The Graphical Language of Symmetric Traced Monoidal Categories

George Kaye

23 November 2020

University of Birmingham

Introduction

Traditionally in mathematics, diagrams have not been considered first-class citizens.

Recently the development of formal diagrammatic semantics have showed us diagrams are much more powerful, and can even be used to formulate proofs.

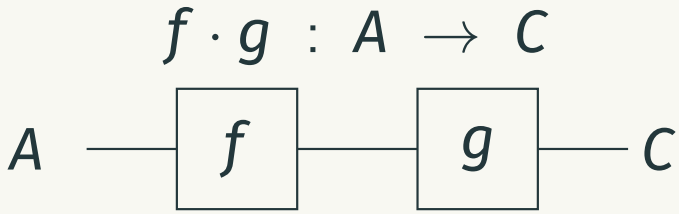
Mainly using compact closed categories, but we will examine a different framework – symmetric traced monoidal categories.

Graphical languages for monoidal categories

Categories

We draw morphisms are boxes – the ‘flow’ is from left to right:

Composition is written in diagram order:

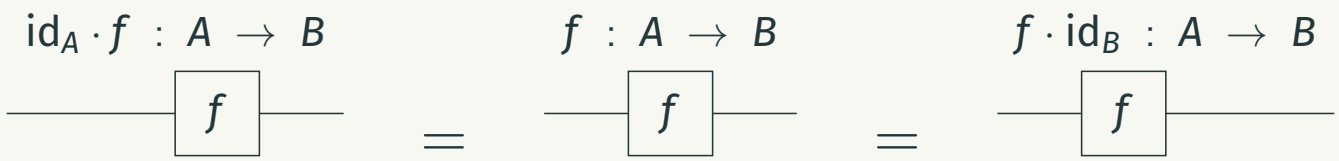


We normally don’t bother labelling the wires.

Categories – identity morphisms

Identity morphisms are drawn as wires:

$$\mathrm{id}_A : A \to A \qquad \mathrm{id}_B : B \to B$$



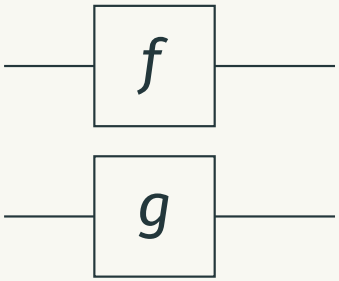
Monoidal categories

A monoidal category is a category with the operation of tensor product $\otimes$ (parallel composition) and a unit object I.

Tensor is written top to bottom.

$f : A \to B$ $\quad$ $g : C \to D$

$f \otimes g : A \otimes C \to B \otimes D$

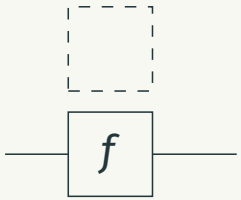


Monoidal categories – monoidal unit

The unit of tensor is id_I, drawn as empty space:

$$\mathrm{id}_I : I \to I$$

$\mathrm{id}_I \otimes f : I \otimes A \to I \otimes B$

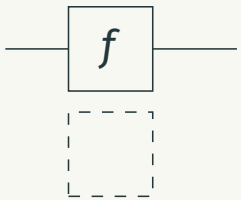


$=$

$f : A \to B$

$=$

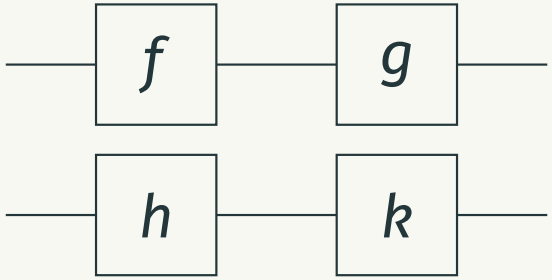
$f \otimes \mathrm{id}_I : A \otimes I \to B \otimes I$



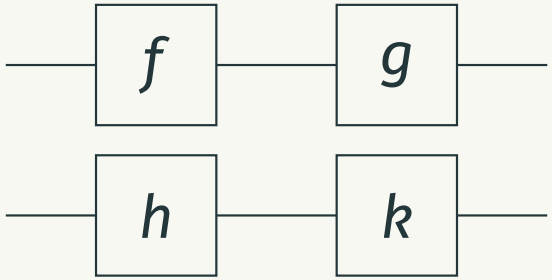
Monoidal categories – functoriality

An important axiom is $(f \otimes h) \cdot (g \otimes k) = (f \cdot g) \otimes (h \cdot k)$

$$(f \otimes h) \cdot (g \otimes k) \ : \ A \otimes D \ \to \ C \otimes F$$



$$(f \cdot g) \otimes (h \cdot k) \ : \ A \otimes D \ \to \ C \otimes F$$



We did it – bureaucracy is no more!

Symmetric monoidal categories

A symmetric monoidal category is a monoidal category with a special morphism called a symmetry for each pair of objects.

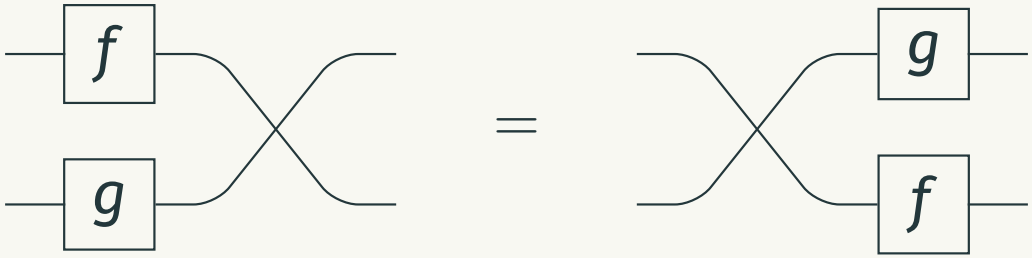
Graphically, we represent this as crossing wires:

$$\times_{A,B} : A \otimes B \to B \otimes A$$

The symmetry satisfies some axioms...

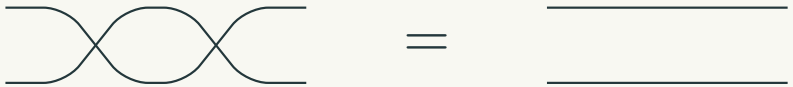
Symmetric monoidal categories – axioms

Naturality



Hexagon

Self-inverse



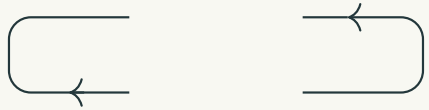
Bending the wires

So far all of our wires have been travelling across the page from left to right.

Sometimes we might want to go the other way – for example to model feedback or recursion.

A common approach is to use compact closed categories, where every object A has a dual written A^*, drawn as a wire travelling from right to left.

Each object A in a compact closed category is equipped with a cup $I \to A \otimes A^*$ and cap $A^* \otimes A \to I$ to bend wires around.



Bending the wires

Because wires in compact closed categories can flow in either direction, morphisms have a notion of a general interface port, where any two ports with the same type can match.

Conversely, systems such as digital circuits have a rigid notion of causality, where outputs of morphisms must connect to inputs.

A compact closed category is not the answer here, and a different way of bending the wires is required...

