = E_F$$

The other vertices behave as in the original hypergraph.

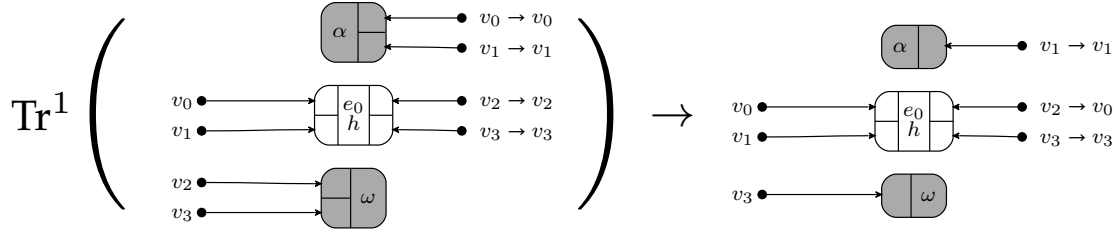$$\lambda_H(v) = \lambda_F \qquad \rho_H(v) = \rho_F$$

The connections function connects the vertex that originally connected to the first output to the vertex originally connected to by the first input, creating a loop.

$$\kappa_H(v) = \begin{cases} \kappa_F(\pi_0(T_F^{\blacklozenge})) & \text{if } \kappa_F(v) = \pi_0(S_F^{\blacklozenge}) \\ \kappa_F(v) & \text{otherwise} \end{cases}$$

The labelling is unchanged.

$$\Lambda_H = \Lambda_F$$

This can be represented formally as follows:

Proposition 27 (Well-formedness of trace). *For any linear hypergraph $F : x + m \to x + n$, $Tr^x(F)$ is a well-formed linear hypergraph.*

3.6 Wire homeomorphisms

The beauty of graphical representations is that constructs such as identities are absorbed into the diagrammatic language. However, we may occasionally wish to explicitly show these identities (e.g. for graph rewriting, see §6, Remark 72). This is achieved by introducing a new class of 'identity' edges, which do not affect the content of the hypergraph. These identity edges can be added (*expansion*) or removed (*smoothing*) at will, as illustrated below. We draw identity edges as a grey diamond.
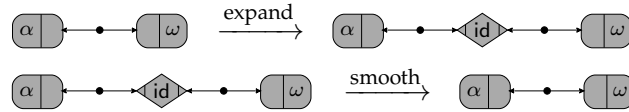
Definition 28 (Homemorphism). *Two hypergraphs F and G that differ only by identity edges are said to be* homeomorphic $F \approx G$.

As with equivariance, we now quotient all of our hypergraphs by homeomorphism, and we always draw the smoothed out version with no identity edges.

4 Soundness

We would like to use hypergraphs as a graphical language for STMCs. In particular we will focus on traced PROPs [21], categories with natural numbers as objects and addition as tensor product. First we consider *soundness*.

We fix a traced PROP **Term**$_\Sigma$ with morphisms freely generated over a signature Σ, and assemble our hypergraphs into the traced PROP **HypTerm**$_\Sigma$, in which the morphisms are hypergraphs of type $m \to n$ over the signature Σ, with composition, symmetry and trace defined as above.

Definition 29 (Interpretation functor). *We define the interpretation functor from terms to hypergraphs as the identity-on-objects traced monoidal functor $[\![-]\!]_\Sigma : \textbf{Term}_\Sigma \to \textbf{HypTerm}_\Sigma$.*

Usually we will omit the signature subscript for clarity. For terms, $[\![-]\!]$ is defined recursively on their syntax. The 'base case' is a generator in Σ, which is translated into a hypergraph with one edge with appropriate arity and coarity. For example, for a generator $f : 2 \to 2$:
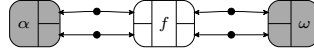
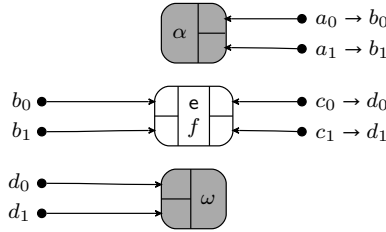Formally, for a generator $f : m \to n$, this is defined as:

$$\llbracket f \rrbracket = ((\mathsf{A} + \mathsf{C}, \leq^{\mathsf{A}} \oslash \leq^{\mathsf{C}}), (\mathsf{B} + \mathsf{D}, \leq^{\mathsf{B}} \oslash \leq^{\mathsf{D}}), \{\mathsf{e}\}, \lambda, \rho, \kappa, \Lambda)$$

where A, B, C, D $\subset \mathcal{A}$ are disjoint, $|\mathsf{A}| = |\mathsf{B}| = m$ and $|\mathsf{C}| = |\mathsf{D}| = n$, $\mathsf{e} \in \mathcal{A}$, and

$$\lambda(v \in \mathsf{A}) = \blacklozenge \qquad \lambda(v \in \mathsf{C}) = \mathsf{e} \qquad \rho(v \in \mathsf{B}) = \mathsf{e} \qquad \rho(v \in \mathsf{D}) = \blacklozenge$$

$$\kappa(\pi_x(\mathsf{A})) = \pi_x(\mathsf{B}) \qquad \kappa(\pi_x(\mathsf{C})) = \pi_x(\mathsf{D}) \qquad \Lambda(\mathsf{e}) = f$$

In the formal graphical notation this is represented as:

Identity and symmetry morphisms translate cleanly into their hypergraph versions (Definitions 17 and 23).

$$\llbracket n \rrbracket = n \qquad \llbracket \times_{m,n} \rrbracket = \times_{m,n}$$

To generate hypergraphs of larger terms, we can combine the morphism, identity and swap hypergraphs using sequential and parallel composition, or by using the trace operator.

$$\llbracket f \cdot g \rrbracket = \llbracket f \rrbracket \cdot \llbracket g \rrbracket \qquad \llbracket f \otimes g \rrbracket = \llbracket f \rrbracket \otimes \llbracket g \rrbracket \qquad \llbracket \mathrm{Tr}^x(f) \rrbracket = \mathrm{Tr}^x(\llbracket f \rrbracket)$$

Proposition 30 (Well-formedness)**.** *For any term f in a traced PROP **Term**$_\Sigma$, $\llbracket f \rrbracket_\Sigma$ is a well-formed hypergraph.*

Proof. Morphism, identity and swap hypergraphs are well-formed, and all other operations involved create well-formed hypergraphs. $\qquad\qquad\square$

To show soundness, we must examine that the axioms of STMCs are satisfied in the language of hypergraphs, as illustrated in Figures 1 and 2.

Definition 31 (Soundness)**.** *Hypergraphs are a sound graphical language for STMCs if, for any morphisms $f, g \in$ **Term**$_\Sigma$, if $f = g$ under the equational theory of the category then their interpreations under $\llbracket - \rrbracket$ are isomorphic $\llbracket f \rrbracket \equiv \llbracket g \rrbracket$*

Theorem 32 (Symmetric traced monoidal category)**.** ***HypTerm**$_\Sigma$ satisfies the axioms of symmetric traced monoidal categories.*

Left identity

Right identity

Associativity

Identity of tensor

Associativity of tensor

Hexagon axiom
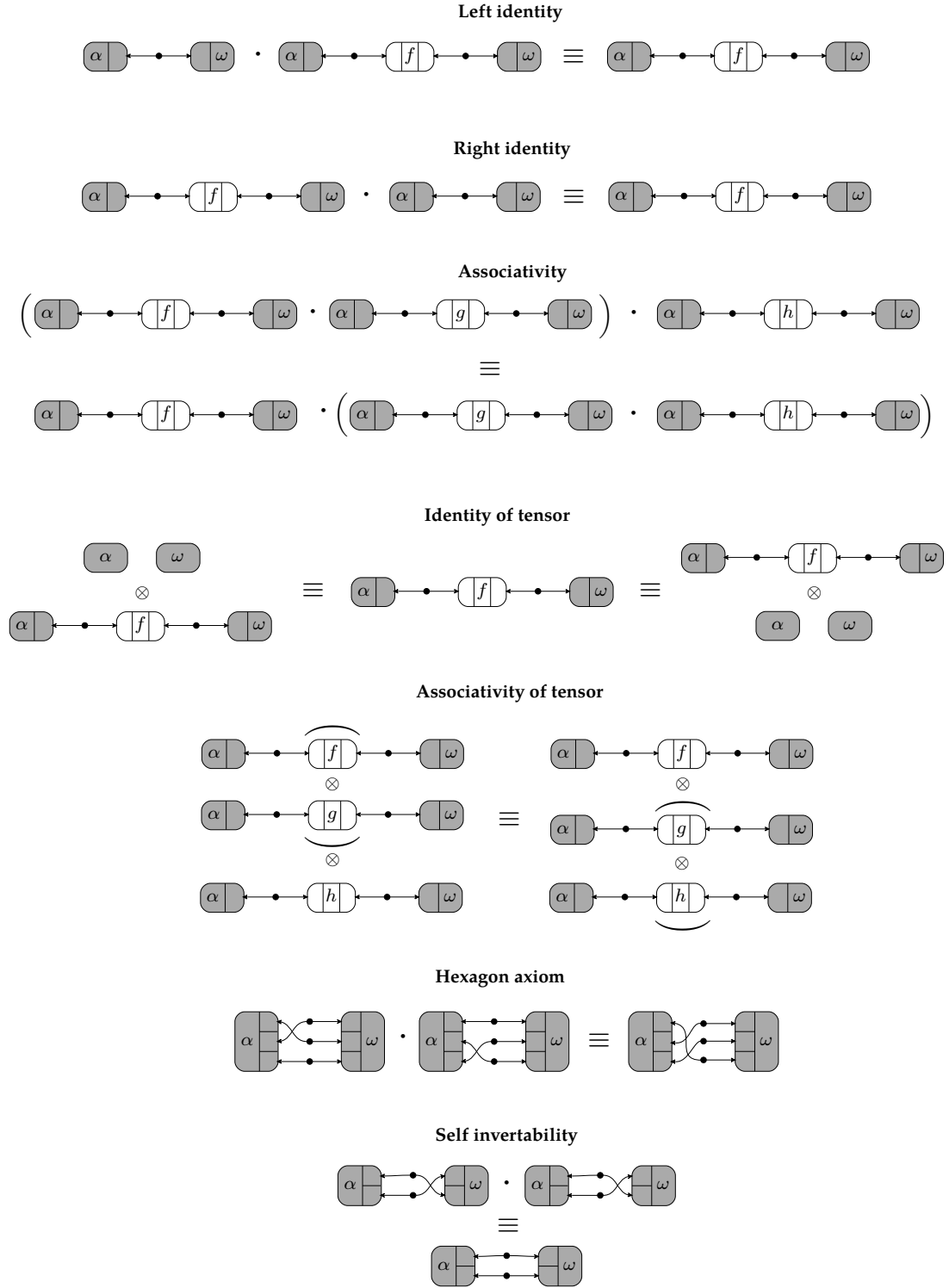

Self invertability

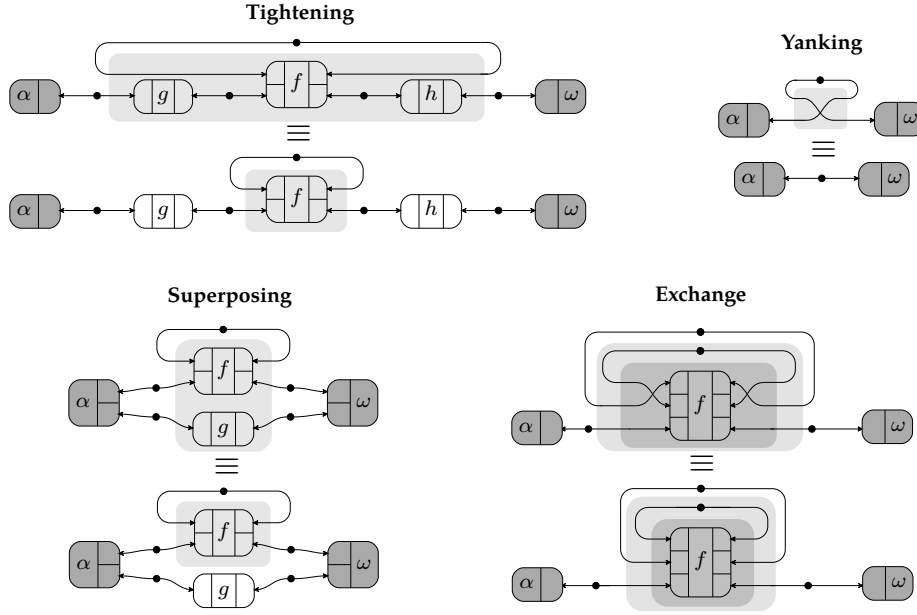Figure 1: Axioms of symmetric monoidal categories represented using hypergraphs.

17

Figure 2: Axioms of symmetric traced monoidal categories represented using hypergraphs.

Proof. Sequential composition produces well-formed hypergraphs (Proposition 16) and satisfies the axioms of categories with the identity hypergraph $\mathrm{id}_n : n \to n$ as the unit of composition for n. Parallel composition produces well-formed hypergraphs (Proposition 19, is a bifunctor (Proposition 21) and satisfies the axioms of (strict) monoidal categories with the empty hypergraph $0 : 0 \to 0$ as the monoidal unit. The swap hypergraph is natural (Proposition 25) and satisfies the axioms of symmetric monoidal categories. The trace operator produces well-formed hypergraphs (Proposition 27) and satisfies the axioms of symmetric traced monoidal categories specified in [14] (tightening, yanking, superposing and exchange). □

Corollary 33 (Soundness). *Hypergraphs are a sound graphical language for STMCs.*

5 Completeness

We are also able to recover categorical terms in an STMC from well-formed hypergraphs.

Definition 34 (Completeness). *Hypergraphs are a complete graphical language for STMCs if for any linear hypergraph F there exists a unique morphism $f \in \mathbf{Term}_\Sigma$, up to the equational theory of the category, such that $[\![f]\!] \equiv F$.*

This is a two stage process. First we show that any well-formed hypergraph has at least one corresponding categorical term (*definability*), then we show that all of these terms are equal in the category (*coherence*).

Definition 35 (Definability). *Hypergraphs are definable if for every well-formed hypergraph F we can retrieve a well-formed categorical equation for which the hypergraph interpretation of that*

term is equivalent to the original graph, i.e. for a candidate term $:$ **HypTerm**$_\Sigma$ $\to$ **Term**$_\Sigma$, *then*

$$[\![\mathsf{term}(F)]\!] \equiv F.$$

The general strategy to retrieve a categorical term from a hypergraph is to use the formal graphical representation, in which all edges are 'stacked', which corresponds to tensoring them together. We then connect wires of opposite polarities by linking them with trace and symmetries.

Example 36. *The hypergraph drawn in the formal notation, on the left, leads to the corresponding categorical term on the right. The procedure will be detailed in the next sections.*

$$Tr^3(\times_{3,2} \cdot \times_{1,1} \otimes 3 \cdot 1 \otimes \times_{2,1} \otimes 1 \cdot \curlywedge \otimes \curlyvee \otimes 2)$$
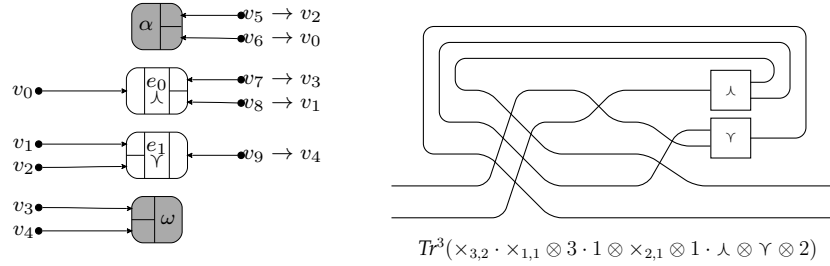
5.1 Stacking

The first step is to fix a total order on the edges, as if we were stacking them. We fix this order $\leq$ globally. The stack operation creates a tensor of the corresponding generators for each edge in the hypergraph.

$$\mathsf{stack}_{\Sigma,\leq} \; : \; \textbf{HypTerm}_\Sigma \; \to \; \textbf{Term}_\Sigma$$

$$\mathsf{stack}_{\Sigma,\leq}(H) = \bigotimes_{e \in (E_H, \leq)} \Lambda(e)$$

5.2 Untangling

With an order $\leq$ on the edges, the hypergraph can be 'tidied up' by adjusting the target and source vertices such that any vertices incident on the same edge are consecutive in their associated orders, with any input and output vertices set to be the 'lowest' and 'highest' in the orders respectively. Untangling is not strictly necessary, but it reduces the size of the output term and simplifies some of the proofs. If we stack the edges and vertices in their orders, fewer crossings of wires and fewer symmetries are required in the end result.

Definition 37. *An untangled hypergraph H is a hypergraph equipped with the fixed order $\leq$ on E such that*

- *for all $e_1, e_2 \in E_H$ if $e_1 < e_2$ then*

  - *for all $v_1 \in \mathsf{s}_H(e_1), v_2 \in \mathsf{s}_H(e_2), v_1 < v_2$*
  - *for all $v_1 \in \mathsf{t}_H(e_1), v_2 \in \mathsf{t}_H(e_2), v_1 < v_2$*
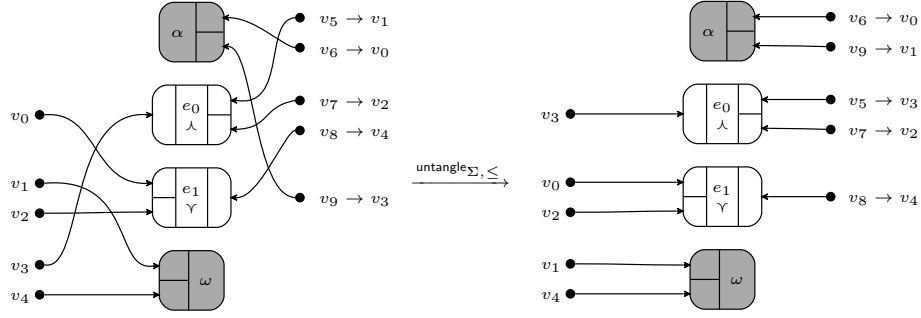
Figure 3: Untangling a hypergraph.

- *For all $v \in T_H^{\blacklozenge}, v' \in T_H - T_H^{\blacklozenge}, v < v'$*

- *For all $v \in S_H^{\blacklozenge}, v' \in S_H - S_H^{\blacklozenge}, v' < v$*

We can untangle a hypergraph with the operation $\mathsf{untangle}_{\Sigma, \leq} : \mathbf{HypTerm}_\Sigma \to \mathbf{HypTerm}_\Sigma$ This is illustrated in Figure 3.

Since the input and output 'edges' do not exist in the stack of boxes, vertices incident on the these edges are set to be the lowest and highest elements in the new order respectively.

Lemma 38. *For any linear hypergraph F and order on its edges $\leq$, there is a unique untangled variant $\hat{F} = \mathsf{untangle}_{\leq}(F)$ such that $\hat{F} \equiv F$.*

Proof. Untangling only affects the orders of the vertices, so the connections and labelling condition are satisfied immediately. The sources and targets conditions also hold because the order of vertices incident on the same edge is preserved. The ordering is unique for a given $\leq$ since there is only one order the vertices incident on each edge can take. $\square$

5.3 Shuffling

When we trace around the wires in the next step, we need to ensure that the outputs of the boxes are all connected to the correct inputs. Therefore we need a 'shuffle' construct consisting of symmetries that will shuffle the wires into the correct order. This construct is defined recursively over the number of vertices. First the input that connects to the 'first' output is determined, by finding the position of $\kappa^{-1}(\pi_0(S))$ in T. A symmetry pulling this wire up to the top is then defined. This wire is now in the correct place and is of no further concern to us. We then recursively perform $\mathsf{shuffle}$ on the remaining source and target vertices until none remain. The procedure is demonstrated in Figure 4 and the algorithm is presented in Algorithm 1.

We now need to show the correctness of the shuffle construct. Namely, we must ensure that its 'input-output' connectivity reflect the connections between target and source vertices in the hypergraph. Since the shuffle construct is comprised only of identities and symmetries, we can effectively 'follow the wires' from left to right and assert that the connections in the hypergraph also hold in the construct.

Lemma 39 (Correctness of shuffle)**.** *For any linear hypergraph F, derived shuffle construct* $\mathsf{shuffle}(F) : |T_F| \to |T_F|$, *and permutation* $p : [|T_F|] \to [|T_F|]$ *such that for any* $x \in [|T_F|]$,

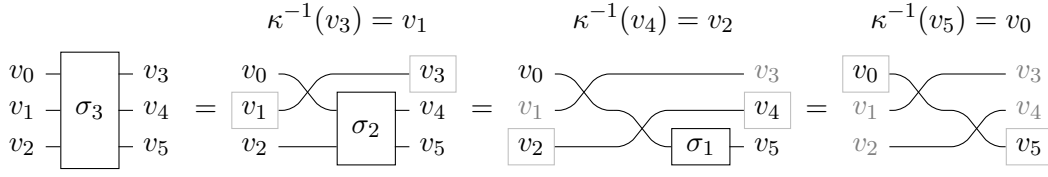$$\kappa^{-1}(v_3) = v_1 \qquad\qquad \kappa^{-1}(v_4) = v_2 \qquad\qquad \kappa^{-1}(v_5) = v_0$$

Figure 4: An example of the shuffle construction (represented by σ_x, where x is the number of source vertices) step-by-step, for $T = \{v_0, v_1, v_2\}$ and $S = \{v_3, v_4, v_5\}$.

Function shuffle$_{\Sigma,\leq}$ $(((T, \leq^T), (S, \leq^S), E, \lambda, \rho, \kappa, \Lambda))$
   | **if** $|S| = 0$ **then**
   | | **return** 0;
   | **end**
   | $v_s \leftarrow \pi_0(S)$;
   | $v_t \leftarrow \kappa^{-1}(v_s)$;
   | $\mathsf{i} \leftarrow x$ where $\pi_x(T) = v_t$;
   | $\mathsf{j} \leftarrow |T| - \mathsf{i} - 1$;
   | $\mathsf{f} \leftarrow \times_{\mathsf{i},1} \otimes \mathsf{j}$;
   | $T \leftarrow T - v_t$;
   | $S \leftarrow S - v_s$;
   | **return** $\mathsf{f} \cdot 1 \otimes$ shuffle$_{\Sigma,\leq}$ $(((T, \leq^T), (S, \leq^S), E, \lambda, \rho, \kappa, \Lambda))$;
end

Algorithm 1: Defining the shuffle construct.

$\kappa(\pi_x(T_F)) = \pi_{p(x)}(S_F)$, *then for some* $\bar{v}$: $0 \rightarrow n = v_0 \otimes v_1 \otimes \cdots \otimes v_{n-1}$, $\bar{v} \cdot \sigma_n = v_{p^{-1}(0)} \otimes v_{p^{-1}(1)} \otimes \cdots \otimes v_{p^{-1}(n-1)}$.

Proof. This is by induction on n. For $n < 2$ the statement holds trivially. For $n = 2$, there are two cases: $\{0 \mapsto 0, 1 \mapsto 1\}$ and $\{0 \mapsto 1, 1 \mapsto 0\}$. By naturality of symmetry, $\bar{v} \cdot \sigma_2$ is equal to $v_0 \otimes (v_1 \cdot \sigma_1) = v_0 \otimes v_1$ in the former and $v_1 \otimes (v_0 \cdot \sigma_1) = v_1 \otimes v_0$, so for both cases the statement holds. For $n > 2$, $\bar{v} \cdot \sigma_n = v_x \otimes (v_0 \otimes \cdots \otimes v_{x-1} \otimes v_{x+1} \otimes \cdots \otimes v_{n-1} \cdot \sigma_{n-1})$ where $x = p^{-1}(0)$ by naturality of symmetry. Therefore the first element of the tensor is correct, and the remaining elements follow by inductive hypothesis. $\square$

Since there is no input 'box', we also need to precompose the shuffle construct with another symmetry to pull the input wires to the 'top' of the term.

5.4 Tracing

The final step is to trace around all of the wires coming out of the edges. Since the inputs do not need to be traced around, we only need to trace with $|T - T^\blacklozenge|$ wires. Any wires representing the output of the term will be shuffled to the bottom of the term and subsequently will not be traced.

21

5.5 Recovering a term

To retrieve a term from a hypergraph H with edge order $\leq$, we simply trace the composition of the corresponding shuffle construct and edge stack:

Definition 40. *We define the the identity-on-objects traced monoidal functor*

$$\mathsf{term}_{\Sigma,\leq} \ : \ \boldsymbol{HypTerm}_{\Sigma} \ \to \ \boldsymbol{Term}_{\Sigma}$$

defined for a given edge order $\leq$ as

$$\mathsf{term}_{\leq}(H) := Tr^{|T-T^{\blacklozenge}|}(\times_{|T-T^{\blacklozenge}|,|T^{\blacklozenge}|} \cdot \mathsf{shuffle}_{\leq}(\mathsf{untangle}_{\leq}(H)) \cdot \mathsf{stack}_{\leq}(\mathsf{untangle}_{\leq}(H)) \otimes |S^{\blacklozenge}|).$$

To conclude definability we must be able to return to the original hypergraph. First we show some lemmas that hold in any STMC.

Lemma 41 (Staging). *Any morphism $f \in \boldsymbol{Term}_{\Sigma}$ can be written as in the form $f = f_0 \cdot f_1 \cdot \cdots \cdot f_{x-1}$, where f_i is a tensor containing only one non-identity morphism, $f_i = \mathsf{id}_m \otimes k \otimes \mathsf{id}_n$.*

Proof. By left/right identity and functoriality. $\qquad\square$

Lemma 42 (Composite symmetry). *Any symmetry $\times_{m,n} \in \boldsymbol{Term}_{\Sigma}$ can be expressed as a combination of multiple symmetries $\times_{1,1}$ and identities.*

Proof. By the hexagon axiom. $\qquad\square$

The derived term is constructed from multiple terms containing no edges, which serve to 'jumble' the wires up. This is reflected in their connections permutations.

Lemma 43 (Edgeless composition). *For any two hypergraphs $F \ : \ m \ \to \ n$ and $G \ : \ n \ \to \ p$, where*

$$F = ((T_F, \leq_F^T), (S_F, \leq_F^S), E_F, \lambda_F, \rho_F, \kappa_F)$$
$$G = ((T_G, \leq_G^T), (S_G, \leq_G^S), E_G, \lambda_G, \rho_G, \kappa_G)$$

and $E_F = E_G = \emptyset$, if we have permutations $p \ : \ [|T_F|] \ \to \ [|T_F|]$ and $q \ : \ [|T_G|] \ \to \ [|T_G|]$ such that $\kappa_F(\pi_x(T_F)) = \pi_{p(x)}(S_F)$ and $\kappa_G(\pi_x(T_G)) = \pi_{q(x)}(S_G)$, then for their composition

$$H = F \cdot G = ((T_H, \leq_H^T), (S_H, \leq_H^S), E_H, \lambda_H, \rho_H, \kappa_H)$$

we have that $\kappa(\pi_x(T_H)) = \pi_{(q \circ p)(x)}(S_H)$.

Proof. By definition of sequential composition. $\qquad\square$

We first tackle the shuffle construct on its own.

Lemma 44 (Definability of shuffle). *For any shuffle construct $\sigma_n \ : \ n \ \to \ n$ and its corresponding interpretation $\llbracket \sigma \rrbracket \ = \ ((T, \leq^T), (S, \leq^S), E, \lambda, \rho, \kappa)$, and permutation $p \ : \ [n] \ \to \ [n]$ such that for some $\bar{v} \ : \ 0 \ \to \ n = v_0 \otimes v_1 \otimes \cdots \otimes v_{n-1}$, $\bar{v} \cdot \sigma = v_{p(0)} \otimes v_{p(1)} \otimes \cdots \otimes v_{p(n-1)}$, then $\kappa(\pi_x(T)) = \pi_{p(x)}(S)$.*

Proof. We first use staging and composite symmetry to arrange the shuffle construct into 'slices' where each slice contains exactly one symmetry $\times_{1,1}$. We then perform induction on n. For $n < 2$ the statement follows trivially. For $n = 2$, we examine the two cases $\{0 \mapsto 0, 1 \mapsto 1\}$ and $\{0 \mapsto 1, 1 \mapsto 0\}$ as in correctness of shuffle (Lemma 39). Their interpretations are illustrated below.
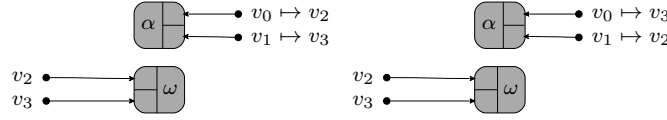
For both cases the statement holds. For $n > 2$, we split the definition of σ_n into two parts: $\sigma'_n = \times_{x,1} \otimes (n - 1 - x)$ where $x = p^{-1}(0)$, and $\sigma''_n = 1 \otimes \sigma_{n-1}$. In $[\![\sigma'_n]\!]$, $\kappa(\pi_x(T)) = \pi_0 \circ S$ by definition of the swap hypergraph. In $[\![\sigma''_n]\!]$, $\kappa(\pi_0(T)) = \pi_0(S)$ by definition of parallel composition. Therefore in $[\![\sigma_n]\!]$, $\kappa(\pi_x(T)) = \pi_0(S)$ by edgeless composition. So for the first vertex in T the statement holds. For the remaining vertices we apply the inductive hypothesis to σ_{n-1} and add one to the indices of each vertex, since it is tensored with an identity wire. $\qquad\square$

Finally, we can take on the entire term.

Proposition 45 (Definability). *For any linear hypergraph F with order $\leq$, where*

$$F = ((T_F, \leq_F^T), (S_F, \leq_F^S), E_F, \lambda_F, \rho_F, \kappa_F, \Lambda_F)$$

$$G = [\![\mathsf{term}_{\Sigma, \leq}(F)]\!] = ((T_G, \leq_G^T), (S_G, \leq_G^S), E_G, \lambda_G, \rho_G, \kappa_G, \Lambda_G)$$

then $F \equiv G$.

Proof. We map the sources and targets of edges and the interface in F to the corresponding vertices in $[\![\mathsf{term}_{\Sigma, \leq}(F)]\!]$. By definition of parallel composition, $[\![\mathsf{term}_{\Sigma, \leq}(F)]\!]$ will also be untangled. The labelling condition is immediate, so we only need to consider the connections condition. Let p be the permutation $[|T_F|] \to [|T_F|]$ such that $\kappa_F(\pi_x(T_F)) = \pi_{p(x)}(S_F)$ and q be the same but for G. For the connections condition to be satisfied, these permutations must be the same.

We examine the permutation q. There are two classes of target vertices to consider: vertices that are inputs and those that are not. For the former, the xth input will be connected to $\pi_{p(x)}(S)$ by correctness and definability of shuffle, and edgeless composition of the initial swap with the shuffle. Since the input vertices are the lowest in the vertex order, the xth non-input is equal to $\pi_{x+|T^\blacklozenge|}(T)$. This will be connected to $\pi_{p(x+|T^\blacklozenge|)}(S)$ by definition of trace and edgeless composition of the initial swap with the shuffle. Therefore the connections permutation is exactly that of p. $\qquad\square$

5.6   Coherence

As there may be multiple orders in which we can stack the edges, there are multiple terms that we can retrieve from term. For *coherence* these need to all be equal in the category.

Definition 46 (Coherence). *Hypergraphs are coherent if, for any well-formed hypergraph F and any two orders on its edges $\leq_1$, $\leq_2$, $\mathsf{term}_{\Sigma, \leq_1}(F) = \mathsf{term}_{\Sigma, \leq_2}(F)$ by axioms of STMCs.*

Fortunately, we need not necessarily consider every possible order of edges on the hypergraph. We simply need to consider switching two consecutive edges while retaining the others, e.g. $x < f < g < y$ becomes $x < g < f < y$. We can then swap any two edges in the set by propogating these swaps throughout the entire set. To enable us to do this, we can combine a tensor of edge boxes into one single box:

Lemma 47 (Combination). *For any tensor $t = \bigotimes\limits_{i=0}^{m} a_i \otimes \bigotimes\limits_{i=0}^{n} b_i \otimes \bigotimes\limits_{i=0}^{p} c_i$ we can rewrite it as $t' = a \otimes \bigotimes\limits_{i=0}^{n} b_i \otimes c$ where $a : \sum\limits_{i=0}^{m} \mathsf{dom}(a_i) \to \sum\limits_{i=0}^{m} \mathsf{cod}(a_i)$ and $c : \sum\limits_{i=0}^{p} \mathsf{dom}(c_i) \to \sum\limits_{i=0}^{p} \mathsf{cod}(c_i)$.*

We need a lemma to show that we can transform the shuffle construct for a given order into one for a different order by adding appropriate symmetries on either side, reflecting that the edge boxes have now swapped over.

Lemma 48 (Coherence of shuffle). *For any untangled hypergraph H, two orders $\leq_1, \leq_2$ on its edges in which only two consecutive elements $e_1 : n' \to n$ and $e_2 : p' \to p$ have been swapped, and shuffle constructs*

$$\mathsf{shuffle}_{\Sigma, \leq_1}(H) : |T^{\blacklozenge}| + m + n + p + q \to m' + n' + p' + q' + |S^{\blacklozenge}|$$

$$\mathsf{shuffle}_{\Sigma, \leq_2}(H) : |T^{\blacklozenge}| + m + p + n + q \to m' + p' + n' + q' + |S^{\blacklozenge}|$$

then

$$\mathsf{shuffle}_{\Sigma, \leq_1} = |T^{\blacklozenge}| \otimes m \otimes \times_{n,p} \otimes q \cdot \mathsf{shuffle}_{\Sigma, \leq_2} \cdot m' \otimes \times_{p',n'} \otimes q' \otimes |S^{\blacklozenge}|$$
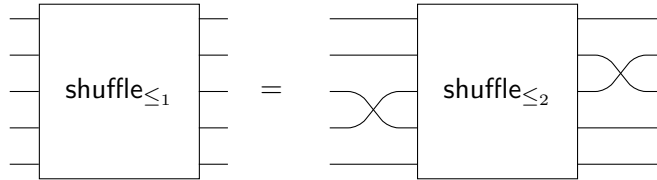

Proof. We show equality by asserting for every 'input' to the term, it always leads to the same 'output'. In the diagram below, each box represents a 'bundle' of wires. By correctness of shuffle (Lemma 39), we know that if there exists a permutation p such that $\kappa(\pi_x(T)) = \pi_{p(x)}(S)$ then $v_0 \otimes v_1 \otimes \cdots \otimes v_{n-1} \cdot \sigma_n = v_{p^{-1}(0)} \otimes v_{p^{-1}(1)} \otimes \cdots \otimes v_{p^{-1}(n-1)}$. Since the shuffle construct is defined by an untangled hypergraph, this means its outputs will be bundles of wires corresponding to the sources of each edge. This means that the LHS and RHS will only differ by the two bundles of wires (labelled n' and p' on the diagram below). We can therefore use naturality of symmetry to to show both expressions are equal.
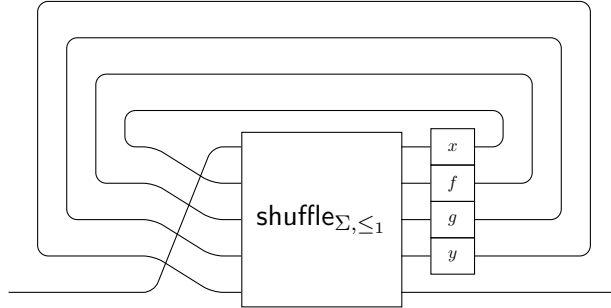
LHS