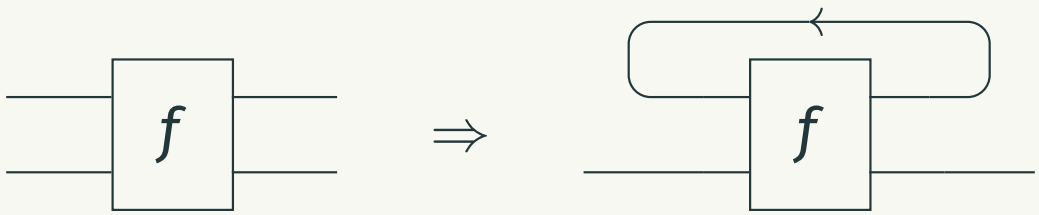
Symmetric traced monoidal categories

A symmetric traced monoidal category (STMC) is a symmetric monoidal category $\mathcal{C}$ equipped with a family of functions

$$\mathrm{Tr}^X_{A,B} \ : \ \mathcal{C}(X \otimes A, X \otimes B) \ \to \ \mathcal{C}(A, B)$$

Effectively, if we have a morphism $f \ : \ X \otimes A \ \to \ X \otimes B$ we can trace it with X wires to form the morphism $\mathrm{Tr}^X(f) \ : \ A \ \to \ B$.

This is represented graphically by joining up the 'first' output of X with the 'first' input.

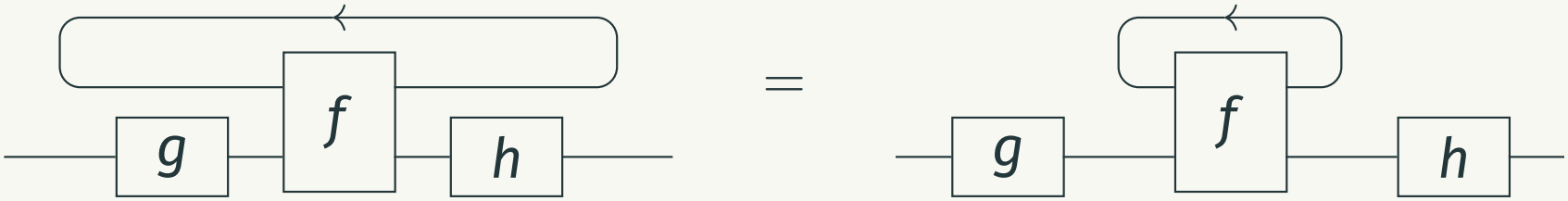


Again we have some axioms...

Symmetric traced monoidal categories – axioms

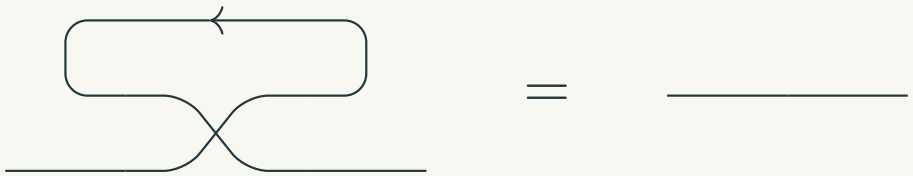
Tightening

$$\mathrm{Tr}^X(\mathrm{id}_X \otimes g \cdot f \cdot \mathrm{id}_X \otimes h) = g \cdot \mathrm{Tr}^X(f) \cdot h$$



Yanking

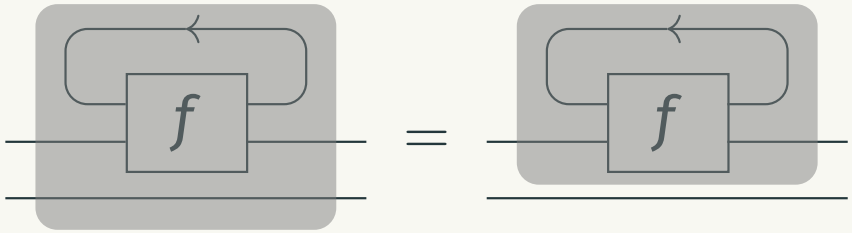
$$\mathrm{Tr}^X(\times_{X,X}) = \mathrm{id}_X$$



Symmetric traced monoidal categories – axioms

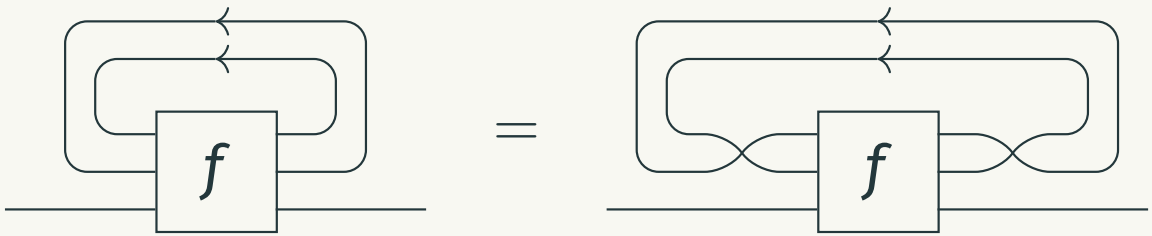
Superposing

$$\mathrm{Tr}^X(f \otimes \mathrm{id}_A) = \mathrm{Tr}^X(f) \otimes \mathrm{id}_A$$



Exchange

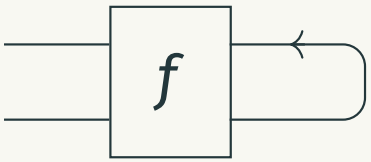
$$\mathrm{Tr}^Y(\mathrm{Tr}^X(f)) = \mathrm{Tr}^X(\mathrm{Tr}^Y(\times_{Y,X} \otimes \mathrm{id}_A \cdot f \cdot \times_{X,Y} \otimes \mathrm{id}_A))$$



Symmetric traced vs compact closed

We can embed any STMC into a compact closed category by constructing the trace from a cup and a cap:

But not every compact closed diagram is valid in a STMC!

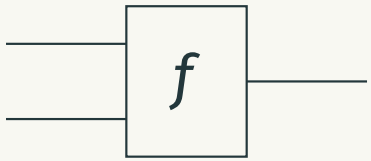


PROPs

An especially natural framework for graphical calculi is that of PROPs, monoidal categories where the objects are natural numbers and tensor product is addition.

This is nice for us as it allows us to draw an object $m \in \mathbb{N}$ as m wires in our diagrams.

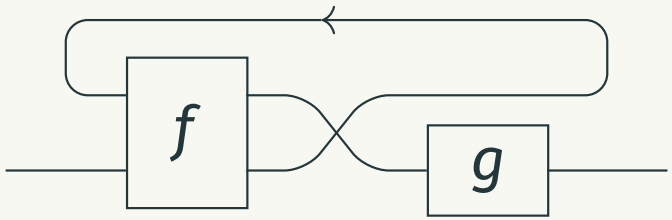
$$f : 2 \to 1$$



Free traced PROPs

A monoidal signature Σ in a PROP is a set of generators, equipped with two functions $dom : \Sigma \to \mathbb{N}$ and $cod : \Sigma \to \mathbb{N}$.

A free traced PROP generated over a signature Σ has as morphisms the combinations of generators using composition, tensor, symmetry and trace.



These morphisms are often called terms. From here on, we will fix an arbitrary PROP $\mathbf{Term}_\Sigma$, generated freely over some signature Σ.

Adding extra structure

As we have seen, graphical calculi ‘absorb’ the painful axioms of categories such as associativity or functoriality.