24
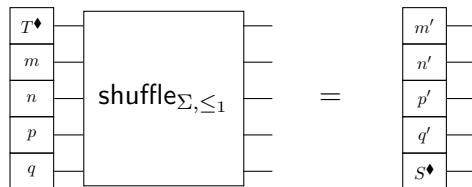
RHS

$\square$
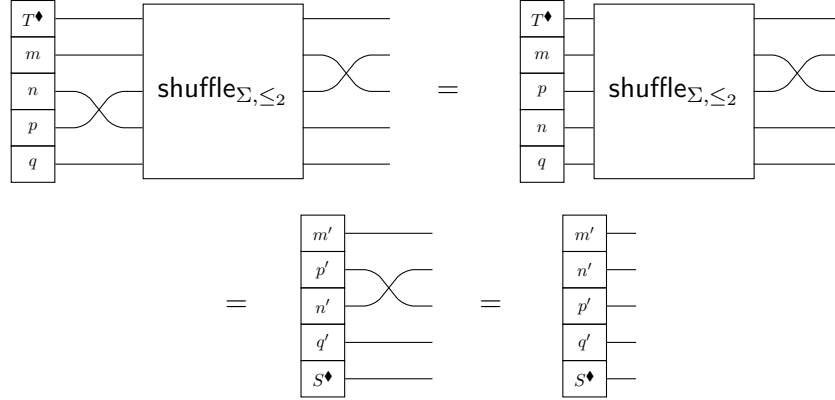
Lemma 49 (Coherence for two edges)**.** *For any linear hypergraph f and two orders $\leq_1, \leq_2$ on its edges which differ only by the swapping of two consecutive elements,* $\mathsf{term}_{\Sigma,\leq_1} = \mathsf{term}_{\Sigma,\leq_2}$.
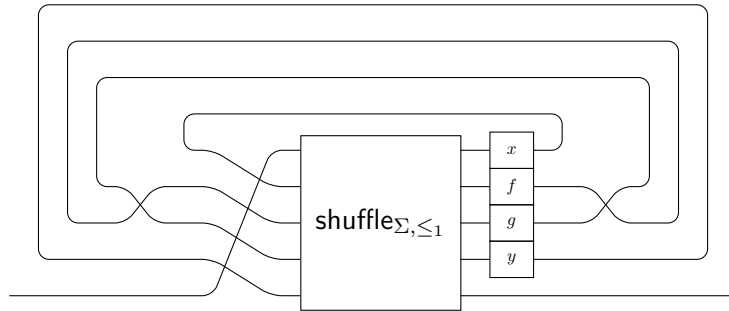
Proof. Any term generated by $\mathsf{term}_{\Sigma,\leq}$ can be rewritten in the form

$$\mathrm{Tr}^{|T - T^\blacklozenge|}(\times_{|T - T^\blacklozenge|, |T^\blacklozenge|} \cdot \mathsf{shuffle}_{\Sigma,\leq} \cdot |T - T^\blacklozenge| \otimes f \otimes g \otimes y \otimes |S^\blacklozenge|)$$
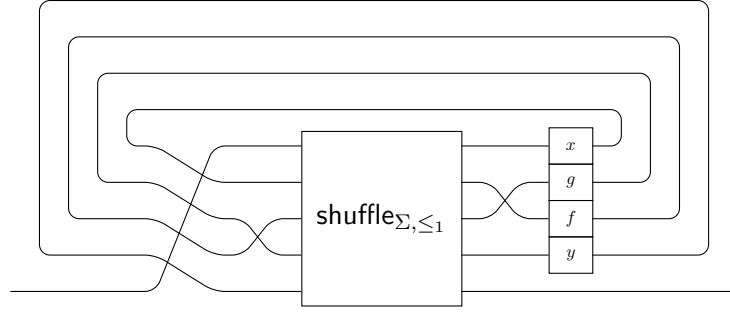
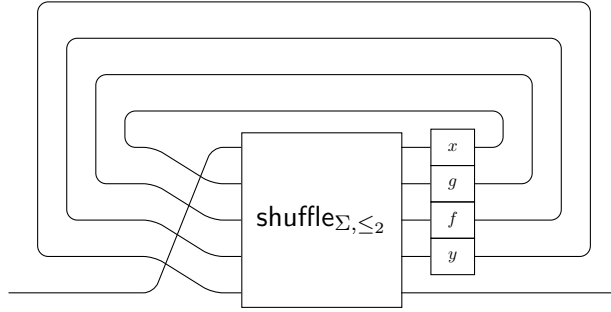by using combination (Lemma 47). So we have a term of the form:

By exchange:

By naturality of symmetry and functoriality:

25

By coherence of shuffle (Lemma 48):

$\square$

We can then extend this lemma to obtain our final coherence result.

Proposition 50 (Coherence). *For all orderings of edges $\leq_x$ on a hypergraph f,*

$$\mathsf{term}_{\Sigma,\leq_1}(f) = \mathsf{term}_{\Sigma,\leq_2}(f) = \cdots = \mathsf{term}_{\Sigma,\leq_x}(f)$$

Proof. By repeatedly applying Lemma 49 until the desired order is obtained. $\square$

Since we now know that the chosen edge order is irrelevant, we show that we can return to the original term after translating it into a hypergraph.

Lemma 51 (Composite definability). *For any two hypergraphs F, G and $x \in \mathbb{N}$,*

$$\mathsf{term}_{\Sigma,\leq}(F \cdot G) = \mathsf{term}_{\Sigma,\leq}(F) \cdot \mathsf{term}_{\Sigma,\leq}(G)$$
$$\mathsf{term}_{\Sigma,\leq}(F \otimes G) = \mathsf{term}_{\Sigma,\leq}(F) \otimes \mathsf{term}_{\Sigma,\leq}(G)$$
$$\mathsf{term}_{\Sigma,\leq}(Tr^x(F)) = Tr^x(\mathsf{term}_{\Sigma,\leq}(F))$$

Proof. By sliding and yanking. $\square$

Lemma 52 (Global trace). *For any morphism $f \in \mathbf{Term}_\Sigma$, we can represent it as $Tr^x(\hat{f})$, where $\hat{f}$ is a morphism containing no trace.*

Proof. By superposing and tightening. $\square$

Proposition 53 (Inverse of $[\![-]\!]$). *For any morphism $f \in \mathbf{Term}_\Sigma$ and chosen edge order $\leq$ on $[\![f]\!]$, $\mathsf{term}_{\Sigma,\leq}([\![f]\!]) = f$.*

Proof. The term f is freely generated over Σ, so by staging and global trace, it will be of the form $\text{Tr}^x(f_0 \cdot f_1 \cdot \cdots \cdot f_{n-1})$, where each f_i is of the form $\text{id}_p \otimes k \otimes \text{id}_q$. Therefore by definition of $[\![-]\!]$, $[\![f]\!] = \text{Tr}^x([\![f_0]\!] \cdot [\![f_1]\!] \cdot \cdots \cdot [\![f_{n-1}]\!])$ and by composite definability, $\text{term}([\![f]\!]) = \text{Tr}^x(\text{term}([\![f_0]\!]) \cdot \text{term}([\![f_1]\!]) \cdot \cdots \cdot \text{term}([\![f_{x-1}]\!]))$. By sliding and yanking, each $\text{term}([\![f_i]\!]) = f_i$. Therefore by composition, $\text{term}([\![f]\!]) = f$. $\qquad\square$

Theorem 54 (Completeness). *For any linear hypergraph $F \in \textbf{LHyp}_\Sigma$ there exists a unique morphism $f \in \textbf{Term}_\Sigma$, up to the equations of the STMC, such that $[\![f]\!] = F$.*

Proof. By definability (Proposition 45) we can recover a set of terms in a STMC and by coherence (Proposition 50) these terms are equal by axioms of STMCs. $\qquad\square$

6 Graph rewriting

For standard STMCs, reasoning diagrammatically using isomorphism of diagrams works well, as the axioms are absorbed into the graphical notation. However, we often wish to add extra structure to our categories, with associated axioms. To solve this problem in the language of terms we use *term rewriting*.

Definition 55 (Subterm). *For any morphisms $f, g \in \textbf{Term}_\Sigma$, we say that g is a subterm of f if there exists $\hat{f}_1, \hat{f}_2 \in \textbf{Term}_\Sigma$ and $n, x \in \mathbb{N}$ such that $f = \text{Tr}^x(\hat{f}_1 \cdot \text{id}_n \otimes g \cdot \hat{f}_2)$, where $\hat{f}_1, \hat{f}_2$ do not contain any traces.*

Definition 56 (Term rewriting). *A rewrite rule in $\textbf{Term}_\Sigma$ is a pair $\langle l, r \rangle$ where $l, r : X \to Y$ are terms in $\textbf{Term}_\Sigma$ with the same domain and codomain. We sometimes write rules as $\langle l, r \rangle : X \to Y$. A rewriting system $\mathcal{E}$ is a set of rewrite rules. We can perform a rewrite step in $\mathcal{E}$ for two terms g and h (written $g \twoheadrightarrow_\mathcal{E} h$) if there exists rewriting rule $\langle l, r \rangle \in \mathcal{E}$ such that l is a subterm of g, i.e. we can write f in the form $\text{Tr}^x(\hat{f}_1 \cdot \text{id} \otimes l \cdot \hat{f}_2)$ for some trace-free terms $\hat{f}_1$ and $\hat{f}_2$.*


As is often the case in the algebraic realm, it can be difficult to identify the occurrence of l in f due to functoriality. However, unlike with the axioms of STMCs, these extra axioms are not an intrinsic part of our diagrams. We can no longer rely purely on hypergraph isomorphisms. Fortunately, we can generalise term rewriting to *graph rewriting*. First we must formalise the notion of a subgraph of a linear hypergraph.

Definition 57 (Subgraph). *For any linear hypergraphs $F, G \in \textbf{LHyp}_\Sigma$, we say that G is a subgraph of F if there exists an embedding monomorphism $G \to F$.*

Intuitively, to transform some subgraph L into a larger graph G we apply a sequence of operations.

Lemma 58 (Operation monomorphisms)**.** *Sequential composition, $(- \cdot F, F \cdot -$ for any hypergraph F), parallel composition $(- \otimes F, F \otimes -$ for any hypergraph F), and trace $(Tr^x(-)$ for any $x \in \mathbb{N})$ can be expressed as monomorphisms in $\mathbf{LHyp}_\Sigma$.*

Proof. For sequential and parallel composition this is immediate as we simply embed the operand into the result. For trace, we must introduce a homeomorphism in the traced wire so that we do not merge the vertices being traced, as illustrated below. The numbers represent the images of the vertices under the monomorphism.

$\square$
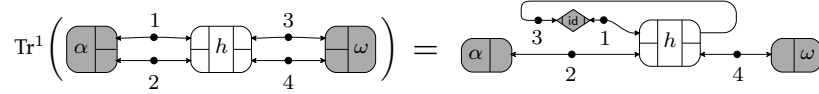
Lemma 59. *For any monomorphism $m : G \to F \in \mathbf{LHyp}_\Sigma$, there exist linear hypergraphs F_1, F_2 and $n, x \in \mathbb{N}$ such that $m = Tr^x(-) \circ (F_2 \cdot -) \circ (- \cdot F_1) \circ (\mathsf{id}_n \otimes -)$.*

Proof. We apply the sequence of operations that transforms G into F. $\square$

Lemma 60. *For any morphisms $f, g \in \mathbf{Term}_\Sigma$, g is a subterm of f iff $[\![g]\!]$ is a subgraph of $[\![f]\!]$.*

Proof. For $(\Rightarrow)$ we assume that g is a subterm of f. Therefore we know that we can write f in the form of Definition 55. By definition of $[\![-]\!]$, $[\![f]\!] = Tr^x([\![\hat{f}_1]\!] \cdot [\![\mathsf{id}_n]\!] \otimes [\![g]\!] \cdot [\![\hat{f}_2]\!])$. As we can express all operations as monomorphisms (Lemma 58), there exists a monomorphism $[\![g]\!] \to [\![f]\!]$, namely $Tr^x(-) \circ ([\![\hat{f}_1]\!] \cdot -) \circ (- \cdot [\![\hat{f}_2]\!]) \circ ([\![\mathsf{id}_n]\!] \otimes -)$.

For $(\Leftarrow)$ we assume that there exists an embedding monomorphism $m : [\![g]\!] \to [\![f]\!]$. Therefore by Lemma 59, there exists hypergraphs F_1, F_2 and $n, x \in \mathbb{N}$ such that $m = Tr^x(-) \circ (F_2 \cdot -) \circ (- \cdot F_1) \circ (\mathsf{id}_n \otimes -)$. This means that $[\![f]\!] = Tr^x(F_2 \cdot \mathsf{id}_n \otimes [\![g]\!] \cdot F_1)$. We apply the definability algorithm to both sides:

$$
\begin{aligned}
\mathsf{term}([\![f]\!]) = f &= \mathsf{term}(Tr^x(F_2 \cdot \mathsf{id}_n \otimes [\![g]\!] \cdot F_1)) && \text{Inverse of } [\![-]\!] \\
&= Tr^x(\mathsf{term}(F_2) \cdot \mathsf{id}_n \otimes \mathsf{term}([\![g]\!]) \cdot \mathsf{term}(F_1)) && \text{Composite definability} \\
&= Tr^x(\mathsf{term}(F_2) \cdot \mathsf{id}_n \otimes g \cdot \mathsf{term}(F_1)) && \text{Inverse of } [\![-]\!]
\end{aligned}
$$

Therefore there exist $f_1 = \mathsf{term}(F_1)$ and $f_2 = \mathsf{term}(F_2)$ such that $f = Tr^x(f_1 \cdot \mathsf{id}_n \otimes g \cdot f_2)$, and thus g is a subterm of f.

$\square$

One of the most popular graph rewriting approaches is known as double pushout (DPO) rewriting [9], and we shall now recall some of its basic notions in relation to linear hypergraphs. We use an extension of the traditional definition that introduces an 'interface' [4].

Definition 61 (DPO)**.** *A DPO rewrite rule $\langle L, R \rangle$ is a span $L \leftarrow K \to R$ in $\mathbf{LHyp}_\Sigma$. A DPO rewriting system $\mathcal{R}$ is a set of DPO rules $\mathcal{R}$. For $G, H \in \mathbf{LHyp}_\Sigma$, we write $G \rightsquigarrow_\mathcal{R} H$ if there exists span $L \leftarrow K \to R \in \mathcal{R}$ and cospan $K \to C \leftarrow J \in \mathbf{LHyp}_\Sigma$ such that the diagram below commutes, and the two squares are pushouts.*

28

$$
\begin{array}{ccccc}
L & \longleftarrow & K & \longrightarrow & R \\
\downarrow & \ulcorner\!\lnot & \downarrow & \ulcorner & \downarrow \\
G & \longleftarrow & C & \longrightarrow & H \\
& \nwarrow & \uparrow & \nearrow & \\
& & J & &
\end{array}
$$
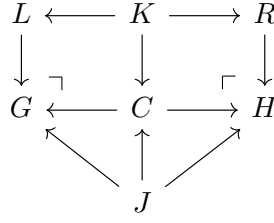
For a term rewriting system $\mathcal{E} \in \mathbf{Term}_\Sigma$, we write $[\![\mathcal{E}]\!]$ as the translation of its equations to spans in $\mathbf{LHyp}_\Sigma$ using $[\![-]\!]$. To perform rewrite rule $\langle L, R \rangle$ in some larger graph G, a *matching* $p : L \to G$ must first be identified. Its *pushout complement* $K \to C \to G$ and the pushout $C \to H \leftarrow R$ can then be computed, yielding us the graph H: the graph G with L replaced by R.

Intuitively, the pushout complement is G with L removed. That is to say, any vertices in G that connect to L will instead point to the interface ♦ in C. In our diagrams, we draw these connections to the interface separately to draw attention to them.

6.1 Adhesive categories

Not all structures are compatible with DPO rewriting. For example, the pushout complement may not be unique. For a successful rewrite system we need to ensure that the DPO approach is always well-defined.

A common framework in which to perform graph rewriting is that of *adhesive categories*, introduced by Lack and Sobociński [22]. The key property of these categories is that pushout complements are always unique, if they exist. Additionally, they also enjoy a local Church-Rosser theorem and a concurrency theorem.

However, our category of hypergraphs $\mathbf{LHyp}_\Sigma$ is not adhesive, as our conditions on the connectivity of vertices means that not all pushouts along monomorphisms exist. Fortunately, there is the notion of *partial adhesive categories* [19], categories that are embedded inside a category that *is* adhesive. These categories 'inherit' enough adhesivity for us to perform graph rewrites on certain classes of spans.

Definition 62 (Partial adhesive category). *A category $\mathcal{P}$ is called a* partial adhesive category *if it is a full subcategory of an adhesive category $\mathcal{A}$ and the inclusion functor $I : \mathcal{P} \to \mathcal{A}$ preserves monomorphisms.*

We already have a candidate for this adhesive category $\mathcal{A}$, the category of simple hypergraphs $\mathbf{SHyp}_\Sigma$. We already know that $\mathbf{LHyp}_\Sigma$ is a full subcategory of $\mathbf{SHyp}_\Sigma$ (Corollary 11) with inclusion functor I defined as in Definition 9.

Proposition 63. *$\mathbf{SHyp}_\Sigma$ is adhesive.*

Proof. $\mathbf{SHyp}_\Sigma$ is a slice category of a presheaf category, so it is adhesive [22]. $\qquad\square$

Proposition 64. *I preserves monomorphisms.*

Proof. Monomorphisms in $\mathbf{SHyp}_\Sigma$ and $\mathbf{LHyp}_\Sigma$ are homomorphisms with injective components (Lemmas 4 and 7). As f_T is injective iff f_S is injective (Lemma 8), we only need to consider f_S and f_E. I preserves E and identifies S with V, so for any morphism f, if f_E and f_S are injective in $\mathbf{LHyp}_\Sigma$, then they must be injective in $\mathbf{SHyp}_\Sigma$. $\qquad\square$

Corollary 65. *$LHyp_\Sigma$ is a partial adhesive category.*

Partial adhesive categories have pushout complements for a certain class of pushouts. We call these I-pushouts.

Definition 66 (I-spans and I-pushouts)**.** *For a partial adhesive category $\mathcal{P}$ with an inclusion functor $I : \mathcal{P} \to \mathcal{A}$, an I-span is a span in $\mathcal{P}$ such that it has a pushout and it is preserved by I. A monomorphism $L \to G$ is called an I-matching if $K \to L \to G$ has a pushout complement $K \to C \to G$.*

It turns out that the required condition for an I-pushout in our context is very similar to the one used by Kissinger for his string graphs.

Definition 67 (Boundary-coherence)**.** *For any linear hypergraphs F, G, H, a pair of hypergraph homomorphisms $m : F \to G$ and $n : F \to H$ is called boundary coherent if*

- *for any input $v \in T_F^\blacklozenge$, at least one of $m(v)$ and $n(v)$ is an input.*

- *for any output $v \in S_F^\blacklozenge$, at least one of $m(v)$ and $n(v)$ is an output.*

Proposition 68. *A span of monomorphisms $G \xleftarrow{m} F \xrightarrow{n} H$ in $LHyp_\Sigma$ has an I-pushout if and only if m and n are boundary coherent.*

Proof. For ($\Rightarrow$), the existence of an I-pushout can be shown by the showing that the pushout hypergraph $K \in SHyp_\Sigma$ is a linear hypergraph, i.e. each vertex is incident on the sources and targets of at most one edge each.

$$
\begin{array}{ccc}
IF & \xrightarrow{\ In\ } & IH \\
{\scriptstyle Im}\big\downarrow & & \big\downarrow{\scriptstyle q} \\
IG & \xrightarrow[\ p\ ]{} & K
\end{array}
$$

IF, IG and IH can obviously be represented as linear hypergraphs, so each vertex is in the sources and targets of at most one edge. Since the sources and targets of edges must be preserved by a hypergraph homomorphism, and no additional edges can be added, the sources and targets of all edges in K are determined by IG and IH. If m and n are boundary-coherent, then every input v in IF will be in the targets of some edge e in at most one of IG and IH. Therefore in K it will also only be in the targets of e and only e. The same follows for outputs and sources. For a vertex v in the targets of edge e in IF, it must be in the targets of $e_G = Im(e)$ and $e_H = In(e)$ by the homomorphism condition, and only these edges because IG and IH are linear hypergraphs. e_G and e_H will be mapped to the same edge e_K by the pushout, so v will only be in the targets of e_K. Again the same holds for vertices in the sources of edges. Therefore, the pushout hypergraph of any boundary-coherent span of monomorphisms in $LHyp_\Sigma$ is a linear hypergraph.

For ($\Leftarrow$), we must show that if the span is not boundary-coherent, then K is not a linear hypergraph. If there is an input v such that $Im(v)$ and $In(v)$ are not inputs, then it must map to two distinct edges $e_G \in E_G$ and $e_H \in E_H$. It cannot map to an edge previously in IF

as the targets of edges must be preserved by the homomorphism. $v_K = pIm(v) = qIn(v)$ therefore exists in the targets of two distinct edges $p(e_G)$ and $q(e_H)$, so it is not a linear hypergraph. Once more the same holds for outputs and sources. Therefore the pushout hypergraph of any non-boundary-coherent span of monomorphisms in $\mathbf{LHyp}_\Sigma$ is not a linear hypergraph. $\square$

In general, although we know that pushout complements are unique, we do not usually have a guarantee that they exist for some arbitrary span of monomorphisms, so cannot arbitrarily find matchings. However, in our case we actually can guarantee they exist!

Remark 69. *A common approach to use when finding the existence of a pushout complements for a monomorphism $p : L \to G$ is the* no-dangling-edges *condition, in which for any vertex $v \in V_L - V_K$, all edges incident on $m(v)$ must be under the image of m. However in our case this holds definitionally, as assigning non-interface vertices additional edges in V_G would result in a malformed* linear *hypergraph.*

Theorem 70 (I-matchings)**.** *For any rewrite rule $L \leftarrow K \to R$ and linear hypergraph G, any monomorphism $L \to G$ in $\mathbf{LHyp}_{\Sigma,\mathcal{C}}$ is an I-matching.*

Proof. Since $K \xrightarrow{b_1} L \xrightarrow{m} G$ are both monos, we can assume that $G \subseteq L \subseteq K$ with no loss of generality. To form a pushout complement C we take $T_C = T_G - (T_L - T_K)$, $S_C = S_G - (S_L - S_K)$, $E_C = T_G - T_L$ and for all $v_1 \in T_L^\blacklozenge$ and $v_2 \in S_L^\blacklozenge$ we set $\rho_C(\kappa(v_1)) = \lambda_C(\kappa^{-1}(v_2)) = \blacklozenge$. This is a well-formed linear hypergraph: vertices not in the image of m are unaffected; $v \in T_C^\blacklozenge$ which originally connected to L on the right are now connected to the interface, and similar for $S_C^\blacklozenge$ and left. Moreover, $L \cap C = K$ and $L \cup C = G$, so C is a pushout complement, thus $L \to G$ is an I-matching. $\square$

We can view identifying a match of L in G as applying operations to L until we reach G. These sequences can be expressed as monomorphisms in $\mathbf{LHyp}_\Sigma$, so they are all matchings. We can therefore perform a rewrite on linear hypergraphs for a rewrite rule $\langle L, R \rangle$ and a matching $L \to G$ by following the procedure described at the beginning of this section and constructing the double pushout diagram. An example of this can be seen in §7, Figure 5.

Theorem 71 (Rewriting system)**.** *For any rewriting system $\mathcal{E}$ in $\mathbf{Term}_\Sigma$,*

$$g \twoheadrightarrow_\mathcal{E} h \qquad \textit{iff} \qquad [\![g]\!] \rightsquigarrow_{[\![\mathcal{E}]\!]} [\![h]\!].$$

Proof. For ($\Rightarrow$), assume that $g \twoheadrightarrow_\mathcal{E} h$. By Definition 56 this means that there exists $\langle l, r \rangle \in \mathcal{R}$ such that

Using $[\![-]\!]$ we can translate the rewrite rule $\langle l, r \rangle$ into a span of hypergraphs $[\![l]\!] \leftarrow K \to [\![r]\!]$ containing only the input and output vertices of $[\![l]\!]$ and $[\![r]\!]$. K is a subgraph of $[\![l]\!]$ and $[\![r]\!]$ so there exist monomorphisms $K \to [\![l]\!]$ and $K \to [\![r]\!]$. Since l is a subterm of f, L corresponds to a subgraph of $[\![g]\!]$ by Lemma 60, so there exists a monomorphism $[\![l]\!] \to [\![g]\!]$. Any monomorphism $[\![l]\!] \to [\![g]\!]$ is an I-matching, so there exists a pushout complement

31

$K \to C \to \llbracket g \rrbracket$. We can use the same procedure for $\llbracket r \rrbracket$ and $\llbracket h \rrbracket$ and assert that the pushout complement $K \to C' \to \llbracket h \rrbracket$ exists. Since $C = \llbracket g \rrbracket - \llbracket l \rrbracket$ and $C' = \llbracket h \rrbracket - \llbracket r \rrbracket$, and $\llbracket g \rrbracket$ and $\llbracket h \rrbracket$ differ only by $\llbracket l \rrbracket$ and $\llbracket r \rrbracket$, $C = C'$. Finally, since $\llbracket g \rrbracket$, C and $\llbracket h \rrbracket$ all share the same interface, we can define J as the linear hypergraph with $\mathrm{dom}(g) + \mathrm{cod}(g)$ vertices that are the sources and targets of the interface, and morphisms $J \to \llbracket g \rrbracket$, $J \to C$ and $J \to \llbracket h \rrbracket$. This forms the complete DPO diagram, so $\llbracket g \rrbracket \rightsquigarrow_{\llbracket \mathcal{E} \rrbracket} \llbracket h \rrbracket$.

For ($\Leftarrow$), we assume that $\llbracket g \rrbracket \rightsquigarrow_{\llbracket \mathcal{E} \rrbracket} \llbracket h \rrbracket$ for some rewrite rule $\langle \llbracket l \rrbracket, \llbracket r \rrbracket \rangle$. This means there exists a DPO diagram with monomorphisms $\llbracket l \rrbracket \to \llbracket g \rrbracket$ and $\llbracket r \rrbracket \to \llbracket h \rrbracket$. Therefore by Lemma 60, l is a subterm of g and r is a subterm of h, so $g \to_{\mathcal{E}} h$.

$\square$

We can therefore perform a rewrite in $\mathbf{LHyp}_\Sigma$ given a rewrite rule $\langle L, R \rangle$ and matching $L \to G$ by following the procedure described at the beginning of this section and constructing the double pushout diagram. An example can be seen in §7, Figure 5.

Remark 72. *The results of this section gives us a graph rewriting system for rules that are spans of monomorphisms, i.e. those in which no vertices of the interface K are collapsed into one. For many rules this is perfectly acceptable, such as to model the transformation of an operation and its arguments into a result. However there are cases, such as a rule introducing an edge to an empty wire, where this situation might arise:*
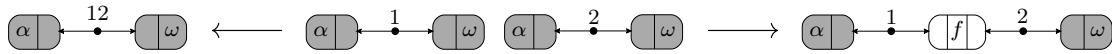
This is since the pushout complement is only uniquely defined for $K \xrightarrow{m} L \to G$ if m is mono. In [3] this phenomenon is permitted thanks to a compact closed structure in which wires can be directed arbitrarily. However in our traced context we have a strict notion of source and target which cannot be altered.

This is where our notion of homeomorphism (Section 3.6) comes into play. Much like when we represented trace as a monomorphism, we can perform an expansion on the left hand side of the rule, yielding us a span of monomorphisms as desired. For the case when n is not mono, we can redefine the rule to include an identity edge and then perform a smoothing to achieve the original.

7   Case study: Digital circuits

The initial motivation for developing this graph rewriting system was for use as an operational semantics for digital circuits. In this section we will detail how we can use our system to define a graph rewriting system for digital circuits.

When an STMC is Cartesian, it admits an *iterator* [13] to represent feedback. This leads immediately to models of digital circuits. For the monoidal tensor to be Cartesian product we require a natural transformation $\Delta_A : A \to A \otimes A$ (*diagonal*), and for the unit object to be terminal. The full list of axioms that hold in a Cartesian category can be seen in Table 1.

Naturality axioms

$f \cdot \Delta_B = \Delta_A \cdot f \otimes f$	$A \to B \otimes B$
$f \cdot \diamond_B = \diamond_A$	$A \to 0$

Commutative comonoid axioms

$\Delta_A \cdot \Delta_A \otimes A = \Delta_A \cdot (A \otimes \Delta_A)$	$A \to A \otimes A \otimes A$
$\Delta_A \cdot (\diamond_A \otimes A) = A$	$A \to A$
$\Delta_A \cdot (A \otimes \diamond_A)$	$A \to A$
$\Delta_A \cdot \times_{A,A} = \Delta_A$	$A \to A \otimes A$

Coherence axioms

|	
$\Delta_0 = 0$	$0 \to 0$
$\Delta_{A \otimes B} \cdot (A \otimes \times_{A,B} \otimes B) = \Delta_A \otimes \Delta_B$	$A \otimes B \to A \otimes B \otimes A \otimes B$
$\diamond_A = 0$	$0 \to 0$
$\diamond_{A \otimes B} = \diamond_A \otimes \diamond_B$	$A \otimes B \to 0$

Table 1: The axioms for a Cartesian monoidal category [24]

The addition of these axioms means that the category will have equations that are no longer captured by graph isomorphism, and rewrite rules are required. In the hypergraph interpretation the diagonal morphism (*copy map*) and the unique morphism into the terminal object (*delete map*) are represented as edges and the Cartesian axioms are expressed as graph rewrite rules.

Example 73. *We wish to interpret the Cartesian axioms in* ***Term***$_\Sigma$ *as graph rewrite rules in* ***LHyp***$_\Sigma$. *We start our example by interpreting naturality of Δ as a span of monomorphisms in* ***LHyp***$_\Sigma$.

$$\Rightarrow_{\Delta_1} \; : \; L \xleftarrow{m} C \xrightarrow{n} R$$
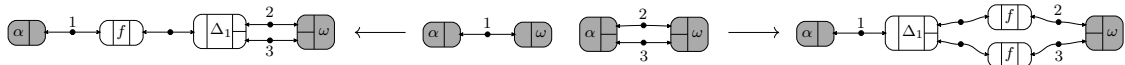
Terms such as $Tr^1(\curlyvee \otimes f \cdot \times_{1,1} \cdot \Delta_1 \otimes 1)$ (with $\curlyvee$ as defined in Example 2) illustrate why the graph rewriting approach is useful. The symmetry obscures the existence of the left hand side of $\Rightarrow_{\Delta_n}$ rule, so rewriting qua term requires the expenditure of additional computation to identify the redex. In contrast, in the hypergraph representation the rule is more easily identifiable as seen in Figure 5.

We present digital circuits as morphisms in a free PROP, where the objects correspond to the number of wires in a bus [11]. Morphisms are freely generated over a *circuit signature* containing values $v : 0 \to 1$, forming a lattice, *gates $k : m \to 1$*, and 'special morphisms' for forking ($\curlywedge : 1 \to 2$), joining ($\curlyvee : 2 \to 1$) and stubbing ($\sim \; : 1 \to 0$) wires. We
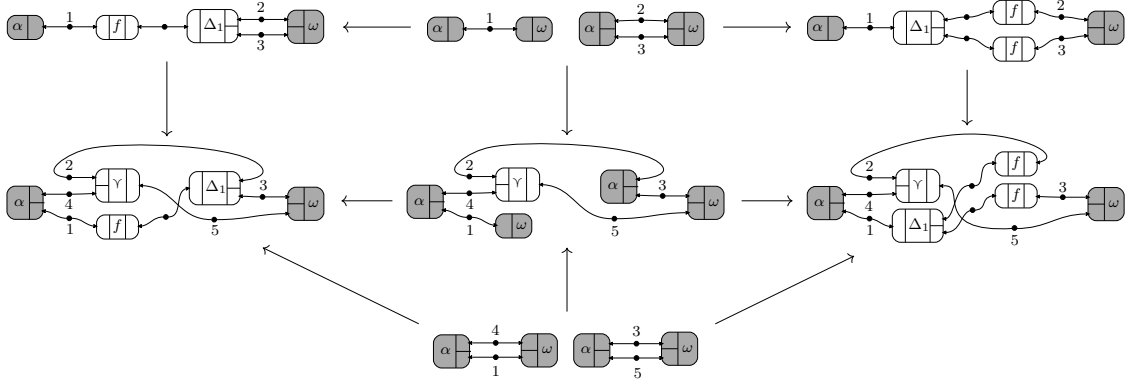
Figure 5: An example of a DPO diagram using the Cartesian copy axiom.

write $\mathbf{v} = v_0 \otimes \cdots \otimes v_m$. Gates and special morphisms have associated axioms:

$$v \cdot \curlywedge = v \otimes v \qquad\qquad 0 \to 2$$

$$v \otimes v' \cdot \curlyvee = v \sqcup v' \qquad\qquad 0 \to 1$$

$$v \cdot \sim\, = 0 \qquad\qquad 0 \to 0$$

$$\mathbf{v} \cdot k = v \text{ s.t. if } v_i \in \mathbf{v} \sqsupseteq v_i' \in \mathbf{v}' \text{ then } \mathbf{v} \cdot k \sqsupseteq \mathbf{v}' \cdot k \qquad\qquad 0 \to 1$$

Additionally, we can use the special morphisms to define Cartesian copy and delete maps, along with a dual notion of maps ∇_n to join buses of arbitrary size.

The outputs of circuits depend wholly on their inputs, so a circuit will always give the same output for some given inputs. It follows that any circuits with zero outputs are considered to be equal.

Delay is represented as a morphism $\delta_t : 1 \to 1$, parameterised over a *time monoid* $t \in \mathbf{T}$. The axioms of delay are:

$$\delta \cdot k = k \cdot \delta \qquad\qquad m \to 1$$

$$\delta^2 \otimes v \otimes v' \cdot \nabla_2 \cdot k = ((\delta^2 \cdot k) \otimes (v \otimes v' \cdot k)) \cdot \nabla_1 \qquad\qquad 2 \to 1$$

$$\perp \cdot \delta = \perp \qquad\qquad 0 \to 0$$

$$\delta \cdot \sim\, = \sim \qquad\qquad 0 \to 0$$

Of most interest is the second axiom (*Streaming*), an interaction between gates and time which corresponds to stream manipulation. Two copies of the gate k are used, one to handle the 'head' of the waveform and the other to handle the 'tail'.

Feedback is represented using the trace, thus the iterator could be used to 'unfold' the circuit:

$$\mathsf{iter}(n \otimes g \cdot f) = g \cdot \mathsf{iter}(f) \qquad\qquad m \to n$$

$$\mathsf{iter}(f) = \langle \mathsf{iter}(f), n \rangle \cdot f \qquad\qquad m \to n$$

$$\mathsf{iter}(\mathsf{iter}(f)) = \mathsf{iter}((\langle n, n \rangle \otimes m) \cdot f) \qquad\qquad m \to n$$

We have already seen in Example 73 that term rewriting can be computationally awkward. Another problem in the reduction of circuits is that some of the local traces can be unproductive due to infinite unfolding. The key result in [11] is that a circuit can be converted to graph form and then brought to a normal form in which the reduction can be made both efficient and guaranteed to be either productive of, if unproductive, the lack of productivity can be efficiently detected. This technique illustrates very well Selinger's observation: 'For practical purposes, reasoning in the graphical language is almost always easier than reasoning from the axioms. On the other hand, the graphical definition is not very useful when one has to check whether a given category is monoidal; in this case, checking finitely many axioms is easier.' [24].

8 Generalisation

So far, we have considered only terms in PROPs, where objects are natural numbers and tensor product is addition. This corresponds especially well with the notion of a linear hypergraph with m inputs and n outputs corresponding to a morphism $m \to n$. However, we may also wish to represent categories with arbitrary objects, as we do with string diagrams. It is not difficult to generalise our hypergraphs to correspond to terms in *any* STMC with set of objects $\mathcal{C}$ and morphisms freely generated over signature Σ. We fix this STMC here as $\mathbf{Term}_{\Sigma,\mathcal{C}}$.

Rather than just thinking of the type of linear hypergraphs as $m \to n$, we could instead decompose them into the form $1 \otimes 1 \otimes \cdots \otimes 1 \to 1 \otimes 1 \otimes \cdots \otimes 1$, since tensor product is addition. In fact, this corresponds even more closely to our graphical notation, as each of the 'wires' in the graph represents the natural number 1. However, when we write the types this way, we see that there is no reason to restrict ourselves to just numbers. The wires can instead correspond to arbitrary objects, as in string diagrams. We can communicate this idea in our hypergraphs by labelling vertices as well as edges. These labels correspond to the objects in our category.

We extend linear hypergraph signatures with this vertex label set $\mathcal{C}$ to create a *generalised linear hypergraph signature*. Since the inputs and outputs of boxes are tensors of objects rather than natural numbers, edge labels now have associated domains and codomains of elements of the free monoid under $\otimes$ on $\mathcal{C}$ rather than arities and coarities of natural numbers. We introduce the vertex labelling operations $\Lambda^T : T \to \mathcal{C}$ and $\Lambda^S : S \to \mathcal{C}$ and the following conditions:

- For all $e \in E$, $\Lambda^{T^\star}(\mathsf{t}(e)) = \mathsf{cod}(\Lambda(e))$

- For all $e \in E$, $\Lambda^{S^\star}(\mathsf{s}(e)) = \mathsf{dom}(\Lambda(e))$

- For all $v_t \in T$ and $v_s \in S$, $(\Lambda^T \circ \kappa)(v_t) = \Lambda^S(v_s)$

Example 74 (Generalised linear hypergraph). *Below is an example of a generalised hypergraph of type $A \otimes B \to C \otimes D$, for a signature*

$$(\Sigma, \mathcal{C}) = (\{f : A \to B, g : B \to A, h : B \otimes A \to C \otimes D\}, \{A, B, C, D\}).$$

The corresponding term is $f \otimes g \cdot h$.