Does this solve all of our problems?

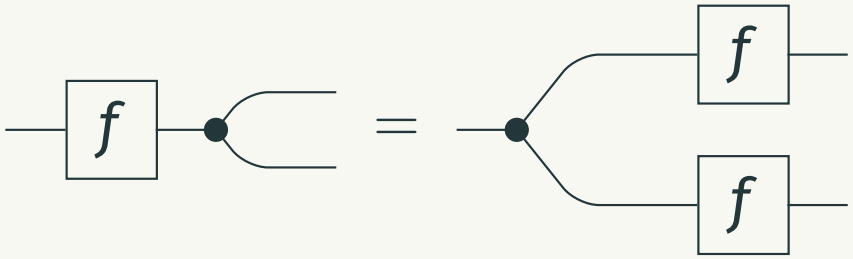
No.

We often work in categories with extra structure – additional axioms that cannot be absorbed by the graphical notation!

Adding extra structure

As an example, we will take the axiom of naturality in a Cartesian category, where $\Delta_n : n \to n \otimes n$ is the diagonal for 'copying' morphisms.

$$f \cdot \Delta_n = \Delta_n \cdot f \otimes f$$

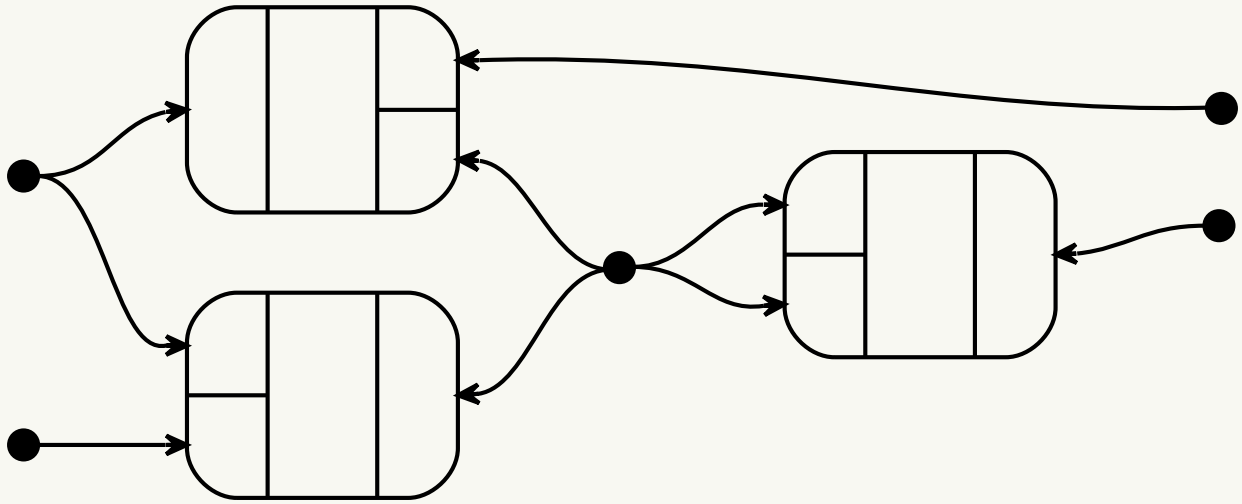


Fortunately we can still implement these axioms into our language, but as graph rewrites.

And for that we're going to need to define our diagrams a little more rigorously...

Hypergraphs

Simple hypergraphs



Simple hypergraphs

Let $\mathbb{A}$ be a countably infinite set of 'atoms', and for a set X let $X^\star$ be the free monoid on X.

Definition (Simple hypergraph)

A simple hypergraph is a tuple $H = (V, E, \mathrm{s}, \mathrm{t})$ where

- $V \subset \mathbb{A}$ is a set of vertices
- $E \subset \mathbb{A}$ is a set of edges
- $\mathrm{s}, \mathrm{t} : E \to V^\star$ are functions returning the list of sources and targets of an edge.

Labelled simple hypergraphs

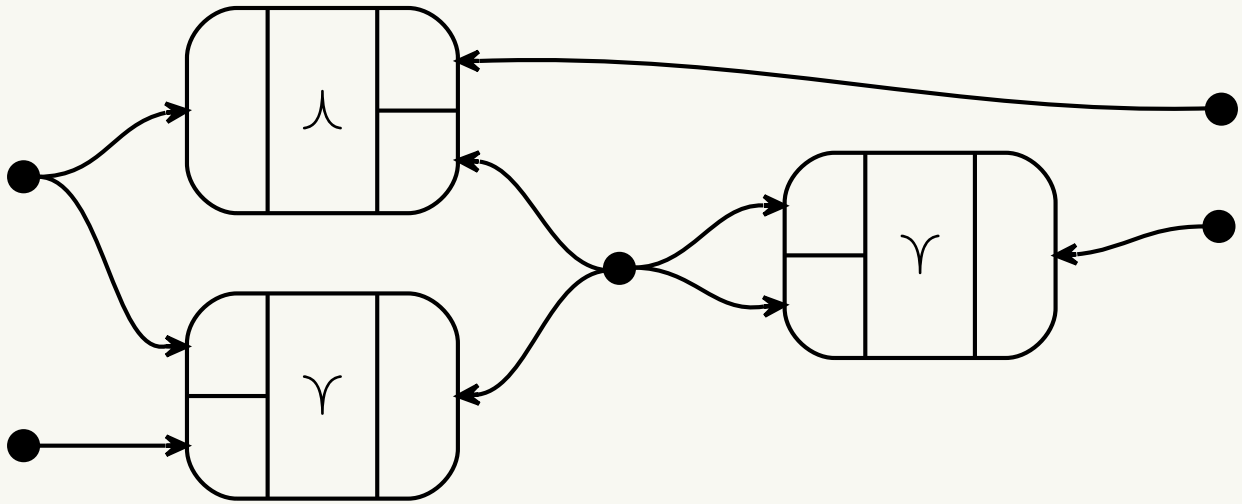
A hypergraph signature is a set of edge labels Σ equipped with two functions $\mathsf{dom}, \mathsf{cod} : \Sigma \to \mathbb{N}$.

A labelled simple hypergraph is a simple hypergraph equipped with a hypergraph signature and a labelling function $\Lambda : E \to \Sigma$, such that if $\Lambda(e) = \sigma$, then $\mathsf{dom}(\sigma) = |\mathsf{s}(e)|$ and $\mathsf{cod}(\sigma) = |\mathsf{t}(e)|$.

Simple hypergraph example

Throughout this talk, we will use the example signature

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



Category of simple hypergraphs

A simple hypergraph homomorphism $h : F \to G$ consists of functions

$$h_V : V_F \to V_G \qquad h_E : E_F \to E_G$$

such that sources, targets and labels are preserved.

If h_V and h_E are bijective then F and G are isomorphic $F \equiv G$.

These are the morphisms in the category of simple hypergraphs **SHyp**, which is defined as a presheaf category.