35
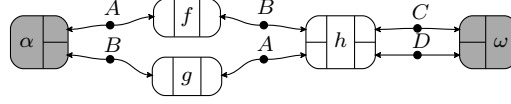
Generalised linear hypergraph homomorphisms are as with linear hypergraphs but with the addition of vertex label conditions.

$$
\begin{array}{ccc}
T_F \xrightarrow{\Lambda_F^T} \mathcal{C} & \quad & S_F \xrightarrow{\Lambda_F^S} \mathcal{C} \\
\Big\downarrow h_T \qquad \Big\downarrow \text{id} & \quad & \Big\downarrow h_S \qquad \Big\downarrow \text{id} \\
T_G \xrightarrow{\Lambda_G^T} \mathcal{C} & \quad & S_G \xrightarrow{\Lambda_G^S} \mathcal{C}
\end{array}
$$

As before, generalised linear hypergraphs over the signature $(\Sigma, \mathcal{C})$ are the objects in the category $\mathbf{LHyp}_{\Sigma,\mathcal{C}}$, with generalised linear hypergraph homomorphisms as the morphisms between them. Operations on generalised hypergraphs are defined similarly to before but with the addition of vertex labels, which are affected in the same way as edge labels. Therefore, we can also form a category where the generalised linear hypergraphs are morphisms:

Definition 75 (Generalised category of hypergraphs). *Generalised hypergraphs over the signature $(\Sigma, \mathcal{C})$ form a symmetric traced monoidal category $\mathbf{HypTerm}_{\Sigma,\mathcal{C}}$ with objects the elements of $\mathcal{C}^\star$, elements of $\mathbf{HypTerm}_{\Sigma,\mathcal{C}}(M, N)$ as the generalised hypergraphs of type $M \to N$, unit of composition as the identity hypergraph, parallel composition as $\otimes$ as the empty hypergraph, symmetry on M and N as $\times_{M,N}$, and trace on X as*

$$
Tr^X(-) : \mathbf{HypTerm}_{\Sigma,\mathcal{C}}(X \otimes M, X \otimes N) \to \mathbf{HypTerm}_{\Sigma,\mathcal{C}}(M, N).
$$

We define the operations

$$
[\![-]\!]_{\Sigma,\mathcal{C}} : \mathbf{Term}_{\Sigma,\mathcal{C}} \to \mathbf{HypTerm}_{\Sigma,\mathcal{C}}
$$
$$
\mathsf{term}_{\Sigma,\mathcal{C},\leq} : \mathbf{HypTerm}_{\Sigma,\mathcal{C}} \to \mathbf{Term}_{\Sigma,\mathcal{C}}
$$

in a similar manner to before. Soundness and completeness translate smoothly into the generalised case.

Theorem 76 (Generalised soundness). *For any morphisms $f, g \in \mathbf{Term}_{\Sigma,\mathcal{C}}$ if $f = g$ under the equational theory of the category then their interpretations as morphisms are isomorphic, $[\![f]\!]_{\Sigma,\mathcal{C}} \equiv [\![g]\!]_{\Sigma,\mathcal{C}}$.*

Theorem 77 (Generalised completeness). *For any hypergraph $F \in \mathbf{HypTerm}_{\Sigma,\mathcal{C}}$ there exists a unique morphism $f \in \mathbf{Term}_{\Sigma,\mathcal{C}}$, up to the equations of the STMC, such that $[\![f]\!]_{\Sigma,\mathcal{C}} = F$.*

9 Conclusion

9.1 Related work

Hypergraphs have been used before as a graphical language for compact closed categories, with interfaces defined via a cospan structure [3, 25, 5]. The cospan structure fits naturally

in this case as the hypergraphs come equipped with a Frobenius monoid which allows vertices to be identified arbitrarily. A similar recipe could have been followed here by requiring the morphisms of the cospan to be injective, which would have simplified some of the proofs. However, we considered a more elementary presentation in which one can arrive at the desired technical result without relying on avoidable mathematical concepts. This has its advantages in reaching a wider audience.

Symmetric traced monoidal categories, as studied here, can also be given a graphical language via their full and faithful embedding into compact closed categories [17]. However, this embedding does not guarantee definability and, moreover, cartesian product automatically becomes a bi-product in the compact closed setting [15], which would be unsound as a model for digital circuits.

The graphical intuitions behind the language of traced monoidal categories are strong, and they have been exploited before, for example in models of cyclic lambda calculi [13]. However, in this work graphs are not studied as combinatorial objects but rather are given as a quasi-informal diagrammatic language to aid understanding. Our formal perspective can allow us to revisit such work from a new perspective which opens the door to new algorithms and implementations.

9.2 Future work

In this paper we have revisited and solidified the mathematical foundation upon which the graphical language for digital circuits of [11] is based. One future avenue is that of *automating* the graph rewrites. While it may be simple to identify potential redexes by eye, it is not so simple for a computer. For example, there is the task of choosing between different rewrites. Furthermore, the inclusion of the trace adds complexity to proceedings. As we can unfold the trace infinitely many times the vanilla graph rewriting system is not terminating. To make our system suitable for automation we may need to add some extra structure, in a similar vein to [12]. There is also the question of confluence. Since we are using DPO with interfaces, confluence should be decidable [4].

Another goal is to refine the existing categorical semantics of digital circuits and identify any missing axioms. Indeed, for us to have a complete diagrammatic semantics the underlying categorical framework must of course also be complete! Instatiating the categorical semantics to a concrete category [12] pointed towards such additional axioms, most notably regarding *unproductive circuits*, circuits with no delay guarded feedback. By finding these axioms we can present our diagrammatic semantics as a complete package for reasoning about digital circuits.

Our firm mathematical, graphical foundation also opens up the opportunity for the development of circuit design tools, bringing the reduction-based operational semantics into a field dominated by simulation. Our approach serves to contrast, not replace, the existing paradigm, and offer an alternative insight into the design and execution of digital circuits.

References

[1] J. C. Baez and B. Fong. A compositional framework for passive linear networks, 2015. URL https://arxiv.org/abs/1504.05625.

[2] F. Bonchi, P. Sobocinski, and F. Zanasi. Full abstraction for signal flow graphs. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '15, page 515–526. Association for Computing Machinery, 2015. doi:10.1145/2676726.2676993.

[3] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobociński, and F. Zanasi. Rewriting modulo symmetric monoidal structure. In *2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10. IEEE, 2016. doi:10.1145/2933575.2935316.

[4] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobociński, and F. Zanasi. Confluence of graph rewriting with interfaces. In *European Symposium on Programming*, pages 141–169. Springer, 2017. doi:10.1007/978-3-662-54434-1_6.

[5] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobocinski, and F. Zanasi. Rewriting with Frobenius. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 165–174, 2018. doi:10.1145/3209108.3209137.

[6] S. Castellan and P. Clairambault. Causality vs. interleavings in concurrent game semantics. In *The 27th International Conference on Concurrency Theory (CONCUR 2016)*, volume 32, pages 1 – 3214, Québec City, Canada, Aug. 2016. doi:10.4230/LIPIcs.CONCUR.2016.32.

[7] B. Coecke and A. Kissinger. Picturing quantum processes. In *International Conference on Theory and Application of Diagrams*, pages 28–31. Springer, 2018. doi:10.1007/978-3-319-91376-6_6.

[8] B. Coecke, M. Sadrzadeh, and S. Clark. Mathematical foundations for a compositional distributional model of meaning, 2010. URL https://arxiv.org/abs/1003.4394.

[9] H. Ehrig, M. Pfender, and H. J. Schneider. Graph-grammars: An algebraic approach. In *14th Annual Symposium on Switching and Automata Theory (swat 1973)*, pages 167–180. IEEE, 1973. doi:10.1109/SWAT.1973.11.

[10] D. R. Ghica. A knot theory for eight year olds. *Mathematical Teaching*, (264-268), 2018.

[11] D. R. Ghica and A. Jung. Categorical semantics of digital circuits. In *Proceedings of the 16th Conference on Formal Methods in Computer-Aided Design*, pages 41–48. FMCAD Inc, 2016. doi:10.1109/FMCAD.2016.7886659.

[12] D. R. Ghica, A. Jung, and A. Lopez. Diagrammatic Semantics for Digital Circuits. In *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*, volume 82, pages 24:1–24:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.CSL.2017.24.

[13] M. Hasegawa. Recursion from cyclic sharing: traced monoidal categories and models of cyclic lambda calculi. In *International Conference on Typed Lambda Calculi and Applications*, pages 196–213. Springer, 1997. doi:10.1007/3-540-62688-3_37.

[14] M. Hasegawa. On traced monoidal closed categories. *Mathematical Structures in Computer Science*, 19(2):217–244, 2009. doi:10.1017/S0960129508007184.

[15] R. Houston. Finite products are biproducts in a compact closed category. *Journal of Pure and Applied Algebra*, 212(2):394–400, 2008. doi:10.1016/j.jpaa.2007.05.021.

[16] A. Joyal and R. Street. The geometry of tensor calculus, i. *Advances in mathematics*, 88 (1):55–112, 1991. doi:10.1016/0001-8708(91)90003-P.

[17] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 119, pages 447–468. Cambridge University Press, 1996. doi:10.1017/S0305004100074338.

[18] G. M. Kelly and M. L. Laplaza. Coherence for compact closed categories. *Journal of pure and applied algebra*, 19:193–213, 1980. doi:10.1016/0022-4049(80)90101-2.

[19] A. Kissinger. Pictures of processes: Automated graph rewriting for monoidal categories and applications to quantum computing, 2012. URL https://arxiv.org/abs/1203.0202.

[20] A. Kissinger and S. Uijlen. A categorical semantics for causal structure. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, 2017. doi:10.1109/LICS.2017.8005095.

[21] S. Lack. Composing PROPs. *Theory and Applications of Categories*, 13(9):147–163, 2004. URL http://www.tac.mta.ca/tac/volumes/13/9/13-09abs.html.

[22] S. Lack and P. Sobociński. Adhesive categories. In *International Conference on Foundations of Software Science and Computation Structures*, pages 273–288. Springer, 2004. doi:10.1007/978-3-540-24727-2_20.

[23] A. Pitts. Nominal techniques. *ACM SIGLOG News*, 3(1):57–72, 2016. URL https://www.cl.cam.ac.uk/~amp12/papers/nomt/nomt.pdf.

[24] P. Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, pages 289–355. Springer, 2010. doi:10.1007/978-3-642-12821-9_4.

[25] F. Zanasi. Rewriting in free hypergraph categories. *Electronic Proceedings in Theoretical Computer Science*, 263:16–30, Dec 2017. ISSN 2075-2180. doi:10.4204/eptcs.263.2.

Appendices

For a function $f : X \to Y$, we can *override* it to specify that values x that satisfy a predicate p are mapped to some other value $y \in Y$ with the syntax $f'(x) = f \oslash \{x \mapsto y \,|\, p\}$.

$$f'(x) = \begin{cases} y & \text{if } p(x) \text{ holds} \\ f(x) & \text{otherwise} \end{cases}$$

A Constructs and operations

Recall that a function must be *everywhere-defined*, that is to say for any function $f : X \to Y$, for all $x \in S$, there exists some $y \in Y$ such that $f(x) = y$. We call a linear hypergraph $F = ((T, \leq^T), (S, \leq^S), E, \lambda, \rho, \kappa, \Lambda)$ *well-formed* if λ, ρ, κ and Λ are everywhere-defined, κ is bijective and the labelling condition is satisfied.

A.1 Well-formedness of sequential composition (Proposition 16)

For any two hypergraphs $F : m \to n$ and $G : n \to p$:

$$F = ((T_F, \leq_F^T), (S_F, \leq_F^S), E_F, \lambda_F, \rho_F, \kappa_F, \Lambda_F)$$
$$G = ((T_G, \leq_G^T), (S_G, \leq_G^S), E_G, \lambda_G, \rho_G, \kappa_G, \Lambda_G)$$

their sequential composition $H : m \to p = F \cdot G$

$$H = ((T_H, \leq_H^T), (S_H, \leq_H^S), E_H, \lambda_H, \rho_H, \kappa_H, \Lambda_H)$$

is a well-formed hypergraph.

$\lambda_H : T_H \to E_H + 1$ **is everywhere-defined.** $\lambda_F : T_F \to E_F + 1$ and $\lambda_G : T_G \to E_G + 1$ are everywhere-defined by definition. For $v \in T_F^{\blacklozenge}$, $\lambda_H(v) = \blacklozenge \in E_H + 1$. For $v \in T_F - T_F^{\blacklozenge}$, $\lambda_H(v) = \lambda_F(v) \in E_F \in E_H + 1$. For $v \in T_G - T_G^{\blacklozenge}$, $\lambda_H(v) = \lambda_G(v) \in E_G \in E_H + 1$. Therefore λ_H is everywhere-defined.

$\rho_H : S_H \to E_H + 1$ **is everywhere-defined.** $\rho_F : S_F \to E_F + 1$ and $\rho_G : S \to E_G + 1$ are everywhere-defined by definition. For $v \in S_F - S_F^{\blacklozenge}$, $\rho_H(v) = \rho_F(v) \in E_F \in E_H + 1$. For $v \in S_G^{\blacklozenge}$, $\rho_H(v) = \blacklozenge \in E_H + 1$. For $v \in S_G - S_G^{\blacklozenge}$, $\rho_H(v) = \rho_G(v) \in E_G \in E_H + 1$. Therefore ρ_H is everywhere-defined.

$\kappa_H : T_H \to S_H$ **is everywhere-defined.** $\kappa_F : T_F \to S_F$ and $\kappa_G : T_G \to T_G$ are everywhere-defined by definition. For $v \in T_F$ where $\kappa_F(v) \notin S_F^{\blacklozenge}$, $\kappa_H(v) = \kappa_F(v) \in S_F - S_F^{\blacklozenge} \in S_H$. For $v \in T_F$ where $\kappa_F(v) = \pi_x(S_F^{\blacklozenge})$, $\kappa_H(v) = \kappa_H(\pi_x(T_G^{\blacklozenge})) \in S_G \in S_H$. For $v \in T_G - T_G^{\blacklozenge}$, $\kappa_H(v) = \kappa_G(v) \in S_G \in S$.

κ_H **is bijective.** $\kappa_F : T_F \to S_F$ and $\kappa_G : T_G \to S_G$ are bijective by definition. **Injectivity:** For $v \in T_F$ when $\kappa_F(v) \notin S_F^{\blacklozenge}$, $\kappa_H(v) = \kappa_F(v)$ which is injective. For $v \in T_F$ when $\kappa_F(v) = \pi_x(S_F^{\blacklozenge})$, $\kappa_H(v) = \kappa_H(\pi_x(T_G^{\blacklozenge}))$. $|S_F^{\blacklozenge}| = |T_G^{\blacklozenge}|$ by definition, so the mapping is injective. For $v \in T_G - T_G^{\blacklozenge}$, $\kappa_H(v) = \kappa_G(v)$, which is injective. **Surjectivity:** For $v \in S_F - S_F^{\blacklozenge}$, v is mapped to by $\kappa_H(v_1)$ where $v_1 \in T_F$. For $v \in \kappa_G^{\star}(T_G^{\blacklozenge})$, v is mapped to by $\kappa^{\star}(v_1)$ where $\kappa_F(v) = S_F^{\blacklozenge}$ For $v \in S_G$, v is mapped to by $\kappa_G^{\star}(v_1)$ where $v_1 \in T_G$. Therefore κ_H is bijective.

Λ_H is everywhere-defined. $\Lambda_F : E_F \to \Sigma$ and $\Lambda_G : E_G \to \Sigma$ are everywhere-defined by definition. For $e \in E_F$, $\Lambda_H(e) = \Lambda_F(e) \in \Sigma$. For $e \in E_G$, $\Lambda_H(e) = \Lambda_G(e) \in \Sigma$. Therefore Λ_H is everywhere-defined.

$\forall e_1, e_2 \in E_H$**, if** $\Lambda_H(e_1) = \Lambda_H(e_2)$ **then** $|\mathsf{s}_H(e_1)| = |\mathsf{s}_H(e_2)|$ **and** $|\mathsf{t}_H(e_1) = \mathsf{t}_H(e_2)|$**.** By definition, the condition holds for Λ_F and Λ_G. For $e \in E_F$, $\Lambda_H(e) = \Lambda_F(e)$ so the condition holds. For $e \in E_G$, $\Lambda_H(e) = \Lambda_G(e)$ so the condition holds.

A.2 Well-formedness of parallel composition (Proposition 19)

For any $F : m \to n$ and $G : p \to q$,

$$F = ((T_F, \leq_F^T), (S_F, \leq_F^S), E_F, \lambda_F, \rho_F, \kappa_F, \Lambda_F)$$
$$G = ((T_G, \leq_G^T), (S_G, \leq_G^S), E_G, \lambda_G, \rho_G, \kappa_G, \Lambda_G)$$

their parallel composition $H : m + p \to n + q$,

$$H = F \otimes G = ((T_H, \leq_H^T), (S_H, \leq_H^S), E_H, \lambda_H, \rho_H, \kappa_H, \Lambda_H)$$

is a well formed hypergraph.

$\lambda_H : T_H \to E_H + 1$ **is everywhere-defined.** $\lambda_F : T_H \to E_F + 1$ and $\lambda_G : T_H \to E_G + 1$ are everywhere-defined by definition. For $v \in T_F^{\blacklozenge} + T_G^{\blacklozenge}$, $\lambda_H(v) = \alpha \in E_H + 1$. For $v \in T_F - T_F^{\blacklozenge}$, $\lambda_H(v) = \lambda_F(v) \in E_F \in E_H + 1$. For $v \in T_G - T_G^{\blacklozenge}$, $\lambda_H(v) = \lambda_G(v) \in E_G \in E_H + 1$. Therefore λ_H is everywhere-defined.

$\rho_H : S_H \to E_H + 1$ **is everywhere-defined.** $\rho_F : S_F \to E_F + 1$ and $\rho_G : S_H \to E_G + 1$ are everywhere-defined by definition. For $v \in S_F^{\blacklozenge} + S_G^{\blacklozenge}$, $\rho_H(v) = \omega \in E_H + 1$. For $v \in S_F - S_F^{\blacklozenge}$, $\rho_H(v) = \rho_F(v) \in E_F \in E_H + 1$. For $v \in S_G - S_G^{\blacklozenge}$, $\rho_H(v) = \rho_G(v) \in E_G \in E_H + 1$. Therefore ρ_H is everywhere-defined.

$\kappa_H : T_H \to S_H$ **is everywhere-defined.** $\kappa_F : T_F \to S_F$ and $\kappa_G : T_G \to T_G$ are everywhere-defined by definition. For $v \in T_F$ $\kappa_H(v) = \kappa_F(v) \in S_F \in S_H$. For $v \in T_G$, $\kappa_H(v) = \kappa_G(v) \in S_G \in S_H$. Therefore κ_H is everywhere-defined.

κ_H **is bijective.** $\kappa_F : T_F \to S_F$ and $\kappa_G : T_G \to S_G$ are bijective by definition. For $v \in S_F$, $\kappa_H(v) = \kappa_F(v)$ which is bijective. For $v \in S_G$, $\kappa_H(v) = \kappa_G(v)$ which is bijective. Therefore κ_H is bijective.

Λ_H **is everywhere-defined.** $\Lambda_F : E_F \to \Sigma$ and $\Lambda_G : E_G \to \Sigma$ are everywhere-defined by definition. For $e \in E_F$, $\Lambda_H(e) = \Lambda_F(e) \in \Sigma$. For $e \in E_G$, $\Lambda_H(e) = \Lambda_G(e) \in \Sigma$. Therefore Λ_H is everywhere-defined.

$\forall e_1, e_2 \in E_H$**, if** $\Lambda_H(e_1) = \Lambda_H(e_2)$ **then** $|\mathsf{s}_H(e_1)| = |\mathsf{s}_H(e_2)|$ **and** $|\mathsf{t}_H(e_1)| = |\mathsf{t}_H(e_2)|$**.** By definition, this holds for Λ_F and Λ_G. For $e \in E_F$, $\Lambda_H(e) = \Lambda_F(e)$ so the condition holds. For $e \in E_G$, $\Lambda_H(e) = \Lambda_G(e)$ so the condition holds.

A.3 Bifunctoriality I (Proposition 21.I)

For any $m, n \in \mathbb{N}$, $m \otimes n \equiv m + n$.

Definitions

$A = m \otimes n$

$$T_A = [m] + [n] \qquad \leq^T_A \; = \; \leq^{[m]} \oslash \leq^{[n]} \qquad S_A = [m] + [n] \qquad \leq^S_A \; = \; \leq^{[m]} \oslash \leq^{[n]}$$
$$E_A = \emptyset \qquad \Lambda = \emptyset \qquad \lambda_A(v) = \blacklozenge \qquad \rho_A(v) = \blacklozenge$$
$$\kappa_A(\pi_x([m])) = \pi_x([m]) \qquad \kappa_A(\pi_x([n])) = \pi_x([n])$$

$B = m + n$

$$T_B = [m+n] \qquad \leq^T_B \; = \; \leq^{[m+n]} \qquad S_B = [m+n] \qquad \leq^S_B \; = \; \leq^{[m+n]}$$
$$E_A = \emptyset \qquad \Lambda = \emptyset \qquad \lambda_A(v) = \blacklozenge \qquad \rho_A(v) = \blacklozenge$$
$$\kappa_A(\pi_x([m+n])) = \pi_x([m+n])$$

Equivalence maps

$$f_T(v) = \begin{cases} \pi_x([m+n]) & \text{if } v = \pi_x([m]) \\ \pi_{x+m}([m+n]) & \text{if } v = \pi_x([n]) \end{cases}$$
$$f_S(v) = \begin{cases} \pi_x([m+n]) & \text{if } v = \pi_x([m]) \\ \pi_{x+m}([m+n]) & \text{if } v = \pi_x([n]) \end{cases}$$
$$f_E = \emptyset$$

A.4 Bifunctoriality II (Proposition 21.II)

For any $F : m \to n, G : r \to s, H : n \to p, K : s \to t$,

$$(F \otimes G) \cdot (H \otimes K) \equiv (F \cdot H) \otimes (G \cdot K)$$

Definitions

$A = F \otimes G \cdot H \otimes K$

$$T_A = T_F + T_G + T_H - T_H^\blacklozenge + T_K - T_K^\blacklozenge \qquad \leq^T_A \; = \; \leq^T_f \oslash \leq^T_g \oslash \leq^T_h \oslash \leq^T_k$$
$$S_A = S_F - S_F^\blacklozenge + S_G - S_G^\blacklozenge + S_H + S_K \qquad \leq^S_A \; = \; \leq^S_f \oslash \leq^S_g \oslash \leq^S_h \oslash \leq^S_k$$
$$E_A = E_F + E_G + E_H + E_K \qquad \Lambda_A = \Lambda_F \oplus \Lambda_G \oplus \Lambda_H \oplus \Lambda_K$$
$$\lambda_A(v) = \lambda_F \oplus \lambda_G \oplus \lambda_H \oplus \lambda_K \oslash \{v \mapsto \blacklozenge \,|\, v \in T_F^\blacklozenge + T_G^\blacklozenge\}$$
$$\rho_A(v) = \rho_F \oplus \rho_G \oplus \rho_H \oplus \rho_K \oslash \{v \mapsto \blacklozenge \,|\, v \in S_F^\blacklozenge + S_G^\blacklozenge\}$$
$$\phi = \{v \mapsto \kappa_H(\pi_x(S_H^\blacklozenge)) \,|\, \kappa_F(v) = \pi_x(S_F^\blacklozenge)\} \oslash \{v \mapsto \kappa_K(\pi_x(S_K^\blacklozenge)) \,|\, \kappa_G(v) = \pi_x(S_G^\blacklozenge)\}$$
$$\kappa_A(v) = \kappa_F \oplus \kappa_G \oplus \kappa_H \oplus \kappa_K \oslash \phi$$

The definition of $B = (F \cdot H) \otimes (G \cdot K)$ is identical to A. **Equivalence maps**

$$f_T = \text{id} \qquad f_S = \text{id} \qquad f_E = \text{id}$$

A.5 Naturality of swap (Proposition 25)

For any $F : m \to n$ and $G : p \to q$,

$$f \otimes g \cdot \times_{n,q} \equiv \times_{m,p} \cdot g \otimes f$$

We define our hypergraphs using the 'alternate' representation (Lemma 24).

$A = F \otimes G \cdot \times_{n,q}$

$$T_A = T_F + T_G \qquad \leq_A^T = S_F \oslash S_G$$
$$S_A = S_F - S_F^{\blacklozenge} + S_G - S_G^{\blacklozenge} + [q] + [n] \qquad \leq_A^S = S_F \oslash S_G \oslash [q] \oslash [n]$$
$$\lambda_A(v) = \lambda_F \oplus \lambda_G \oslash \{v \mapsto \blacklozenge \mid v \in T_F^{\blacklozenge} + T_G^{\blacklozenge}\} \qquad \rho_A(v) = \rho_F \oplus \rho_G \oslash \{v \mapsto \blacklozenge \mid v \in [q] + [n]\}$$
$$\kappa_A(v) = \kappa_F \oplus \kappa_F \oslash \{v \mapsto \pi_x([n]) \mid \kappa_F(v) = \pi_x(S_F^{\blacklozenge})\} \oslash \{v \mapsto \pi_x([q]) \mid \kappa_F(v) = \pi_x(S_G^{\blacklozenge})\}$$

$B = \times_{m,p} \cdot G \otimes F$

$$T_B = [m] + [p] + T_G - T_G^{\blacklozenge} + T_F - T_F^{\blacklozenge} \qquad \leq_B^T = [m] \oslash [p] \oslash T_G \oslash T_F$$
$$S_B = S_G + S_F \qquad \leq_B^S = S_G \oslash S_F$$
$$\lambda_B(v) = \lambda_G \oplus \lambda_F \oslash \{v \mapsto \blacklozenge \mid v \in [m] + [p]\} \qquad \rho_B(v) = \rho_G \oplus \rho_F \oslash \{v \mapsto \blacklozenge \mid v \in S_G^{\blacklozenge} + S_F^{\blacklozenge}\}$$
$$\kappa_B(v) = \kappa_F \oplus \kappa_F \oslash \{v \mapsto \kappa_F(\pi_x(T_F^{\blacklozenge})) \mid v = \pi_x([m]\} \oslash \{v \mapsto \kappa_G(\pi_x(S_G^{\blacklozenge}) \mid v = \pi_x([p]))\}$$

Equivalence maps

$$f_T(v) = \begin{cases} \pi_x([m]) & \text{if } v = \pi_x(T_F^{\blacklozenge}) \\ \pi_x([p]) & \text{if } v = \pi_x(S_G^{\blacklozenge}) \\ \text{id} & \text{otherwise} \end{cases} \qquad f_S(v) = \begin{cases} \pi_x(S_G^{\blacklozenge}) & \text{if } v = \pi_x([q]) \\ \pi_x(S_F^{\blacklozenge}) & \text{if } v = \pi_x([n]) \\ \text{id} & \text{otherwise} \end{cases}$$

$$f_E = \emptyset$$

f_T and f_S are bijections as f has m inputs and n outputs, and g has p inputs and q outputs, so there is a one-to-one correspondence between the sets.

A.6 Well-formedness of trace (Proposition 27)

For any $x \in \mathbb{N}$ and $F : x + m \to x + n$

$$F = ((T_F, \leq_F^T), (S_F, \leq_F^S), E_F, \lambda_F, \rho_F, \kappa_F, \Lambda_F)$$

its trace of x wires $\text{Tr}^x(f) : m \to n$

$$H = \text{Tr}^x(F) = ((T_H, \leq_H^T), (S_H, \leq_H^S), E_H, \lambda_H, \rho_H, \kappa_H, \Lambda_H)$$

is a well-formed linear hypergraph. For $x = 0$, $\text{Tr}^0(F) = F$, which is well-formed. For $x = 1$, we check the conditions in turn:

$\lambda_H : T_H \to E_H + 1$ **is everywhere-defined.** $\lambda_F : T \to E_F+1$ is everywhere-defined by definition. For $v \in T_F^\blacklozenge$, $\lambda_H(v) = \alpha$. Otherwise, $\lambda_H(v) = \lambda_F(v) \in E_F \in E$. Therefore λ_H is everywhere-defined.

$\rho_H : S_H \to E_H + 1$ **is everywhere-defined.** $\rho_F : S_F \to E_F + 1$ is everywhere-defined by definition. For $v \in S_F^\blacklozenge$, $\rho_H(v) = \omega \in E_H + 1$. Otherwise, $\rho_H(Hv) = \rho_F(v) \in E_F \in E_H + 1$. Therefore ρ_H is everywhere-defined.

$\kappa_H : T_H \to S_H$ **is everywhere-defined.** $\kappa_F : T_F \to S_F$ is everywhere-defined by definition. For v where $\kappa_F(v) = \pi_0(S_F^\blacklozenge)$, $\kappa_H(v) = \kappa_F(\pi_0(T_F^\blacklozenge)) \in S_F$. $\kappa_F(\pi_0(S_F^\blacklozenge)) \neq \pi_0(S_F^\blacklozenge)$, since this implies that $v = \pi_0(T_F^\blacklozenge)$, a deleted vertex. Therefore, $\kappa_F(\pi_0(S_F^\blacklozenge)) \in S$. Otherwise, $\kappa_H(v) = \kappa_F(v) \in S_F \in S_H$. Therefore, κ_H is everywhere-defined.

κ_H **is bijective** $\kappa_F : T_F \to S_F$ is bijective by definition. **Injectivity:** For v where $\kappa_F(v) = \pi_0(S_F^\blacklozenge)$, $\kappa_H(v) = \kappa_F(\pi_0(S_F^\blacklozenge))$. Otherwise, $\kappa_H(v) = \kappa_F(v) \neq \kappa_F(\pi_0(S_F^\blacklozenge))$ since $\kappa_F(\pi_0(S_F^\blacklozenge)) \notin T$. Since κ_F is injective, κ_H is injective. **Surjectivity:** For $v = \kappa_F(\pi_0(S_F^\blacklozenge))$, $\kappa_H(v_1) = v$, where v_1 satisfies $\kappa_H(v_1) = \pi_0(S_F^\blacklozenge)$. Otherwise, $\kappa_H(v_1) = v$, where v_1 satisfies $\kappa_F(v_1) = v$. Therefore κ_H is bijective.

Λ_H **is everywhere-defined:** $\forall e \in E_H, \exists l \in \Sigma$ **s.t.** $\Lambda_H(e) = l$ $\Lambda_F : E_F \to \Sigma$ is everywhere-defined by definition. $\Lambda_H = \Lambda_F$, therefore Λ_H is everywhere-defined.

$\forall e_1, e_2 \in E_H$, **if** $\Lambda_H(e_1) = \Lambda_H(e_2)$ **then** $|\mathsf{s}_H(e_1)| = |\mathsf{s}_H(e_2)|$ **and** $|\mathsf{t}_H(e_1) = \mathsf{t}_H(e_2)|$ By definition, the condition holds for Λ_F. $E_H = E_F$, $\Lambda_H = \Lambda_F$, therefore the condition holds.

For $x = k + 1$, $\mathrm{Tr}^{k+1}(F) = \mathrm{Tr}^1(\mathrm{Tr}^k(F))$. By our base case, $\mathrm{Tr}^1(F)$ is well-formed for any f, and by inductive hypothesis $\mathrm{Tr}^k(F)$ is also well-formed. Therefore $\mathrm{Tr}^{k+1}(F)$ is also well-formed for any $k \in \mathbb{N}$.

B Categorical axioms

B.1 Left identity

For any linear hypergraph $F : m \to n$, $m \cdot F \equiv F$.

Definitions

$A = m \cdot F$

$$T_A = [m] + T_F - T_F^\blacklozenge \qquad \leq_A^T = \leq^m \oslash \leq_f^T \qquad S_A = S_F \qquad \leq_A^S = \leq_f^S$$

$$E_A = E_F \qquad \Lambda_A = \Lambda_F$$

$$\lambda_A(v) = \lambda_F(v) \oslash \{v \mapsto \blacklozenge \mid v \in [m]\} \qquad \rho_A(v) = \rho_F(v)$$

$$\kappa_A(v) = \kappa_F \oslash \{v \mapsto \kappa_F(\pi_x(T_F^\blacklozenge)) \mid v = \pi_x([m])\}$$

Equivalence maps

$$f_T(v) = \begin{cases} \pi_x(T_F^\blacklozenge), & \text{if } v = \pi_x([m]) \\ \mathsf{id}, & \text{if } v \in T_F - T_F^\blacklozenge \end{cases} \qquad f_S = \mathsf{id} \qquad f_E = \mathsf{id}$$

f_T is bijective since there are n elements in $S_F^\blacklozenge$ by definition.

B.2 Right identity

For any linear hypergraph $F : m \to n$, $F \cdot n \equiv F$.

Definitions

$A = F \cdot n$

$$T_A = T_F \qquad \leq_A^T = \leq_F^T \qquad S_A = S_F - S_F^\blacklozenge + [n] \qquad \leq_A^S = \leq_F^S \oslash \leq^n$$

$$E_A = E_F \qquad \Lambda_A = \Lambda_F$$

$$\lambda_A = \lambda_F \qquad \rho_A(v) = \rho_F(v) \oslash \{v \in [n] \mapsto \blacklozenge\}$$

$$\kappa_A(v) = \kappa_F \oslash \{v \mapsto \pi_x([n]) \mid \kappa_F(v) = \pi_x(S_F^\blacklozenge)\}$$

Equivalence maps

$$f_T = \mathsf{id} \qquad f_S(v) = \begin{cases} \mathsf{id} & \text{if } v \in T_F - S_F^\blacklozenge \\ \pi_x(S_F^\blacklozenge) & \text{if } v = \pi_x([n]) \end{cases} \qquad f_E = \mathsf{id}$$

f_S is bijective since there are n elements in $S_F^\blacklozenge$ by definition.

B.3 Associativity

For any three linear hypergraphs $F : m \to n$, $G : n \to p$, $H : p \to q$,

$$F \cdot (G \cdot H) \equiv (F \cdot G) \cdot H.$$

Definitions

$a = (f \cdot g) \cdot h$

$$T_A = T_F + T_G - T_F^\blacklozenge + T_H - T_H^\blacklozenge \qquad \leq_A^T = \leq_f^T \oslash \leq_g^T \oslash \leq_h^T$$

$$S_A = S_F - S_F^\blacklozenge + S_G - S_G^\blacklozenge + S_H \qquad \leq_A^S = \leq_f^S \oslash \leq_g^S \oslash \leq_h^S$$

$$E_A = E_F + E_G + E_H \qquad \Lambda_A = \Lambda_F \oplus \Lambda_G \oplus \Lambda_H$$

$$\lambda_A(v) = (\lambda_F \oplus \lambda_G \oplus \lambda_H) \oslash \{v \in T_F^\blacklozenge \mapsto \alpha\}$$

$$\rho_A(v) = (\rho_F \oplus \rho_G \oplus \rho_H) \oslash \{v \in S_H^\blacklozenge \mapsto \omega\}$$

$$\kappa_A(v) = (\kappa_F \oplus \kappa_G \oplus \kappa_H) \oslash \phi$$

where

$$\phi(v) = \begin{cases} \kappa_H(\pi_y(T_H^\blacklozenge)) & \text{if } \kappa_F(v) = \pi_x(S_F^\blacklozenge) \wedge \kappa_G(\pi_x(S_G^\blacklozenge)) = \pi_y(S_G^\blacklozenge) \\ \kappa_H(\pi_x(T_H^\blacklozenge)) & \text{if } \kappa_G(v) = \pi_x(S_G^\blacklozenge) \\ \kappa_G(\pi_x(T_G^\blacklozenge)) & \text{if } \kappa_F(v) = \pi_x(S_F^\blacklozenge) \end{cases}$$

The definition of $F \cdot (G \cdot H)$ is identical.

Equivalence maps

$$f_S = \mathsf{id} \qquad f_T = \mathsf{id} \qquad f_E = \mathsf{id}$$

B.4 Left identity of tensor

For any linear hypergraph $F : m \to n, 0 \otimes f \equiv f$.

Definitions
$A = 0 \otimes F$

$$T_A = T_F \qquad \leq_A^T = \leq_F^T \qquad S_A = S_F \qquad \leq_A^S = \leq_F^S$$
$$E_A = E_F \qquad \Lambda_A = \Lambda_F$$
$$\lambda_A(v) = \lambda_F \oslash \{v \in T_F^{\blacklozenge} \mapsto \alpha\} \qquad \rho_A(v) = \rho_F \oslash \{v \in S_F^{\blacklozenge} \mapsto \omega\}$$
$$\kappa_A = \kappa_F$$

Equivalence maps
$$f_T = \mathsf{id} \qquad f_S = \mathsf{id} \qquad f_E = \mathsf{id}$$

B.5 Right identity of tensor

For any linear hypergraph $F : m \to n, f \otimes 0 \equiv f$.
The definition is identical to that above.