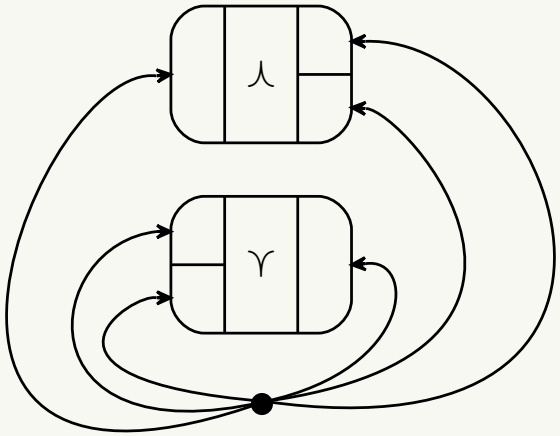
Category of simple hypergraphs

Definition

Let **SHyp** be the functor category $[\mathbf{X}, \mathbf{Set}]$, where the objects of $\mathbf{X}$ are pairs of natural numbers (m, n), along with one extra object $\star$. For each object $x = (m, n)$, there are $m + n$ arrows from x to $\star$.

Category of simple hypergraphs

Hypergraph signatures can be seen as simple hypergraphs with a single vertex v and edges for each label $m \to n$ in the signature, with v appearing m (n respectively) in its sources (targets respectively).



The category of labelled simple hypergraphs $\mathbf{SHyp}_\Sigma = \mathbf{SHyp}/\Sigma$ is defined as the slice category over a hypergraph signature Σ.

Simple hypergraphs are not enough

As we have seen, vertices in hypergraphs can connect to an arbitrary number of edges.

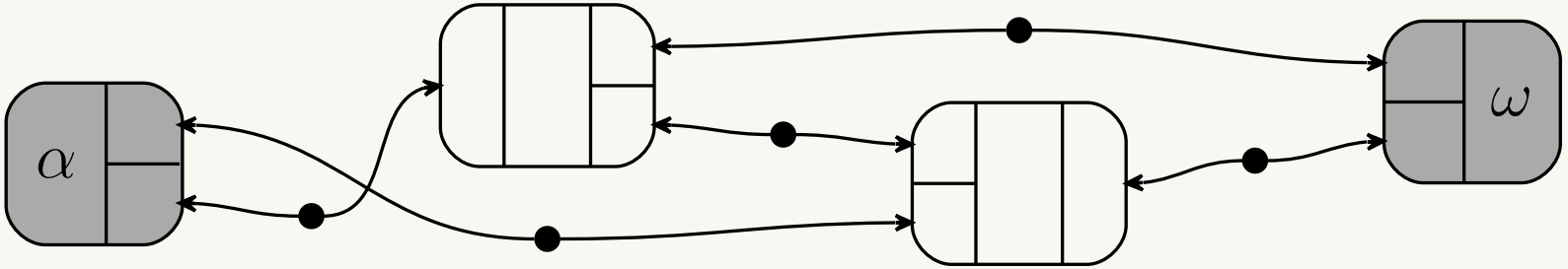
But in string diagrams, wires cannot fork without an explicit morphism or additional structure.

We also want to keep track of the interfaces of our hypergraph.

We could add them using ordered cospans, but we prefer to build them directly into the hypergraph itself.

To fix these problems we introduce a restriction on simple hypergraphs that we call linear hypergraphs.

Linear hypergraphs



Linear hypergraphs

Definition (Linear hypergraphs)

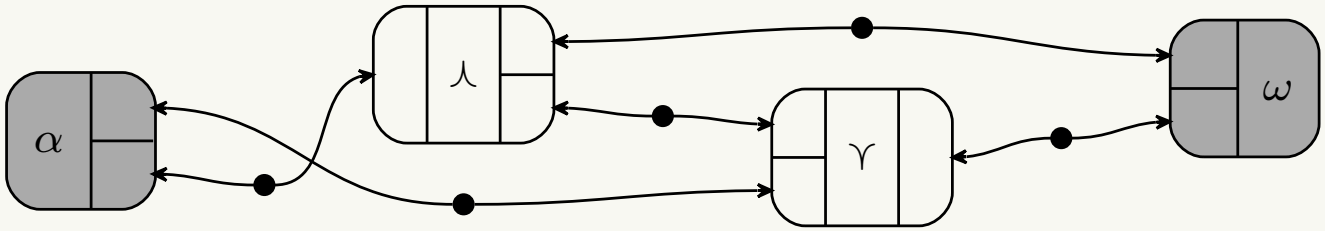
A linear hypergraph is a tuple $H = (T, S, E, \lambda, \rho, \kappa)$ where

- $T, S \subset \mathbb{A}$ are two finite disjoint totally ordered sets of equal size, 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 functions indicating the 'left' and 'right' connections of vertices.
- $\kappa : T \to S$ is a *connections* bijection linking target vertices to source vertices.

Labelled linear hypergraphs

As with simple hypergraphs we can define *labelled linear hypergraphs* over a signature Σ with labelling function Λ.

$$\Sigma = \{\lambda : 1 \to 2, \upsilon : 2 \to 1\}$$



Category of linear hypergraphs

A labelled linear hypergraph homomorphism $h : F \to G$ consists of functions

$$h_T : T_F \to T_G \qquad h_S : S_F \to S_G \qquad h_E : E_F \to E_G$$

such that sources, targets, connections and labels are preserved.

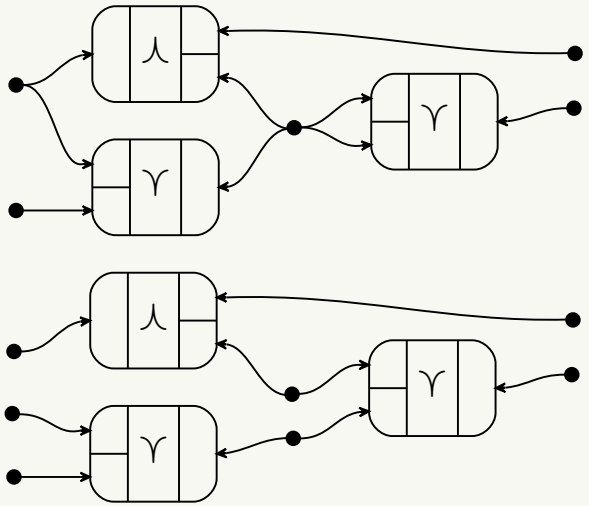
If h_T, h_S and h_E are bijective then F and G are isomorphic $F \equiv G$.

Linear hypergraphs form a category $\mathbf{LHyp}_\Sigma$ with objects the hypergraphs over signature Σ and morphisms the hypergraph homomorphisms between them.

Category of linear hypergraphs

Linear hypergraphs are merely simple hypergraphs with some restrictions, so they form a subcategory.

Intuitively, linear hypergraphs are simple hypergraphs in which each vertex is in the sources and targets of at most one edge.



no

yes

Proposition

LHyp$_\Sigma$ *is a full subcategory of* ***SHyp***$_\Sigma$.

Soundness and completeness

Soundness

We propose linear hypergraphs as a graphical language for STMCs.

Definition (Soundness)

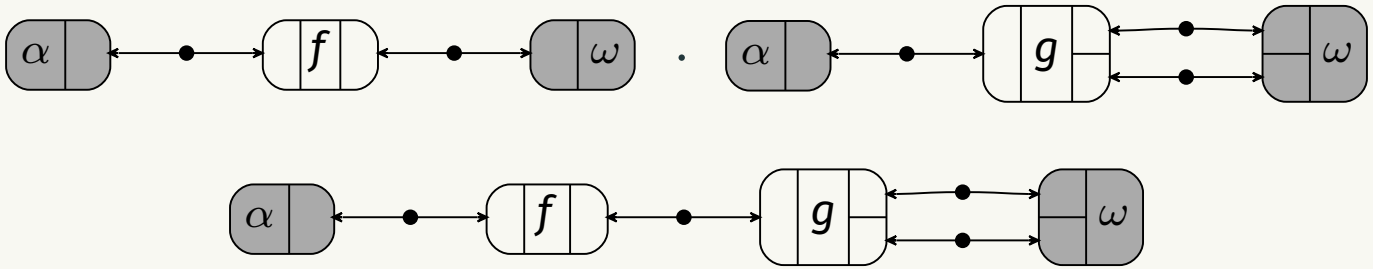
A graphical language is sound if for any morphism $f, g \in \mathbf{Term}_\Sigma$, if $f = g$ under the equational theory of the category, then their interpretations as linear hypergraphs under some interpretation $[\![-]\!]$ are isomorphic $[\![f]\!] \equiv [\![g]\!]$.

To show this, we need to examine that all of the axioms of STMCs also hold for linear hypergraphs.

Therefore we need to define the operations of composition, tensor, symmetry and trace for linear hypergraphs.

Composition

$$F \cdot G : 1 \to 2$$

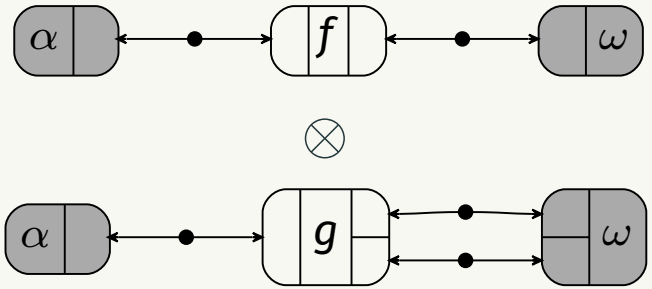


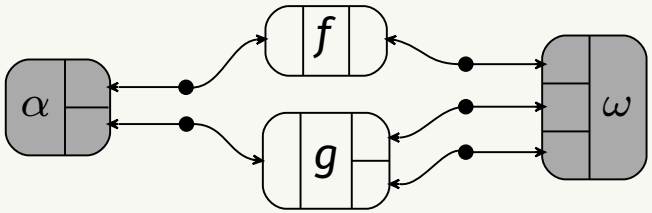
The unit of composition is the identity hypergraph.

$$\mathrm{id}_1 : 1 \to 1$$

Monoidal tensor

$$F \otimes G \;:\; 1+1 \;\to\; 1+2$$



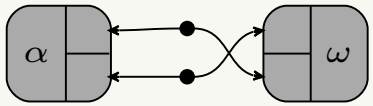


The unit of tensor is the empty hypergraph.

$$\mathrm{id}_o \;:\; o \;\to\; o$$

Symmetry

$$\times_{1,1} : 2 \to 2$$

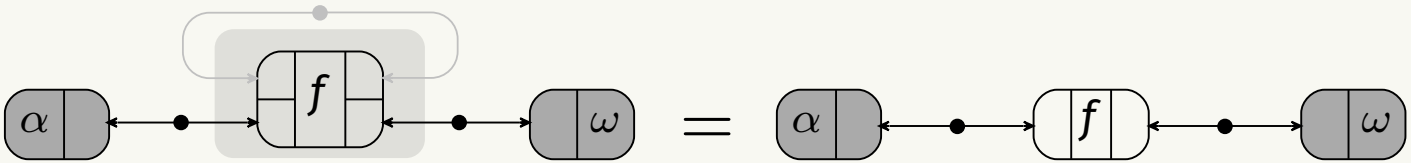


We can build up larger symmetries by composing multiple copies of $\times_{1,1}$ together with identities.

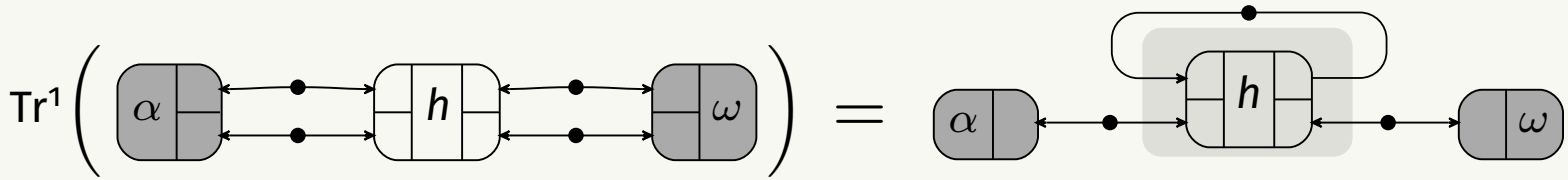
Trace

Trace is defined recursively over the number of wires.

$$\mathrm{Tr}^0(F) \;:\; 1 \to 1$$

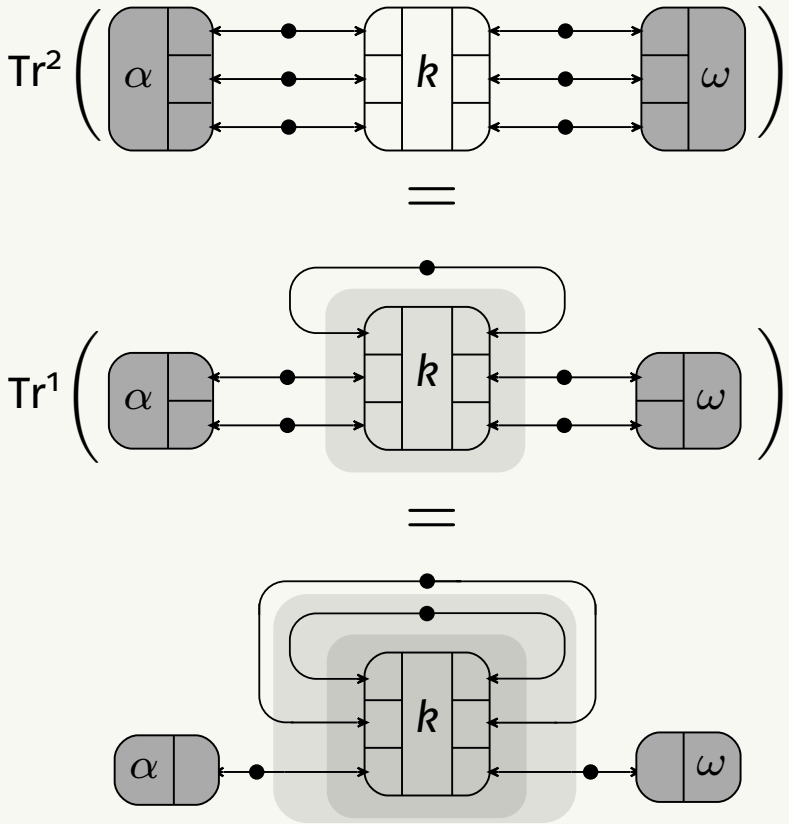


$$\mathrm{Tr}^1(H) \;:\; 1 \to 1$$



Trace

$$\mathrm{Tr}^2(K) : 1 \to 1$$



Interpreting terms as graphs

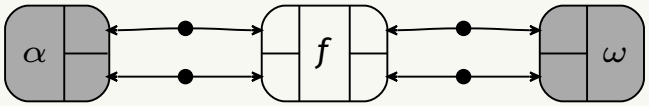
We assemble our hypergraphs into a traced PROP $\mathbf{HypTerm}_\Sigma$, in which the morphisms $m \to n$ are hypergraphs of type $m \to n$ over the signature Σ, with operations defined as previously.

Definition

We define the interpretation functor as the identity-on-objects traced monoidal functor $[\![-]\!] : \mathbf{Term}_\Sigma \to \mathbf{HypTerm}_\Sigma$.

Interpreting terms as graphs

For a generator f, $[\![f]\!]$ is defined as:



Identity and symmetry morphisms translate cleanly into their hypergraph versions:

$$[\![\mathsf{id}_n]\!] = \mathsf{id}_n \qquad [\![\times_{m,n}]\!] = \times_{m,n}$$

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

$$[\![f \cdot g]\!] = [\![f]\!] \cdot [\![g]\!] \qquad [\![f \otimes g]\!] = [\![f]\!] \otimes [\![g]\!] \qquad [\![\mathsf{Tr}^X(f)]\!] = \mathsf{Tr}^X([\![f]\!])$$

Soundness

To ensure soundness all axioms of STMCs must still hold in the language of hypergraphs.

They do.

Theorem (Soundness)

For any morphism $f, g \in \mathbf{Term}_\Sigma$, if $f = g$ under the equational theory of the category, then their interpretations as linear hypergraphs are isomorphic.

Completeness

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

Definition (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 $[\![f]\!] \equiv F$.

There are two steps to this: definability and coherence.

Definability

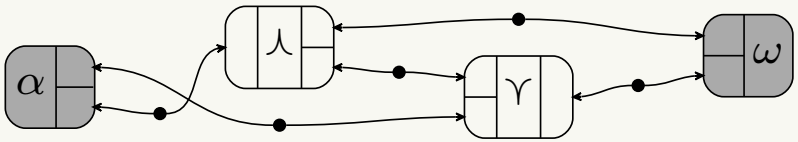
Definition (Definability)

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 functor term : $\mathbf{HypTerm}_\Sigma \to \mathbf{Term}_\Sigma$, then

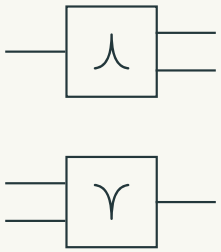
$$[\![\mathrm{term}(F)]\!] \equiv F.$$

Stacking

First we set an order $\leq$ on our edges and stack them in a tensor.

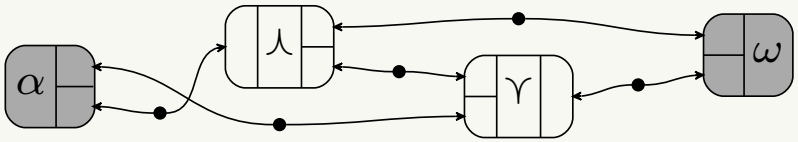


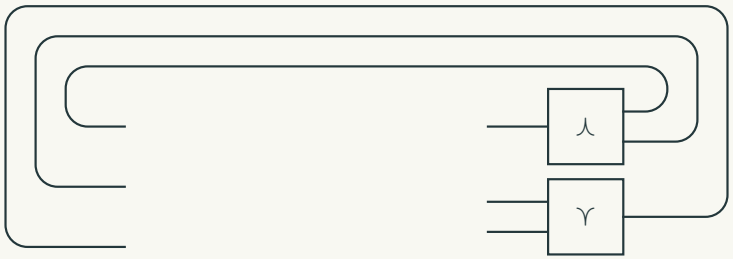
$$\lambda \otimes \gamma$$



Tracing

Then we trace around all the outputs of the stack:

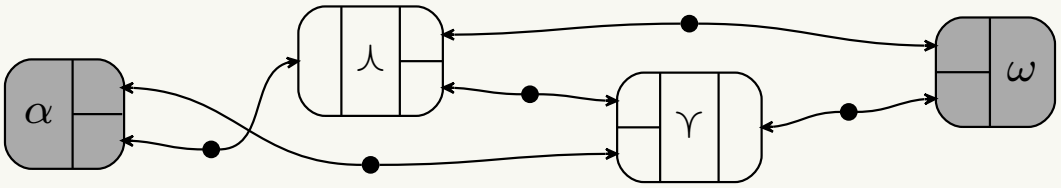


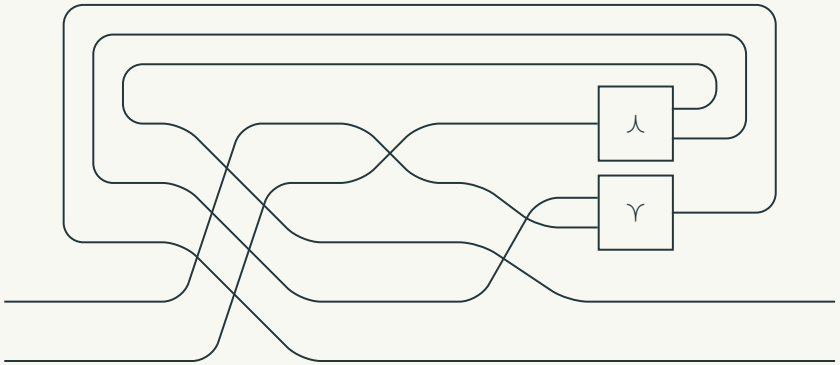


$\mathsf{Tr}^3(? \cdot ʎ \otimes ϒ)$

Shuffling

We introduce the input wires, and then shuffle everything so all the wires connect to the right place.





$$\mathrm{Tr}^3(\times_{3,2} \cdot \times_{1,1} \otimes 3 \cdot 1 \otimes \times_{2,1} \otimes 1 \cdot ʎ \otimes ϒ \otimes 2)$$

(**Exercise:** follow around the wires and make sure this is true)

Definability

Definition

We define the the identity-on-objects traced monoidal functor

$$\mathrm{term}_{\leq} \;:\; \mathbf{HypTerm}_{\Sigma} \;\to\; \mathbf{Term}_{\Sigma}$$

defined for a linear hypergraph F with given edge order $\leq$ by following the procedure illustrated.

Proposition (Definability)

For every well-formed hypergraph F then $[\![\mathrm{term}_{\leq}(F)]\!] \equiv F$.

Coherence

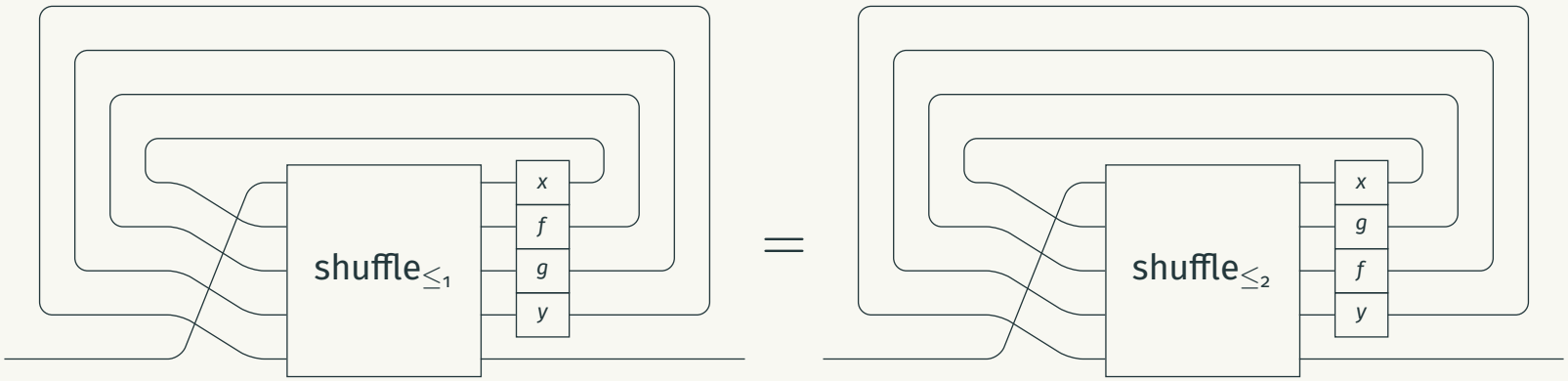
But we cannot conclude completeness yet! Since there are multiple orders on the edges we can set, we must show coherence.

Definition (Coherence)

Hypergraphs are coherent if for any well-formed hypergraph F and any two orders on its edges $\leq_1, \leq_2$, $\mathrm{term}_{\leq_1}(F) = \mathrm{term}_{\leq_2}(F)$ by the equations of the STMC.

Coherence

Fortunately, we just need to show it for swapping two edges.



Proposition (Coherence)

For all orderings of edges $\leq_x$ on a hypergraph F,

$$\mathrm{term}_{\leq_1}(F) = \mathrm{term}_{\leq_2}(F) = \cdots = \mathrm{term}_{\leq_n}(F)$$

Completeness

Theorem (Completeness)

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

Graph rewriting

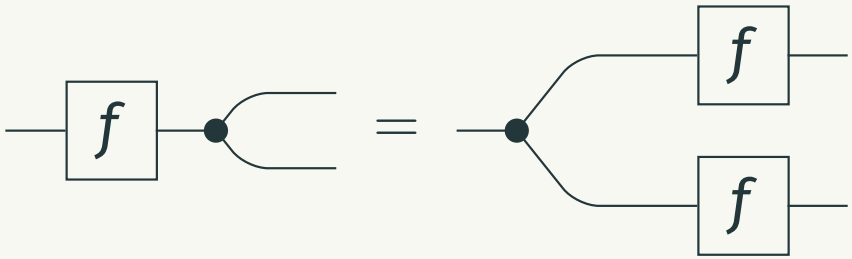
Rewrite rules

Extra structure on categories is often expressed as additional axioms.

These axioms can be expressed as rewrite rules in $\mathbf{LHyp}_\Sigma$.

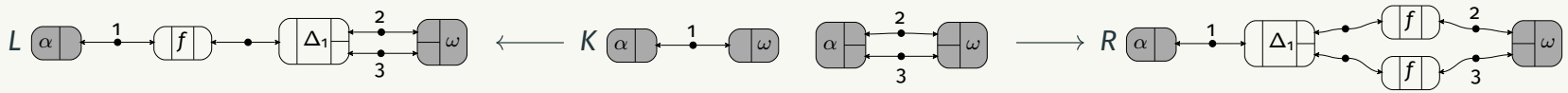
Recall the axiom of naturality in a Cartesian category, where $\Delta_n : n \to n \otimes n$ is the diagonal for 'copying' morphisms.

$$f \cdot \Delta_n = \Delta_n \cdot f \otimes f$$



Rewrite rules

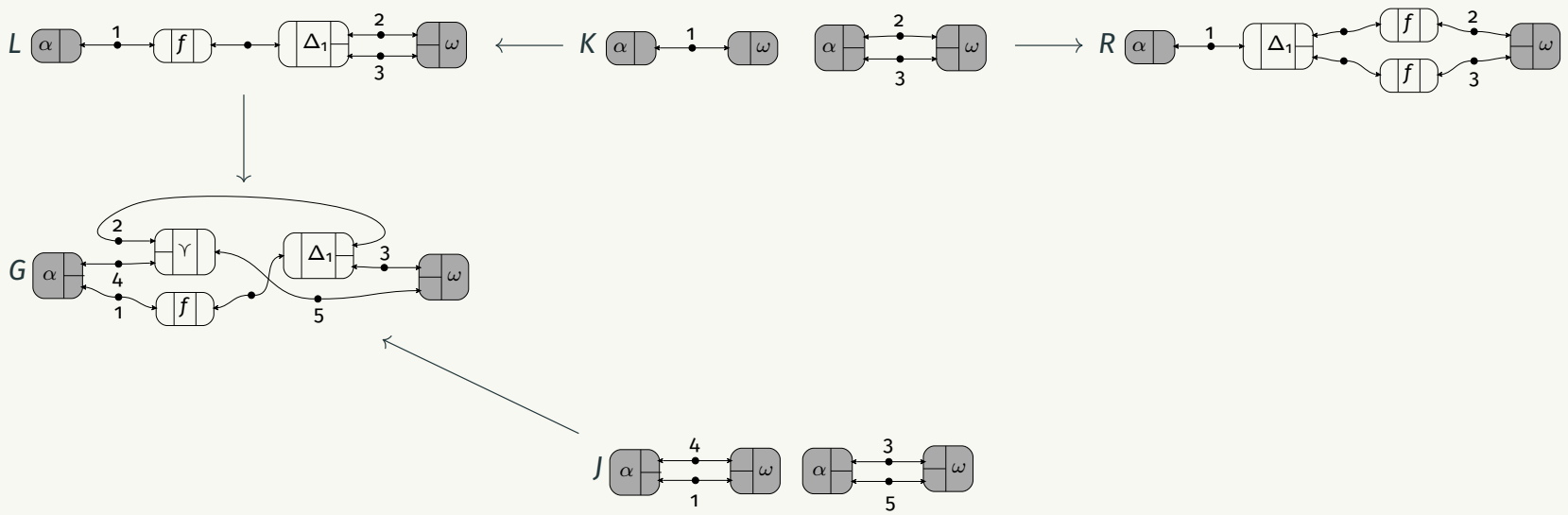
A rewrite rule $L \Rightarrow R$ in $\mathbf{LHyp}_\Sigma$ is a span of monomorphsisms $L \leftarrow K \rightarrow R$, where K is the 'common interface' of L and R.



For a set of axioms $\mathcal{E} \in \mathbf{Term}_\Sigma$, we write $[\![\mathcal{E}]\!]$ for their conversion into spans in $\mathbf{LHyp}_\Sigma$.

DPO rewriting

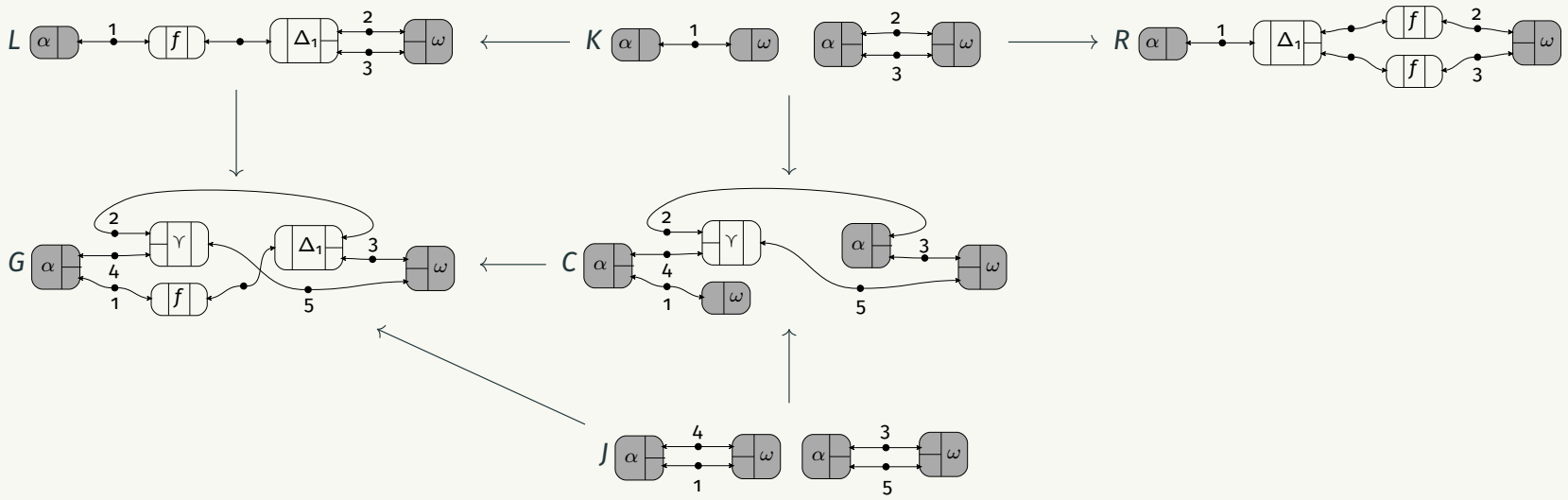
To perform DPO rewriting on a linear hypergraph G with 'interface' J, we must first identify a matching monomorphism $L \to G$.



DPO rewriting

We then compute the pushout complement $K \to C \to G$.

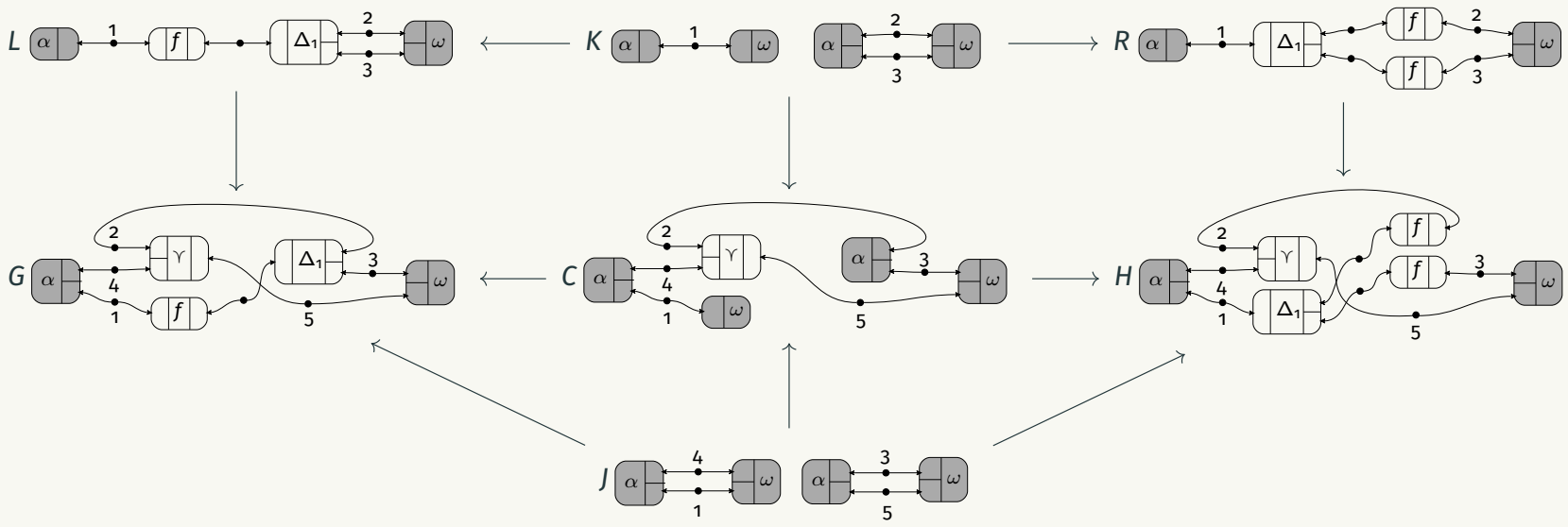
The hypergraph C is effectively 'G with L cut out'.



DPO rewriting

Then we perform a pushout on $C \leftarrow K \rightarrow R$ to obtain our final hypergraph H.

This is 'G with the subgraph L replaced with R.



We write $G \rightsquigarrow_{[\![\mathcal{E}]\!]} H$ if rewriting can be performed in this way.

Adhesive categories

Not all structures are compatible with DPO rewriting – most importantly, the pushout complement must be unique.

The framework of adhesive categories is often used to ensure that pushout complements are always unique, if they exist.

Some fun facts about adhesivity:

- **Set** is adhesive
- If $\mathbf{C}$ is adhesive then so is any functor category $[\mathbf{X}, \mathbf{C}]$.
- If $\mathbf{C}$ is adhesive then so is any slice category $\mathbf{C}/\mathbf{X}$.

Adhesive categories

Unfortunately, $\mathbf{LHyp}_\Sigma$ is not adhesive, as not all pushouts along monomorphsisms exist.

If only there were a slightly weaker definition...

Definition (Partial adhesive categories (Kissinger))

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.

Partial adhesive categories

We know that $\mathbf{LHyp}_\Sigma$ is a full subcategory of the category of simple hypergraphs $\mathbf{SHyp}_\Sigma$. $\mathbf{SHyp}_\Sigma$ is a slice of a presheaf category.

Proposition

$\mathbf{SHyp}_\Sigma$ is an adhesive category.

I does not collapse any vertices or edges.

Proposition

The inclusion functor $I : \mathbf{SHyp}_\Sigma \to \mathbf{LHyp}_\Sigma$ preserves monomorphisms.

Partial adhesive categories

Theorem

$\mathbf{LHyp}_\Sigma$ is a partial adhesive category.

A partial adhesive category has unique pushout complements for a certain class of spans.

In our case, this includes our rewrite rules!

Theorem

For any terms $f, g \in \mathbf{Term}_\Sigma$ and set of axioms $\mathcal{E}$, $f = g$ by the equations of the STMC and $\mathcal{E}$ if and only if $[\![f]\!] \rightsquigarrow_{[\![\mathcal{E}]\!]} [\![g]\!]$.

Generalisation

We have a sound and complete graphical calculus for traced PROPS with extra structure.

Can we generalise this to form a graphical language for arbitrary STMCs?

Yes, we just add vertex labels to represent the objects in our category and ensure our operations are all compatible with this.

Conclusion

We have defined a combinatorial graphical language for symmetric traced monoidal categories, and shown that it is sound and complete.

This allows us to reason in STMCs purely graphically.

Even when adding extra axioms to our category, we can still reason graphically with graph rewrites, since linear hypergraphs form a partial adhesive category.

To use this language for digital circuits, all we need to do is formulate the appropriate axioms as rewrite rules.

References i

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.

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.

References ii


M. Hasegawa.
On traced monoidal closed categories.
Mathematical Structures in Computer Science, 19(2):217–244, 2009.


A. Kissinger.
Pictures of processes: Automated graph rewriting for monoidal categories and applications to quantum computing, 2012.


P. Selinger.
A survey of graphical languages for monoidal categories.
In *New structures for physics*, pages 289–355. Springer, 2010.