Equivalence maps

$$f_T = \mathsf{id} \qquad f_S = \mathsf{id} \qquad f_E = \mathsf{id}$$

B.6 Associativity of tensor

For any three linear hypergraphs $F : m \to n$, $G : p \to q$ and $H : r \to s$, $F \otimes (G \otimes H) \equiv (F \otimes G) \otimes H$.

Definitions $a = (f \otimes g) \otimes h$

$$T_A = T_F + T_G + T_H \qquad \leq_A^T = \leq_f^T \oslash \leq_g^T \oslash \leq_h^T$$
$$S_A = S_F + S_G + S_H \qquad \leq_A^S = \leq_f^S \oslash \leq_g^S \oslash \leq_h^S$$
$$E_A = E_F + E_G + E_H$$
$$\lambda_A(v) = (\lambda_F \oplus \lambda_G \oplus \lambda_H) \oslash \{v \in T_F^{\blacklozenge} + T_G^{\blacklozenge} + T_H^{\blacklozenge} \mapsto \alpha\}$$
$$\rho_A(v) = (\rho_F \oplus \rho_G \oplus \rho_H) \oslash \{v \in S_F^{\blacklozenge} + S_G^{\blacklozenge} + S_H^{\blacklozenge} \mapsto \omega\}$$
$$\kappa_A(v) = \kappa_F \oplus \kappa_G \oplus \kappa_H$$

The definition for $f \otimes (g \otimes h)$ is identical.

Equivalence maps
$$f_T = \mathsf{id} \qquad f_S = \mathsf{id} \qquad f_E = \mathsf{id}$$

B.7 Self-invertability

For any $m, n \in \mathbb{N}$, $\times_{m,n} \cdot \times_{n,m} \equiv m + n$.

Definitions

$a = \times_{m,n} \cdot \times_{n,m}$

$$T_A = [m] + [n] \qquad \leq_A^T = \leq^{[m]} \oslash \leq^{[n]}$$
$$S_A = [m] + [n] \qquad \leq_A^S = \leq^{[m]} \oslash \leq^{[n]}$$
$$E_A = \emptyset \qquad L_A = \emptyset \qquad \Lambda_A = \emptyset$$
$$\lambda_A = \alpha \qquad \rho_A = \omega \qquad \kappa_A(\pi_x([m])) = \pi_x([m]) \qquad \kappa_A(\pi_x([n])) = \pi_x([n])$$

Equivalence maps

$$f_T = \text{id} \qquad f_S = \text{id} \qquad f_E = \text{id}$$

B.8 Hexagon axioms

For any $m, n, p \in \mathbb{N}$, $\times_{m,n} \otimes p \cdot n \otimes \times_{m,p} \equiv \times_{m,n+p}$

Definitions $a = \times_{m,n} \otimes p \cdot n \otimes \times_{m,p}$

$$T_A = [m] + [n] + [p] \qquad \leq_A^T = \leq^{[m]} \oslash \leq^{[n]} \oslash \leq^{[p]}$$
$$S_A = [n] + [p] + [m] \qquad \leq_A^S = \leq^n \oslash \leq^{[p]} \oslash \leq^{[m]}$$
$$\lambda_A = \alpha \qquad \rho_A = \omega \qquad E_A = \emptyset \qquad \Lambda_A = \emptyset$$

$$\kappa_A(v) = \begin{cases} \pi_x([m]), & \text{if } v = \pi_x([m]) \\ \pi_x([n]), & \text{if } v = \pi_x([n]) \\ \pi_x([p]), & \text{if } v = \pi_x([p]) \end{cases}$$

$b = \times_{m,n+p}$

$$T_B = [m] + [n + p] \qquad \leq_B^T = \leq^{[m]} \oslash \leq^{[n+p]}$$
$$S_B = [n + p] + [m] \qquad \leq_B^S = \leq^{[n+p]} \oslash \leq^{[m]}$$
$$\lambda_B = \alpha \qquad \rho_B = \omega \qquad E_B = \emptyset \qquad \Lambda_A = \emptyset$$

$$\kappa_B(v) = \begin{cases} \pi_x([m]) & \text{if } v = \pi_x([m]) \\ \pi_x([n + p]), & \text{if } v = \pi_x([n + p]) \end{cases}$$

Equivalence maps

$$f_T = \begin{cases} \pi_x([m]), & \text{if } v = \pi_x([m]) \\ \pi_x([(n + p)]) & \text{if } v = \pi_x([n]) \\ \pi_{x+n}([(n + p)]), & \text{if } v = \pi_x([p]) \end{cases} \qquad f_S = \begin{cases} \pi_x([(n + p)]), & \text{if } v = \pi_x([n]) \\ \pi_{x+n}([(n + p)]), & \text{if } v = \pi_x([p]) \\ \pi_x([m]), & \text{if } v = \pi_x([m]) \end{cases}$$

$$f_E = \emptyset$$

Since $\times_{m,n}$ is symmetric, $\times_{m,n} = \times_{n,m}^{-1}$ and the second hexagon axiom is identical to the first.

B.9 Tightening

For any linear hypergraphs $F : x + m \to x + n$, $G : p \to m$, $H : n \to q$,

$$\text{Tr}^x(x \otimes G \cdot F \cdot x \otimes H) \equiv G \cdot \text{Tr}^x(F) \cdot H.$$

This is proved by induction on x.

Zero case: $x = 0$

$$\text{Tr}^0(0 \otimes G \cdot F \cdot 0 \otimes H) \equiv G \cdot \text{Tr}^0(F) \cdot H$$

$$
\begin{aligned}
\text{Tr}^0(0 \otimes G \cdot F \cdot 0 \otimes H) &= 0 \otimes G \cdot F \cdot 0 \otimes H && \text{definition of trace} \\
&= G \cdot F \cdot 0 \otimes H && \text{left identity of tensor} \\
&= G \cdot F \cdot H && \text{left identity of tensor} \\
&= G \cdot \text{Tr}^0(F) \cdot H && \text{definition of trace}
\end{aligned}
$$

Base case: $x = 1$

$$\text{Tr}^1(1 \otimes G \cdot F \cdot 1 \otimes H) \equiv G \cdot \text{Tr}^1(F) \cdot H$$

$a = \text{Tr}^1(1 \otimes G \cdot F \cdot 1 \otimes H)$

$$
\begin{aligned}
T_A &= T_G + T_F - T_F^{\blacklozenge} + T_H - T_H^{\blacklozenge} & \leq_A^T &= \leq_g^T \oslash \leq_f^T \oslash \leq_h^T \\
S_A &= S_G - S_G^{\blacklozenge} + S_F - S_F^{\blacklozenge} + S_H & \leq_A^S &= \leq_g^S \oslash \leq_f^S \oslash \leq_h^S \\
E_A &= E_G + E_F + E_H & \Lambda_A &= \Lambda_G \oplus \Lambda_F \oplus \Lambda_H \\
\lambda_A(v) &= \lambda_G \oplus \lambda_F \oplus \rho_H & \rho_A(v) &= \rho_G \oplus \rho_F \oplus \rho_H \\
\kappa_A(v) &= (\kappa_G \oplus \kappa_F \oplus \kappa_H) \oslash \phi
\end{aligned}
$$

where

$$
\phi(v) = \begin{cases}
\kappa_H(\pi_{z-1}(T_H^{\blacklozenge})) & \text{if } \kappa_G(v) = \pi_y(S_G^{\blacklozenge}) \\
& \quad \wedge \kappa_F(\pi_{y+1}(T_F^{\blacklozenge})) = \pi_0(S_F^{\blacklozenge}) \\
& \quad \wedge \kappa_F(\pi_0(S_F^{\blacklozenge})) = \pi_z(S_F^{\blacklozenge}), z \geq 1 \\
\kappa_F(\pi_0(S_F^{\blacklozenge})) & \text{if } \kappa_G(v) = \pi_y(S_G^{\blacklozenge}) \\
& \quad \wedge \kappa_F(\pi_{y+1}(T_F^{\blacklozenge})) = \pi_0(S_F^{\blacklozenge}) \\
\kappa_H(\pi_{y-1}(T_H^{\blacklozenge})) & \text{if } \kappa_F(v) = \pi_0(S_F^{\blacklozenge}) \\
& \quad \wedge \kappa_F(\pi_0(S_F^{\blacklozenge})) = \pi_y(S_F^{\blacklozenge}), y \geq 1 \\
\kappa_F(\pi_0(S_F^{\blacklozenge})) & \text{if } \kappa_F(v) = \pi_0(S_F^{\blacklozenge}) \\
\kappa_H(\pi_{z-1}(T_H^{\blacklozenge})) & \text{if } \kappa_G(v) = \pi_y(S_G^{\blacklozenge}) \\
& \quad \wedge \kappa_F(\pi_{y+1}(T_F^{\blacklozenge})) = \pi_z(S_F^{\blacklozenge}), z \geq 1 \\
\kappa_H(\pi_{y-1}(T_H^{\blacklozenge})) & \text{if } \kappa_F(v) = \pi_y(S_F^{\blacklozenge}), y \geq 1 \\
\kappa_F(\pi_{y+1}(T_F^{\blacklozenge})) & \text{if } \kappa_G(v) = \pi_y(S_G^{\blacklozenge})
\end{cases}
$$

$B = G \cdot \text{Tr}^1(F) \cdot H$

$$
\begin{aligned}
T_B &= T_G + T_F - T_F^{\blacklozenge} + T_H - T_H^{\blacklozenge} & \leq_B^T &= \leq_G^T \oslash \leq_F^T \cdot \leq_H^T \\
S_B &= S_G - S_G^{\blacklozenge} + S_F - S_F^{\blacklozenge} + S_H & \leq_B^S &= \leq_G^S \oslash \leq_F^S \cdot \leq_H^S \\
\lambda_B &= \lambda_G \oplus \lambda_F \oplus \lambda_H & \rho_B &= \rho_G \oplus \rho_F \oplus \rho_H \\
\kappa_B &= (\kappa_G \oplus \kappa_F \oplus \kappa_H) \oslash \phi
\end{aligned}
$$

where

$$
\phi(v) = \begin{cases}
\kappa_H(\pi_z(T_H^\blacklozenge)) & \text{if } \kappa_G(v) = \pi_y(S_G^\blacklozenge) \\
& \quad \wedge \kappa_F(\pi_{y+1}(T_F^\blacklozenge)) = \pi_0(S_F^\blacklozenge) \\
& \quad \wedge \kappa_F(\pi_0(S_F^\blacklozenge)) = \pi_{z+1}(S_F^\blacklozenge) \\
\kappa_H(\pi_z(T_H^\blacklozenge)) & \text{if } \kappa_G(v) = \pi_y(S_G^\blacklozenge) \\
& \quad \wedge \kappa_F(\pi_{y+1}(T_F^\blacklozenge)) = \pi_{z+1}S_F^\blacklozenge) \\
\kappa_H(\pi_y(S_H^\blacklozenge)) & \text{if if } \kappa_F(v) = \pi_0(S_F^\blacklozenge) \\
& \quad \wedge \kappa_F(\pi_0(S_F^\blacklozenge)) = \pi_{y+1}(S_F^\blacklozenge) \\
\kappa_H(\pi_y(S_H^\blacklozenge)) & \text{if } \kappa_F(v) = \pi_{y+1}(S_F^\blacklozenge) \\
\kappa_F(\pi_0(S_F^\blacklozenge)) & \text{if } \kappa_G(v) = \pi_y(S_G^\blacklozenge) \\
& \quad \wedge \kappa_F(\pi_{y+1}(T_F^\blacklozenge)) = \pi_0(S_F^\blacklozenge) \\
\kappa_F(\pi_{y+1}(T_F^\blacklozenge)) & \text{if } \kappa_G(v) = \pi_y(S_G^\blacklozenge) \\
\kappa_F(\pi_0(S_F^\blacklozenge)), & \text{if } \kappa_F(v) = \pi_0(S_F^\blacklozenge)
\end{cases}
$$

Equivalence maps

$$
f_T = \mathsf{id} \qquad f_S = \mathsf{id} \qquad f_E = \mathsf{id}
$$

Inductive case: $x = k + 1$

$$
\mathrm{Tr}^{k+1}(k + 1 \otimes G \cdot F \cdot k + 1 \otimes H) \equiv G \cdot \mathrm{Tr}^{k+1}(f) \cdot H
$$

$$
\begin{aligned}
\mathrm{Tr}^{k+1}(k + 1 \otimes G \cdot F \cdot k + 1 \otimes H) &= \mathrm{Tr}^1(\mathrm{Tr}^k(k + 1 \otimes G \cdot F \cdot k + 1 \otimes H)) && \text{definition of trace} \\
&= \mathrm{Tr}^1(\mathrm{Tr}^k(k \otimes 1 \otimes G \cdot F \cdot k \otimes 1 \otimes H)) && \text{bif of tensor I} \\
&= \mathrm{Tr}^1(1 \otimes G \cdot \mathrm{Tr}^k(F) \cdot 1 \otimes H) && \text{IH (1)} \\
&= G \cdot \mathrm{Tr}^1(\mathrm{Tr}^k(f)) \cdot H && \text{base case (2)} \\
&= G \cdot \mathrm{Tr}^{k+1}(f) \cdot H && \text{definition of trace}
\end{aligned}
$$

At (1), we use our inductive hypothesis with $m = 1 + m$ and $n = 1 + n$, since it applies for any $m, n \in \mathbb{N}$. At (2), we can use our base case since $\mathrm{Tr}^k(f)$ has type $1 + m \to 1 + n$.

B.10 Superposing

For any two hypergraphs $F : x + m \to n$ and $G : p \to q$, $\mathrm{Tr}^x(f \otimes g) \equiv \mathrm{Tr}^x(f) \otimes g$

Zero case $x = 0$
This follows immediately by definition of trace.

Base case $x = 1$ $A = \mathrm{Tr}^1(F \otimes G)$

$$T_A = T_F - \pi_0(S_F^{\blacklozenge}) + T_G \qquad \leq_A^T = \leq_F^T \oslash \leq_G^T$$
$$S_A = S_F - \pi_0(S_F^{\blacklozenge}) + S_G \qquad \leq_A^S = \leq_F^S \oslash \leq_G^S$$
$$E_A = E_F + E_G \qquad \Lambda_A = \Lambda_F \oplus \Lambda_G$$
$$\lambda_A = \lambda_F \oplus \lambda_G \oslash \{v \mapsto \blacklozenge \mid v \in T_F^{\blacklozenge} + T_G^{\blacklozenge}\}$$
$$\rho_A = \rho_F \oplus \rho_G \oslash \{v \mapsto \blacklozenge \mid v \in S_F^{\blacklozenge} + S_G^{\blacklozenge}\}$$
$$\kappa_A = \kappa_F \oplus \kappa_G \oslash \{v \mapsto \kappa_F(\pi_0(T_F^{\blacklozenge})) \mid \kappa_F(v) = \pi_0(S_F^{\blacklozenge})\}$$

The definition of $B = \mathrm{Tr}^1(F) \otimes G$ is identical.

Equivalence maps

$$f_T = \mathrm{id} \qquad f_S = \mathrm{id} \qquad f_E = \mathrm{id}$$

Inductive case: $x = k + 1$

$$\mathrm{Tr}^{k+1}(F \otimes G) = \mathrm{Tr}^{k+1}(F)$$

$$
\begin{aligned}
\mathrm{Tr}^{k+1}(F \otimes G) &= \mathrm{Tr}^1(\mathrm{Tr}^k(F \otimes G)) & \text{definition of trace}\\
&= \mathrm{Tr}^k(\mathrm{Tr}^1(F \otimes G)) & \text{comm of trace}\\
&= \mathrm{Tr}^k(\mathrm{Tr}^1(F) \otimes G) & \text{base case (1)}\\
&= \mathrm{Tr}^k(\mathrm{Tr}^1(F)) \otimes G & \text{IH (2)}\\
&= \mathrm{Tr}^1(\mathrm{Tr}^k(F)) \otimes G & \text{comm of trace}\\
&= \mathrm{Tr}^{k+1}(F) \otimes G & \text{definition of trace}
\end{aligned}
$$

At (1), we observe that as $F : k + 1 + m \to k + 1 + n$, we can use our base case with $m' = k + m$. Therefore $\mathrm{Tr}^1(F \otimes G) = \mathrm{Tr}^1(F) \otimes G$. At (2), $\mathrm{Tr}^1(F) : k + m \to k + n$ so we can apply our inductive hypothesis. Therefore for any $x \in \mathbb{N}$, $\mathrm{Tr}^x(F \otimes G) = \mathrm{Tr}^x(F) \otimes G$.

B.11 Yanking

For any $x \in \mathbb{N}$, $\mathrm{Tr}^x(\times_{x,x}) \equiv x$. This proof is by induction on x.

Zero case: $x = 0$

$$\mathrm{Tr}^0(\times_{0,0}) = 0$$

This follows immediately by definition of trace and symmetry.

Base case: $x = 1$

$$\mathrm{Tr}^1(\times_{1,1}) \equiv 1$$

$A = \mathrm{Tr}^1(\times_{1,1})$

$$T_A = [1] \qquad \leq_A^T = \leq^{[1]} \qquad S_A = [1] \qquad \leq_A^S = \leq^{[1]}$$
$$E_A = \emptyset \qquad \Lambda_A = \emptyset \qquad \lambda_A = \blacklozenge \qquad \rho_A = \blacklozenge \qquad \kappa_A(\pi_0(T_A)) = \pi_0(S_A)$$

Equivalence maps

$$f_T \equiv \text{id} \qquad f_S = \text{id} \qquad f_E = \emptyset$$

Inductive case: $x = k + 1$

$$\text{Tr}^{k+1}(\times_{k+1,k+1}) \equiv k + 1$$

The inductive case begins by manipulating $\text{Tr}^{k+1}(\times_{k+1,k+1})$ into a form with traces of only one wire.

$$
\begin{aligned}
\text{Tr}^{k+1}(\times_{k+1,k+1}) &= \text{Tr}^{k+1}(k \otimes \times_{1,n} \otimes 1 \cdot \times_{k,k} \otimes \times_{1,1} \cdot k \otimes \times_{k,1} \otimes 1) && \text{inductive swap} \\
&= \text{Tr}^1(\text{Tr}^k(k \otimes \times_{1,k} \otimes 1 \cdot \times_{k,k} \otimes \times_{1,1} \cdot k \otimes \times_{k,1} \otimes 1)) && \text{definition of trace} \\
&= \text{Tr}^1(\times_{1,k} \otimes 1 \cdot \text{Tr}^k(\times_{k,k} \otimes \times_{1,1}) \cdot \times_{k,1} \otimes 1) && \text{tightening} \\
&= \text{Tr}^1(\times_{1,k} \otimes 1 \cdot \text{Tr}^k(\times_{k,k}) \otimes \times_{1,1} \cdot \times_{k,1} \otimes 1) && \text{superposing} \\
&= \text{Tr}^1(\times_{1,k} \otimes 1 \cdot k \otimes \times_{1,1} \cdot \times_{k,1} \otimes 1) && \text{IH}
\end{aligned}
$$

$$A = \text{Tr}^1(\times_{1,k} \otimes 1 \cdot k \otimes \times_{1,1} \cdot \times_{k,1} \otimes 1)$$

$$
\begin{aligned}
T_A &= [k] + [1] & \leq_f^T &= \leq^k \oslash \leq^1 \\
S_A &= [k] + [1] & \leq_f^S &= \leq^k \oslash \leq^1
\end{aligned}
$$

$$E_A = \emptyset \qquad \Lambda_F = \emptyset \qquad \lambda_A(v) = \blacklozenge \qquad \rho_A(v) = \blacklozenge \qquad \kappa_A(\pi_x[T_A]) = \pi_x S_A$$

Equivalence maps

$$f_T = \text{id} \qquad f_S = \text{id} \qquad f_E = \emptyset$$

B.12 Exchange

For any linear hypergraph $F : x + y + m \rightarrow x + y + n$,

$$\text{Tr}^y(\text{Tr}^x(f)) \equiv \text{Tr}^x(\text{Tr}^y(\times_{y,x} \otimes m \cdot f \cdot \times_{x,y} \otimes n))$$

Zero case I $x = 0, y = k$

$$\text{Tr}^k(\text{Tr}^0(F)) \equiv \text{Tr}^0(\text{Tr}^k(\times_{k,0} \otimes m \cdot F \cdot \times_{0,k} \otimes n))$$

$$
\begin{aligned}
\text{Tr}^k(\text{Tr}^0(F)) &= \text{Tr}^k(F) && \text{definition of trace} \\
&= \text{Tr}^k(k + n \cdot F) && \text{left identity} \\
&= \text{Tr}^k(k \otimes n \cdot F) && \text{bifunctoriality} \\
&= \text{Tr}^k(\times_{k,0} \otimes m \cdot F) && \text{definition of swap} \\
&= \text{Tr}^k(\times_{k,0} \otimes m \cdot F \cdot k + n) && \text{right identity} \\
&= \text{Tr}^k(\times_{k,0} \otimes m \cdot F \cdot k \otimes n) && \text{bifunctoriality} \\
&= \text{Tr}^k(\times_{k,0} \otimes m \cdot F \cdot \times_{0,k} \otimes n) && \text{definition of swap} \\
&= \text{Tr}^0(\text{Tr}^k(\times_{k,0} \otimes m \cdot F \cdot \times_{0,k} \otimes n)) && \text{definition of trace}
\end{aligned}
$$

Zero case II $x = k, y = 0$

$$\mathrm{Tr}^0(\mathrm{Tr}^k(F)) \equiv \mathrm{Tr}^k(\mathrm{Tr}^0(\times_{0,k} \otimes m \cdot f \cdot \times_{k,0} \otimes n))$$

$$
\begin{aligned}
\mathrm{Tr}^0(\mathrm{Tr}^k(f)) &= \mathrm{Tr}^k(F) && \text{definition of trace} \\
&= \mathrm{Tr}^k(k + n \cdot F) && \text{left identity} \\
&= \mathrm{Tr}^k(k \otimes n \cdot F) && \text{bifunctoriality} \\
&= \mathrm{Tr}^k(\times_{0,k} \otimes m \cdot F) && \text{definition of swap} \\
&= \mathrm{Tr}^k(\times_{0,k} \otimes m \cdot F \cdot k + n) && \text{right identity} \\
&= \mathrm{Tr}^k(\times_{0,k} \otimes m \cdot F \cdot k \otimes n) && \text{bifunctoriality} \\
&= \mathrm{Tr}^k(\times_{0,k} \otimes m \cdot F \cdot \times_{k,0} \otimes n) && \text{definition of swap} \\
&= \mathrm{Tr}^k(\mathrm{Tr}^0(\times_{0,k} \otimes m \cdot F \cdot \times_{k,0} \otimes n)) && \text{definition of trace}
\end{aligned}
$$

Base case $x = 1, y = 1$
$$\mathrm{Tr}^1(\mathrm{Tr}^1(F)) \equiv \mathrm{Tr}^1(\mathrm{Tr}^1(\times_{1,1} \cdot F \times_{1,1}))$$

$A = \mathrm{Tr}^1(\mathrm{Tr}^1(F))$

$$
\begin{aligned}
T_A &= T_F - \{\pi_0(S_F^\blacklozenge)), \pi_1(S_F^\blacklozenge))\} & \leq_A^T &= \leq_f^T \\
S_A &= S_F - \{\pi_0(S_F^\blacklozenge)), \pi_1(S_F^\blacklozenge))\} & \leq_A^S &= \leq_f^S \\
E_A &= E_F \qquad \Lambda_A = \Lambda_F
\end{aligned}
$$
$$\lambda_A(v) = \lambda_F \oslash \{v \in T_F^\blacklozenge \mapsto \alpha_A\} \qquad \rho_A = \rho_F(v) = \rho_F \oslash \{v \in S_F^\blacklozenge \mapsto \omega_A\}$$
$$\kappa_A(v) = \kappa_F \oslash \phi$$

where

$$
\phi(v) = \begin{cases}
\kappa_F(\pi_1(S_F^\blacklozenge)) & \text{if } \kappa_F(v) = \pi_0(S_F^\blacklozenge) \wedge \kappa_F(\pi_0(S_F^\blacklozenge)) = \pi_1(S_F^\blacklozenge) \\
\kappa_F(\pi_0(S_F^\blacklozenge)) & \text{if } \kappa_F(v) = \pi_1(S_F^\blacklozenge) \wedge \kappa_F(\pi_1(S_F^\blacklozenge)) = \pi_0(S_F^\blacklozenge) \\
\kappa_F(\pi_1(S_F^\blacklozenge)) & \text{if } \kappa_F(v) = \pi_1(S_F^\blacklozenge) \\
\kappa_F(\pi_0(S_F^\blacklozenge)) & \text{if } \kappa_F(v) = \pi_0(S_F^\blacklozenge)
\end{cases}
$$

$B = \mathrm{Tr}^1(\mathrm{Tr}^1(\times_{1,1} \otimes m \cdot F \cdot \times_{1,1} \otimes n))$

$$
\begin{aligned}
T_B &= [m] + T_F - T_F^\blacklozenge & \leq_B^T &= \leq^m \oslash \leq_F^T \\
S_B &= S_F - S_F^\blacklozenge + [n] & \leq_B^S &= \leq^n \oslash \leq_F^S \\
E_B &= E_F \qquad \Lambda_B = \Lambda_F
\end{aligned}
$$
$$\lambda_B(v) = \lambda_F \oplus \{v \mapsto \blacklozenge \mid v \in [m]\} \qquad \rho_B(v) = \rho_F(v) \oplus \{v \mapsto \blacklozenge \mid v \in [n]\}$$
$$\kappa_B(v) = \kappa_F(v) \oslash \phi$$

where

$$
\phi(v) = \begin{cases}
\pi_{z-2}([n]) & \text{if } v = \pi_y([m]) \wedge \kappa_F(\pi_{2+y}(T_F^{\blacklozenge})) = \pi_1(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_1(S_F^{\blacklozenge})) = \pi_0(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_0(S_F^{\blacklozenge})) = \pi_z(S_F^{\blacklozenge}), z > 1 \\
\kappa_F(\pi_0(S_F^{\blacklozenge})) & \text{if } v = \pi_y([m]) \wedge \kappa_F(\pi_{2+y}(T_F^{\blacklozenge})) = \pi_1(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_1(S_F^{\blacklozenge})) = \pi_0(S_F^{\blacklozenge}) \\
\pi_{y-2}([n]) & \text{if } \kappa_F(v) = \pi_1(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_1(S_F^{\blacklozenge})) = \pi_0(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_0(S_F^{\blacklozenge})) = \pi_y(S_F^{\blacklozenge}), y > 1 \\
\kappa_F(\pi_0(S_F^{\blacklozenge})) & \text{if } \kappa_F(v) = \pi_1(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_1(S_F^{\blacklozenge})) = \pi_0(S_F^{\blacklozenge}) \\
\pi_{y-2}([n]) & \text{if } v = \pi_y([m]) \wedge \kappa_F(\pi_{2+y}(T_F^{\blacklozenge})) = \pi_0(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_0(S_F^{\blacklozenge})) = \pi_1(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_1(S_F^{\blacklozenge})) = \pi_z(S_F^{\blacklozenge}), z > 1 \\
\kappa_F(\pi_1(S_F^{\blacklozenge})) & \text{if } v = \pi_y([m]) \wedge \kappa_F(\pi_{2+y}(T_F^{\blacklozenge})) = \pi_0(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_0(S_F^{\blacklozenge})) = \pi_1(S_F^{\blacklozenge}) \\
\pi_{z-2}([n]) & \text{if } v = \pi_y([m]) \wedge \kappa_F(\pi_{2+y}(T_F^{\blacklozenge})) = \pi_0(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_0(S_F^{\blacklozenge})) = \pi_z(S_F^{\blacklozenge}), z > 1 \\
\kappa_F(\pi_0(S_F^{\blacklozenge})) & \text{if } v = \pi_y([m]) \wedge \kappa_F(\pi_{2+y}(T_F^{\blacklozenge})) = \pi_0(S_F^{\blacklozenge}) \\
\pi_{y-2}([n]) & \text{if } \kappa_F(v) = \pi_0(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_0(S_F^{\blacklozenge})) = \pi_1(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_1(S_F^{\blacklozenge})) = \pi_y(S_F^{\blacklozenge}), y > 1 \\
\kappa_F(\pi_1(S_F^{\blacklozenge})) & \text{if } \kappa_F(v) = \pi_0(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_0(S_F^{\blacklozenge})) = \pi_1(S_F^{\blacklozenge}) \\
\pi_{y-2}([n]) & \text{if } \kappa_F(v) = \pi_0(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_0(S_F^{\blacklozenge})) = \pi_y(S_F^{\blacklozenge}), y > 1 \\
\kappa_F(\pi_0(S_F^{\blacklozenge})) & \text{if } \kappa_F(v) = \pi_0(S_F^{\blacklozenge}) \\
\pi_{y-2}([n]) & \text{if } v = \pi_y([m]) \wedge \kappa_F(\pi_{2+y}(T_F^{\blacklozenge})) = \pi_1(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_1(S_F^{\blacklozenge})) = \pi_z(S_F^{\blacklozenge}), z > 1 \\
\kappa_F(\pi_1(S_F^{\blacklozenge})) & \text{if } v = \pi_y([m]) \wedge \kappa_F(\pi_{2+y}(T_F^{\blacklozenge})) = \pi_1(S_F^{\blacklozenge}) \\
\pi_{y-2}([n]) & \text{if } \kappa_F(v) = \pi_1(S_F^{\blacklozenge}) \\
& \qquad \wedge \kappa_F(\pi_1(S_F^{\blacklozenge})) = \pi_y(S_F^{\blacklozenge}), y > 1 \\
\kappa_F(\pi_1(S_F^{\blacklozenge})) & \text{if } \kappa_F(v) = \pi_1(S_F^{\blacklozenge}) \\
\pi_{z-2}([n]) & \text{if } v = \pi_y([m]) \wedge \kappa_F(\pi_{2+y}(T_F^{\blacklozenge})) = \pi_z(S_F^{\blacklozenge}), z > 1 \\
\pi_{y-2}([n]) & \text{if } \kappa_F(v) = \pi_y(S_F^{\blacklozenge}), y > 1 \\
\kappa_F(\pi_{2+y}(T_F^{\blacklozenge})) & \text{if } v = \pi_y([m])
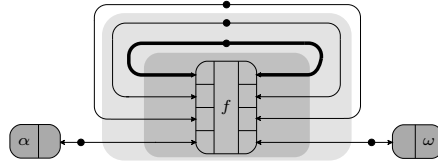\end{cases}
$$

Equivalence maps

$$f_T = \begin{cases} \pi_z([m]) & \text{if } v = \pi_{2+z}(T_F^{\blacklozenge}) \\ \text{id} & \text{otherwise} \end{cases} \qquad f_S = \begin{cases} \pi_z([n]) & \text{if } v = \pi_{2+z}(S_F^{\blacklozenge}) \\ \text{id} & \text{otherwise} \end{cases}$$
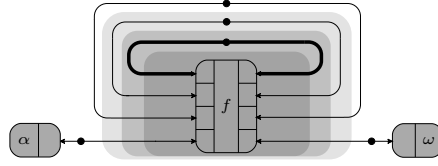
$$f_E = \text{id}$$

Inductive case I: $x = k + 1, y = 1$

$$\text{Tr}^1(\text{Tr}^{k'+1}(f)) \equiv \text{Tr}^{k'+1}(\text{Tr}^{k+1}(\times_{1,k+1} \otimes m \cdot F \cdot \times_{k+1,1} \otimes n))$$
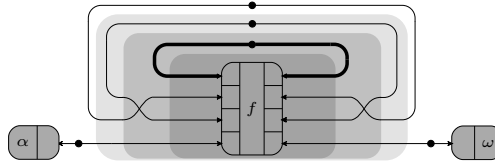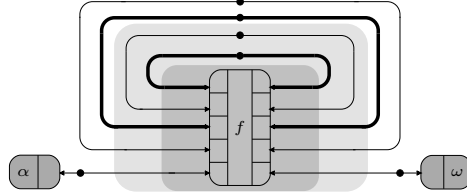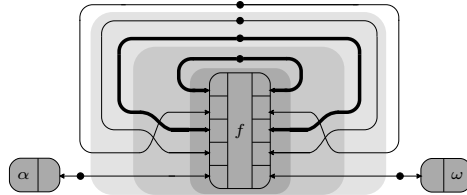
In these diagrams, the 'thicker' wire represents k wires.

By definition of trace:

By base case:

By tightening:
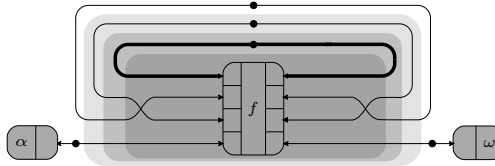
By inductive hypothesis:
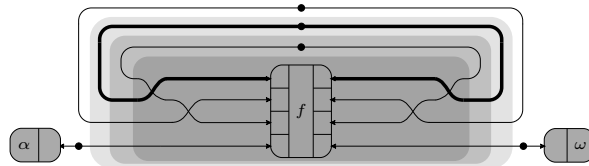
By various algebraic manipulations and definition of trace:
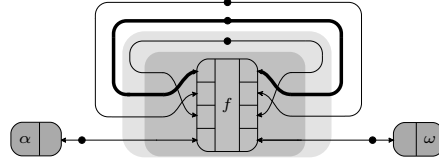
Inductive case II $x = 1, y = k + 1$

$$\mathrm{Tr}^1(\mathrm{Tr}^{k'+1}(F)) \equiv \mathrm{Tr}^{k'+1}(\mathrm{Tr}^1(\times_{k'+1,1} \otimes m \cdot F \cdot \times_{1,k'+1} \otimes n))$$

The uses a similar strategy to the first inductive case.

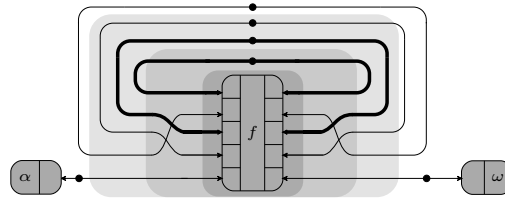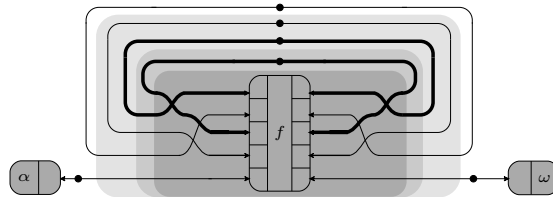**Inductive case III** $x = k + 1, y = k' + 1$

$$\mathrm{Tr}^{k+1}(\mathrm{Tr}^{k'+1}(F)) \equiv \mathrm{Tr}^{k'+1}(\mathrm{Tr}^{k+1}(\times_{k'+1,k+1} \otimes m \cdot F \cdot \times_{k+1,k'+1} \otimes n))$$

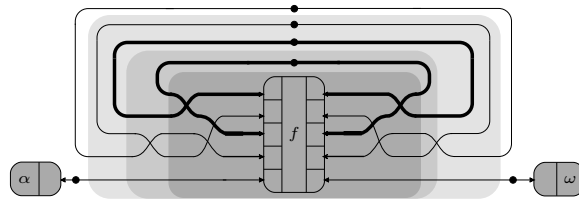By definition of trace and inductive case II:
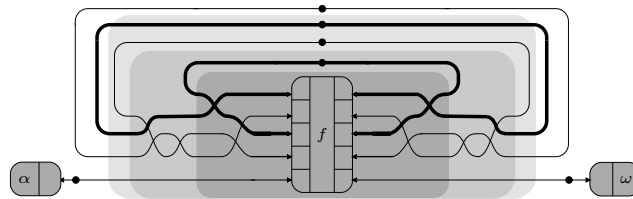

By tightening:

By definition of trace and inductive hypothesis:
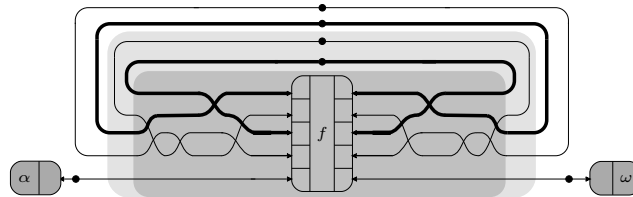
By definition of trace and base case:

By definition of trace and inductive case I:

By tightening:

By various algebraic manipulations and definition of trace: