

REWRITING MODULO TRACED COMONOID STRUCTURE*

DAN R. GHICA AND GEORGE KAYE

University of Birmingham, UK
e-mail address: d.r.ghica@bham.ac.uk, georgejkaye@gmail.com

ABSTRACT. In this paper we adapt previous work on rewriting string diagrams using hypergraphs to the case where the underlying category has a *traced comonoid structure*, in which wires can be forked and the outputs of a morphism can be connected to its input. Such a structure is particularly interesting because any traced Cartesian (dataflow) category has an underlying traced comonoid structure. We show that certain subclasses of hypergraphs are fully complete for traced comonoid categories: that is to say, every term in such a category has a unique corresponding hypergraph up to isomorphism, and from every hypergraph with the desired properties, a unique term in the category can be retrieved up to the axioms of traced comonoid categories. We also show how the framework of double pushout rewriting (DPO) can be adapted for traced comonoid categories by characterising the valid pushout complements for rewriting in our setting. We conclude by presenting a case study in the form of recent work on an equational theory for *sequential circuits*: circuits built from primitive logic gates with delay and feedback. The graph rewriting framework allows for the definition of an *operational semantics* for sequential circuits.

1. INTRODUCTION

String diagrams constitute a useful and elegant conceptual bridge between term rewriting and graph rewriting. Since their introduction in the 90s [JS91, JSV96], their use has exploded recently both for use in diverse fields such as cyclic lambda calculi [Has97], fix-point operators [Has03], quantum protocols [AC04], signal flow diagrams [BSZ14, BSZ15], linear algebra [BSZ17, Zan15, BPSZ19, BP22], finite state automata [PZ21], dynamical systems [BE15, FSR16], electrical and electronic circuits [BS22, GKS24], and automatic differentiation [AGSZ23]. Although not the first use of graphical notation by any means [Pen71], string diagrams are notable because they are *more* than just a visual aid; they are a sound and complete representation of categorical terms in their own right [JS91, Kis14]. This means one can reason solely in string diagrams without fear that they may somehow be working with malformed terms. String diagrams have been adapted to accommodate all sorts of categorical structure; the survey [Sel11] is a suitable entry point to the literature.

While string diagrams have proven to be immensely useful for equational reasoning with terms in symmetric monoidal categories, they still have their shortcomings in that they are difficult for computers to manipulate compared to combinatorial graph-based structures. This can be remedied by mapping string diagram terms into certain categories of graphs

Key words and phrases: symmetric traced monoidal categories, string diagrams, graph rewriting, comonoid structure, double pushout rewriting .

* This is a revised and extended version of [GK23].

and performing *double pushout (DPO) rewriting* [EK76], a categorical framework in which rewrites are performed by using morphisms between graphs. Recent work on this subject has used *framed point graphs* [Kis12, DK13] and more recently *hypergraphs* [BGK⁺22a, BGK⁺22b, BGK⁺22c].

Vanilla string diagram terms with no extra structure are fairly basic: all wires are *progressive* in that they are always flowing in one direction across the page. Put simply, the output of a box can only ever connect to the input of another box. To model more complicated processes, much work on string diagram rewriting has considered string diagrams in the presence of extra structure. A common setting involves the addition of a comonoid structure ('fork') and a monoid structure ('join') along with special equations to yield what is known as a *Frobenius structure*; among other properties Frobenius terms form a *compact closed category*. One effect of this structure is that wires in Frobenius string diagrams are 'bidirectional': one can connect a wire from the output of a box to another output of a box. This can be contrasted with settings equipped with a *trace*, in which wires can be bent backwards but must be unbent before reaching their endpoints; much like the ordinary symmetric monoidal setting, outputs can only ever be connected to inputs.

This paper is concerned with bringing string diagram rewriting techniques to a particular combination of the structures detailed above: *traced categories with a comonoid structure* but without a monoid structure. This is motivated by a particular class of categories known as *dataflow categories* [C§90, C§94, Has97], in which the comonoid structure is the Cartesian product; one application of string diagrammatic reasoning that exhibits a dataflow structure is the semantics of digital circuits [GKS24]. The gap between the kind of semantic models which use an underlying compact closed structure and those which use a traced monoidal structure is significant: the former have a *relational* nature with subtle causality (e.g. quantum or electrical circuits) whereas the latter are *functional* with clear input-output causality (e.g. digital or logical circuits), so it is not surprising that the underlying rewrite frameworks should differ.

Reasoning with these categories is technically challenging, as it falls in a gap between compact closed structures constructible via Frobenius and symmetric monoidal categories without trace. For example, it is well known that if the Cartesian product exists in a compact closed category then it is degenerate and identified with the coproduct. Even without invoking copying, we will see how trying to perform rewriting in a traced category with a comonoid structure can also lead to inconsistencies. This is a firm indication that a bespoke rewriting framework needs to be constructed to fill this particular situation.

Traced categories were considered by the aforementioned work on framed point graphs, but this requires rewriting modulo so-called *wire homeomorphisms*. This style of rewriting is awkward and is increasingly considered as obsolete as compared to the more recent work on rewriting with hypergraphs; our work seeks to extend the latter work to apply to the traced comonoid case, by building on the results on hypergraph string diagram rewriting modulo Frobenius structure [BGK⁺22a], symmetric monoidal structure [BGK⁺22b], and settings equipped with a monoid structure [FL23, MPZ23].

Contributions. This paper makes two distinct technical contributions. The first is to show that one subclass of cospans of hypergraphs ('partial monogamous') is fully complete for traced terms (Corollary 3.32), and another class ('partial left-monogamous') is fully complete for traced comonoid terms (Theorem 4.14). What this means is that every string diagram in a traced setting corresponds to a unique partially monogamous cospan of hypergraphs up

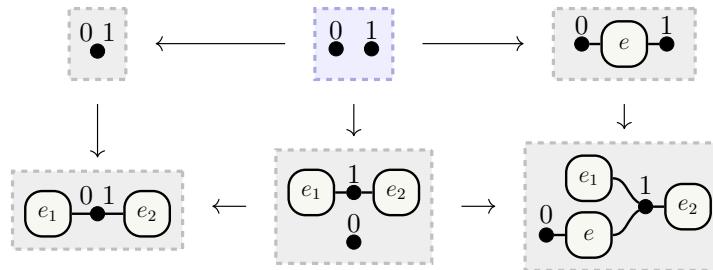
to isomorphism, and every partially monogamous cospan of hypergraphs corresponds to a unique string diagram, such that the process of mapping between the two interpretations are inverses.

The challenge in this step is not so much in proving the correctness of the construction but in defining precisely what these combinatorial structures should be. In particular, the extremal point of tracing the identity: $\text{Tr}(\square) = \square$, corresponding graphically to a closed loop, provides a litmus test. The way this is resolved must be robust enough to handle the addition of the comonoid structure, in which one can ‘trace a forking wire’:

$$\text{Tr}(\square) = \text{forking wire}.$$

The second contribution is concerned with the well-definedness of graph rewriting with partial monogamous and partial left-monogamous cospans of hypergraphs when using DPO rewriting. A graph rewrite in DPO rewriting is fully determined by the choice of a *pushout complement* for a rewrite rule and instance of said rule in a larger graph; the pushout complement is the context of a rewrite step. For a given rule and graph, there may be multiple such pushout complements, but not all of these may represent a valid rewrite in a given string diagram setting. When rewriting with Frobenius structure, every pushout complement is valid [BGK⁺22a] whereas when rewriting with symmetric monoidal structure exactly one pushout complement is valid [BGK⁺22b]; for the traced case some pushout complements are valid and some are not. Our contribution here is to characterise the valid pushout complements as ‘traced boundary complements’ (Definition 5.16) for the traced setting and as ‘traced left-boundary complements’ (Definition 5.25) for the traced comonoid setting. Subsequently we show that DPO rewriting using traced boundary complements is well-defined for cospans of partial monogamous cospans of hypergraphs (Theorem 5.24), and DPO rewriting using traced left-boundary complements is well-defined for cospans of partial left-monogamous cospans of hypergraphs (Theorem 5.29).

This is best illustrated with an example in which there is a pushout complement that is valid in a Frobenius setting because it uses the monoid structure, but it is not valid neither in a traced, nor even in a traced comonoid setting. Imagine we have a rule $\langle \square, -[e]- \rangle$ and a term $[e_1] - [e_2]$, and rewrite it as follows.



This corresponds to the term rewrite $[e_1] - [e_2] = \text{forking wire} - [e_2] = \text{forking wire} - e$, which holds

in a Frobenius setting but not a setting without a commutative monoid structure. On the other hand, the rewriting system for symmetric monoidal categories [BGK⁺22b] is too restrictive as it enforces that any matching must be mono: this prevents matchings such as

 in . Here again the challenge is precisely identifying the concept of traced boundary complement mathematically. The solution, although not immediately obvious, is not complicated, again requiring a generalisation from monogamy to partial monogamy and partial left-monogamy.

Towards the end of this paper, we provide two extended case studies on how string diagram rewriting modulo traced comonoid structure can be applied. The first is a discussion on generic rewriting in settings where the comonoid structure is a Cartesian product; the second is an overview of how graph rewriting can be applied to the categorical theory of digital circuits presented in [GKS24], resulting in an automated operational semantics for sequential circuits.

This paper is an extended version of a paper of the same title published in the proceedings of FSCD 2023 [GK23]. The results presented are the same, but this version contains a refined narrative, full proofs for all results, and expanded examples.

2. MONOIDAL THEORIES AND HYPERGRAPHS

When modelling a system using monoidal categories, its components and properties are specified using a *monoidal theory*. A class of SMCs particularly interesting to us is that of *PROPs* ('categories of *PRO*ducts and *P*ermutations') [Mac65, Chap. V.24], which have natural numbers as objects and addition as tensor product on objects.

Definition 2.1 (Symmetric monoidal theory). A (*single-sorted*) *symmetric monoidal theory* (SMT) is a tuple (Σ, \mathcal{E}) where Σ is a set of *generators* in which each generator $\phi \in \Sigma$ has an associated *arity* $\text{dom}(\phi) \in \mathbb{N}$ and *coarity* $\text{cod}(\phi) \in \mathbb{N}$, and \mathcal{E} is a set of equations. Given a SMT (Σ, \mathcal{E}) , let \mathbf{S}_Σ be the strict symmetric monoidal category freely generated over Σ and let $\mathbf{S}_{\Sigma, \mathcal{E}}$ be \mathbf{S}_Σ quotiented by the equations in \mathcal{E} .

Remark 2.2. One can also define a *multi-sorted* SMT, in which wires can be of multiple colours. In this paper we will only consider the single-sorted case, but the results generalise easily using the results of [BGK⁺22a, BGK⁺22b].

While one could reason in \mathbf{S}_Σ using the one-dimensional categorical term language, it is more intuitive to reason with *string diagrams* [JS91, Sel11], which represent *equivalence classes* of terms up to the axioms of SMCs. In the language of string diagrams, a generator $\phi: m \rightarrow n$ is drawn as a box $m \dashv \phi \dashv n$, the identity id_n as $n \dashv \square \dashv n$, and the symmetry $\sigma_{m,n}$ as $\begin{smallmatrix} m & \\ n & \times \\ m & \end{smallmatrix}$. Composite terms will be illustrated as wider boxes $m \dashv [f] \dashv n$ to distinguish them from generators. Diagrammatic order composition $m \dashv [f] \dashv n ; n \dashv [g] \dashv p$ is defined as horizontal juxtaposition $m \dashv [f] \dashv [g] \dashv p$ and tensor $m \dashv [f] \dashv n \otimes p \dashv [g] \dashv q$ as vertical juxtaposition $\begin{array}{c} m \dashv [f] \dashv n \\ p \dashv [g] \dashv q \end{array}$.

Remark 2.3. String diagrams are not restricted to PROPs, and can be used to provide a graphical notation for any flavour of symmetric monoidal category. In this paper we will stick to the PROP case, motivated by our use of monoidal theories containing generators with natural numbers as their domain and codomain.

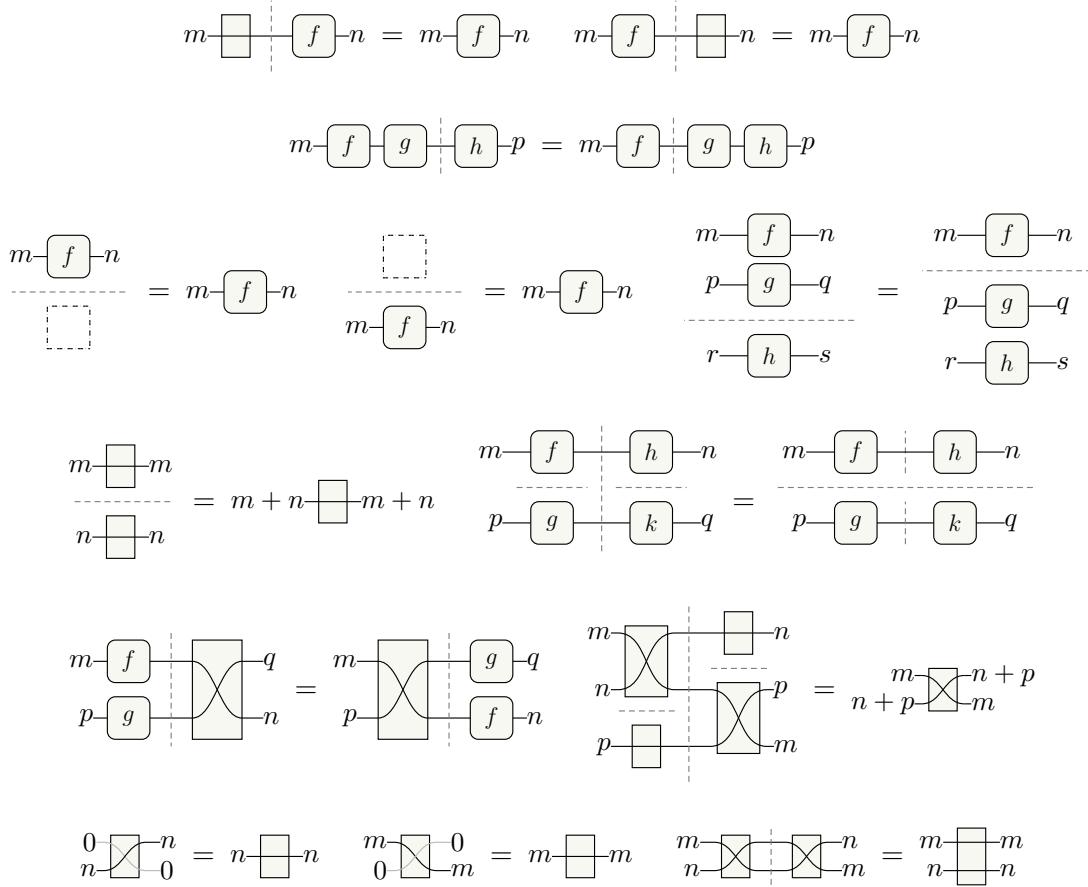


Figure 1: The equations of symmetric monoidal categories, expressed string diagrammatically in the language of PROPs

Remark 2.4. Although we have drawn a generator $m-\boxed{\phi}-n$ as a box with a single wire as the input and output, this is actually syntactic sugar for drawing m and n individual wires respectively; this avoids cluttering up diagrams with lots of parallel wires. A way of turning this syntactic sugar into a formal syntactic construct is discussed in [WGZ23].

On a related note, we may omit the labels from wires when clear from context; in the absence of such context, a wire with no label can be taken to mean a wire for the object 1.

The power of string diagrams comes from how they ‘absorb’ the equations of SMCs, as shown in Figure 1. This is not merely a convenient notation; string diagrams for symmetric monoidal categories are a mathematically rigorous language in their own right.

Theorem 2.5 [JS91, Thm. 2.3]. *Given two terms $f, g \in \mathbf{S}_\Sigma$, $f = g$ by axioms of SMCs if and only if their string diagrams are isomorphic.*

String diagrams clearly illustrate the differences between the *syntactic* category \mathbf{S}_Σ and the *semantic* category $\mathbf{S}_{\Sigma, \mathcal{E}}$. In the former, only ‘structural’ equalities of the axioms of SMCs hold: moving boxes around while retaining connectivity. In the latter, more equations hold so terms with completely different boxes and connectivity can be equal.

(M1) = —
(M3) =
(M4) =

Figure 2: Equations $\mathcal{E}_{\text{CMon}}$ of a *commutative monoid*

(C1) = —
(C3) =
(C4) =

Figure 3: Equations $\mathcal{E}_{\text{CComon}}$ of a *commutative comonoid*

(F1) =
(F2) =
(F3) = —

Figure 4: Equations $\mathcal{E}_{\text{Frob}}$ of a *special commutative Frobenius algebra*, in addition to those in Figures 2 and 3

Example 2.6. The monoidal theory of *special commutative Frobenius algebras* is defined as $(\Sigma_{\text{Frob}}, \mathcal{E}_{\text{Frob}})$ where $\Sigma_{\text{Frob}} := \{ \square \bullet, \bullet \square, \square \square, \square \square \}$ and the equations of $\mathcal{E}_{\text{Frob}}$ are listed in Figures 2, 3, and 4. We write $\mathbf{Frob} := \mathbf{S}_{\Sigma_{\text{Frob}}, \mathcal{E}_{\text{Frob}}}$.

Terms in \mathbf{Frob} are all the ways of composing the generators of Σ_{Frob} in sequence and parallel. For example, the following are all terms in \mathbf{Frob} :

Using the equations of $\mathcal{E}_{\text{Frob}}$, we can show that the latter two terms are equal in \mathbf{Frob} :

M1 ≡
F1 ≡
M3 ≡

Since the diagrammatic notation takes care of the axioms of SMCs, we only need to worry about the equations of the monoidal theory.

2.1. Categories of hypergraphs. Reasoning equationally using string diagrams is certainly attractive as a pen-and-paper method, but for larger systems it quickly becomes intractable to do this by hand. Instead, it is desirable to perform equational reasoning *computationally*. Unfortunately, string diagrams as topological objects are not particularly suited for this purpose; it is more suitable to use a combinatorial representation. Fortunately, this has been well studied recently, first with *string graphs* [DK13, Kis12] and later with *hypergraphs* [BGK⁺22a, BGK⁺22b, BGK⁺22c], a generalisation of regular graphs in which edges can be the source or target of an arbitrary number of nodes. In this paper we are concerned with the latter.

Hypergraphs are formally defined as objects in a functor category.

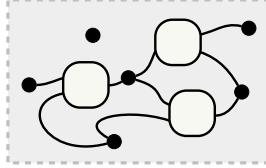
Definition 2.7 (Hypergraph). Let \mathbf{X} be the category containing objects (k, l) for $k, l \in \mathbb{N}$ and one additional object \star . For each (k, l) there are $k + l$ morphisms $(k, l) \rightarrow \star$. Let \mathbf{Hyp} be the functor category $[\mathbf{X}, \mathbf{Set}]$.

An object in **Hyp** maps \star to a set of nodes, and each pair (k, l) to a set of hyperedges with k sources and l targets. Given a hypergraph $F \in \mathbf{Hyp}$, we write F_\star for its set of vertices and $F_{k,l}$ for the set of edges with k sources and l targets. Subsequently each functor induces functions $s_i: F_{k,l} \rightarrow F_\star$ for $i < k$ and $t_j: F_{k,l} \rightarrow F_\star$ for $j < l$.

Example 2.8. We define the hypergraph F as follows:

$$\begin{aligned} F_\star &:= \{v_0, v_1, v_2, v_3, v_4, v_5\} & F_{2,1} &:= \{e_0, e_2\} & F_{1,2} &:= \{e_1\} \\ s_0(e_0) &:= v_0 & s_1(e_0) &:= v_2 & s_0(e_1) &:= v_3 & s_0(e_2) &:= v_3 & s_1(e_2) &:= v_2 \\ t_0(e_0) &:= v_3 & t_0(e_1) &:= v_5 & t_1(e_1) &:= v_4 & t_0(e_2) &:= v_4 \end{aligned}$$

Much like regular graphs, it is easier to comprehend hypergraphs graphically. nodes are drawn as black dots and hyperedges as boxes. Tentacles from edges to nodes identify the (ordered) sources and targets. The hypergraph F is drawn as follows:



Note that the nodes themselves do not have any notion of ordering or directionality.

The graphical notation for hypergraphs already looks very similar to that of string diagrams. However, the boxes are lacking labels for generators in a signature; to remedy this we shall translate signatures themselves into hypergraphs.

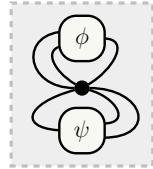
Definition 2.9 (Hypergraph signature [BGK⁺22a, Sec. 3.1]). For a given monoidal signature Σ , its corresponding *hypergraph signature* $\llbracket \Sigma \rrbracket$ is the hypergraph with a single node v and edges $e_\phi \in \llbracket \Sigma \rrbracket_{\text{dom}(\phi), \text{cod}(\phi)}$ for each $\phi \in \Sigma$. For a hyperedge e_ϕ , $i < \text{dom}(\phi)$ and $j < \text{cod}(\phi)$, $s_i(e_\phi) = t_j(e_\phi) = v$.

A labelling is then a morphism from a hypergraph to a signature. A morphism of hypergraphs $f: F \rightarrow G \in \mathbf{Hyp}$ consists of functions f_\star and $f_{k,l}$ for each $k, l \in \mathbb{N}$ preserving sources and targets in the obvious way.

Example 2.10. Let $\Sigma = \{\phi: 2 \rightarrow 1, \psi: 1 \rightarrow 2\}$ be a monoidal signature. The corresponding monoidal signature $\llbracket \Sigma \rrbracket$ is

$$\begin{aligned} \llbracket \Sigma \rrbracket_\star &:= \{v_0\} & \llbracket \Sigma \rrbracket_{2,1} &:= \{e_\phi\} & \llbracket \Sigma \rrbracket_{1,2} &:= \{e_\psi\} \\ s_0(e_\phi) &:= v_0 & s_1(e_\phi) &:= v_0 & s_0(e_\psi) &:= v_0 & t_0(e_\phi) &:= v_0 & t_0(e_\psi) &:= v_0 & t_1(e_\psi) &:= v_0 \end{aligned}$$

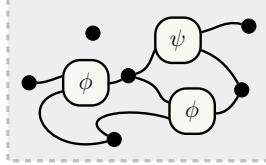
and is drawn as follows:



Recall the hypergraph F from Example 2.8; one labelling $\Gamma: F \rightarrow \llbracket \Sigma \rrbracket$ could be defined as

$$\Gamma_\star(-) := v_0 \quad \Gamma_{2,1}(e_0) := e_\phi \quad \Gamma_{1,2}(e_1) := e_\psi \quad \Gamma_{2,1}(e_2) := e_\phi$$

We simply call the morphism Γ a *labelled hypergraph* and draw it in the same manner as a regular hypergraph but with labelled edges.



Note that if there are multiple generators with the same arity and coarity in a signature, there may well be multiple valid labellings of a hypergraph.

A category of labelled hypergraphs is defined using another piece of categorical machinery.

Definition 2.11 (Slice category [Law63, pg. 36]). For a category \mathbf{C} and an object $C \in \mathbf{C}$, the *slice category* \mathbf{C}/C has objects the morphisms of \mathbf{C} with target C , and has a morphism $(f: X \rightarrow C) \rightarrow (f': X' \rightarrow C)$ if there is a morphism $g: X \rightarrow X' \in \mathbf{C}$ such that $f' \circ g = f$.

Definition 2.12 (Labelled hypergraph [BGK⁺22a, Sec. 3.1]). For a monoidal signature Σ , let the category \mathbf{Hyp}_Σ be defined as the slice category $\mathbf{Hyp}/[\![\Sigma]\!]$.

There is another difference between hypergraphs and string diagrams. While (labelled) hypergraphs may have dangling nodes, they do not have *interfaces* specifying the order of inputs and outputs. These can be provided using *cospans*.

Definition 2.13 [BGK⁺22a, Def. 2.10]. For a category \mathbf{C} with finite colimits, a *cospans* from $X \rightarrow Y$ is a pair of arrows $X \rightarrow A \leftarrow Y$. A *cospans morphism* $(X \xrightarrow{f} A \leftarrow Y) \rightarrow (X \xrightarrow{h} B \leftarrow Y)$ is a morphism $\alpha: A \rightarrow B \in \mathbf{C}$ such that the following diagram commutes:

$$\begin{array}{ccccc} & f & \nearrow & \swarrow & g \\ & A & & & \\ X & \downarrow \alpha & & & Y \\ & h & \searrow & \swarrow & k \\ & B & & & \end{array}$$

Two cospans $X \rightarrow A \leftarrow Y$ and $X \rightarrow B \leftarrow Y$ are *isomorphic* if there exists a morphism of cospans as above where α is an isomorphism. Composition is by pushout:

$$\begin{array}{ccccc} & & D & & \\ & & \nearrow & \swarrow & \\ & A & & B & \\ f \nearrow & \downarrow g & \nearrow h & \swarrow k & \\ X & & Y & & Z \end{array}$$

The identity is $X \xrightarrow{\text{id}_X} X \xleftarrow{\text{id}_X} X$. The category of cospans over \mathbf{C} , denoted $\mathbf{Csp}(\mathbf{C})$, has as objects the objects of \mathbf{C} and as morphisms the isomorphism classes of cospans. This category has monoidal product given by the coproduct in \mathbf{C} and has monoidal unit given by the initial object $0 \in \mathbf{C}$.

The interfaces of a hypergraph can be specified as cospans by having the ‘legs’ of the cospan pick nodes in the graph at the apex.

Definition 2.14 (Discrete hypergraph). A hypergraph is called *discrete* if it has no edges.

A discrete hypergraph F with $|F_\star| = n$ is written as n when clear from context. Morphisms from discrete hypergraphs to another hypergraph pick out the nodes in the ‘inputs’ and ‘outputs’ of the latter. To establish a category of cospans in which the legs are discrete hypergraphs, and moreover to establish an *ordering* on these discrete hypergraphs, we move from using plain cospans of hypergraphs to cospans in which the legs are in the image of some functor.

Theorem 2.15 [BGK⁺22a, Thm. 3.6]. *Let \mathbb{X} be a PROP whose monoidal product is a coproduct, \mathbf{C} a category with finite colimits, and $F: \mathbb{X} \rightarrow \mathbf{C}$ a coproduct-preserving functor. Then there exists a PROP $\mathsf{Csp}_F(\mathbf{C})$ whose arrows $m \rightarrow n$ are isomorphism classes of \mathbf{C} cospans $Fm \rightarrow C \leftarrow Fn$.*

Proof. Composition is by pushout. For cospans $Fm \rightarrow C \leftarrow Fn$ and $Fp \rightarrow C \leftarrow Fq$, their coproduct is given by $Fm + Fp \rightarrow C + D \leftarrow Fn + Fq$: $F(m+p) \cong Fm + Fp$ and $F(n+q) \cong Fn + Fq$ because F preserves coproducts. Symmetries in \mathbb{X} are determined by the universal property of the coproduct; they are inherited by $\mathsf{Csp}_F(\mathbf{C})$ because F preserves coproducts. \square

Theorem 2.16 [BGK⁺22a, Thm. 3.8]. *Let \mathbb{X} be a PROP whose monoidal product is a coproduct, \mathbf{C} a category with finite colimits, and $F: \mathbb{X} \rightarrow \mathbf{C}$ a colimit-preserving functor. Then there is a homomorphism of PROPs $\tilde{F}: \mathsf{Csp}(\mathbb{X}) \rightarrow \mathsf{Csp}_F(\mathbf{C})$ that sends $m \xrightarrow{f} X \xleftarrow{g} n$ to $Fm \xrightarrow{Ff} FX \xleftarrow{Fg} Fn$. If F is full and faithful, then \tilde{F} is faithful.*

Proof. Since F preserves finite colimits, it preserves composition (pushout) and monoidal product (coproduct); symmetries are clearly preserved. To show that \tilde{F} is faithful when F is full and faithful, suppose that $\tilde{F}(m \xrightarrow{f} X \xleftarrow{g} n) = \tilde{F}(m \xrightarrow{f'} X \xleftarrow{g'} n)$. This gives us the following commutative diagram in \mathbf{C} :

$$\begin{array}{ccccc} & & FX & & \\ & Ff \swarrow & \downarrow \phi & \searrow Fg & \\ Fm & & \downarrow & & Fn \\ & Ff' \searrow & & \swarrow Fg' & \\ & & FY & & \end{array}$$

where ϕ is an isomorphism as objects in $\mathsf{Csp}_F(\mathbf{C})$ are isomorphism classes of cospans. As F is full, there exists $\psi: X \rightarrow Y$ such that $F\psi = \phi$. As F is faithful, ψ is an isomorphism; this means $m \xrightarrow{f} X \xleftarrow{g} n$ and $m \xrightarrow{f'} X \xleftarrow{g'} n$ are equal in $\mathsf{Csp}(\mathbb{X})$, so \tilde{F} is faithful. \square

In our setting, this functor will be from the category of finite sets.

Definition 2.17. Let **FinSet** be the PROP with morphisms $m \rightarrow n$ the functions between finite sets $[m] \rightarrow [n]$.

Definition 2.18 [BGK⁺22a, Sec. 3.2]. Let $D: \mathbf{FinSet} \rightarrow \mathbf{Hyp}_\Sigma$ be a faithful, coproduct-preserving functor that sends each object $[n] \in \mathbf{FinSet}$ to the discrete hypergraph $n \in \mathbf{Hyp}_\Sigma$ and each morphism to the induced homomorphism of discrete hypergraphs.

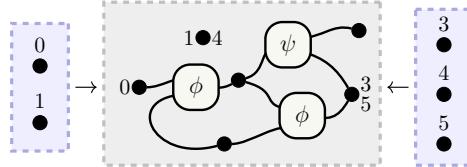
D (for ‘discrete’) maps a finite set $[n] := \{0, 1, \dots, n-1\}$ to the discrete hypergraph with n nodes. The functor induces an underlying function between sets $[n] \rightarrow n_\star$; this establishes an ordering on the interfaces.

By combining the above theorems and definitions we obtain the category $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ with objects *discrete cospans of hypergraphs*. Since the legs of each cospan are discrete hypergraphs with n nodes, this is another PROP.

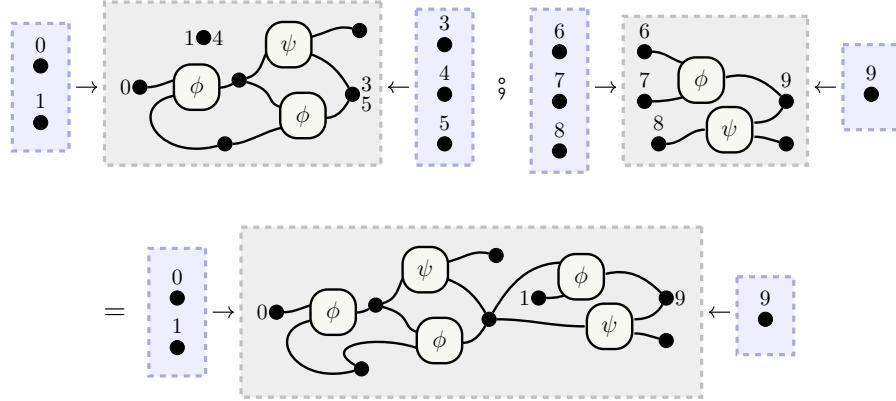
Example 2.19. Recall the labelled hypergraph F from Example 2.10. We assign interfaces to it as the cospan $3 \xrightarrow{f} F \xleftarrow{g} 3$, where

$$f(D0) = v_0 \quad f(D1) = v_1 \quad g(D0) = v_4 \quad g(D1) = v_1 \quad g(D2) = v_4$$

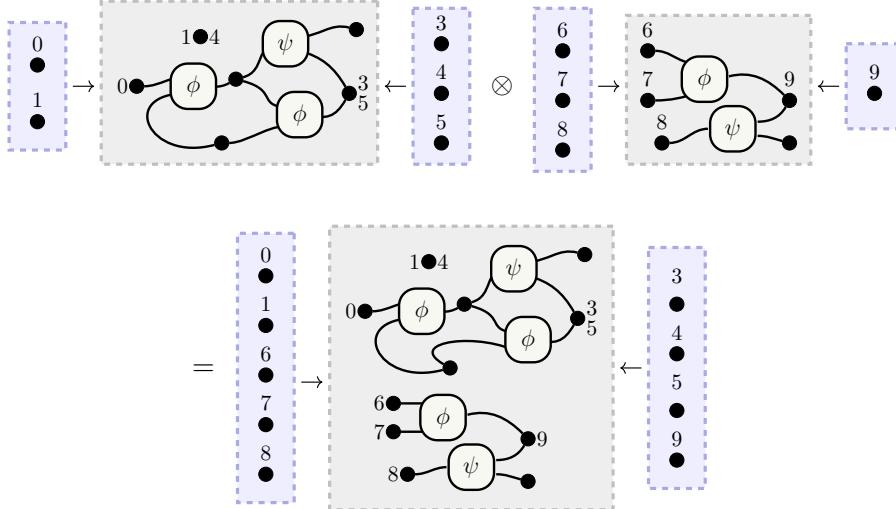
Interfaces of the hypergraph F are drawn to its left and right, with numbers illustrating the action of the cospan maps. For clarity we number the outputs after the inputs.



Composition in $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is by pushout; effectively the nodes in the output of the first cospan are ‘glued together’ with the inputs of the second.



Tensor in $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is by direct product; putting cospans on top of each other.



$$\begin{array}{c}
 x+y \bullet x+y = \begin{array}{c} x \\ y \\ x \\ y \end{array} \quad x+y \bullet x+y = \begin{array}{c} x \\ y \\ x \\ y \end{array} \\
 \bullet x+y = \begin{array}{c} \bullet x \\ \bullet y \end{array} \quad x+y \bullet = \begin{array}{c} x \bullet \\ y \bullet \end{array}
 \end{array}$$

Figure 5: Equations \mathcal{E}_{Hyp} of a *hypergraph category*, as well as those in Figures 2, 3, and 4.

2.2. Frobenius terms as hypergraphs. The main result of [BGK⁺22a] is that the category of ‘Frobenius terms’ $\mathbf{S}_\Sigma + \mathbf{Frob}$ is in correspondence with $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$: every morphism in the former (modulo the Frobenius equations in Figures 2, 3, and 4) corresponds to exactly one isomorphism class of cospans in $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$, and vice versa. To show this, we make use of another type of category known as a *hypergraph category*

Definition 2.20 (Hypergraph category [FS19, Def. 2.12]). A *hypergraph category* is a symmetric monoidal category in which each object X has a special commutative Frobenius structure in the sense of Example 2.6 satisfying the equations in Figure 5.

The prototypical hypergraph category over a signature is the freely generated symmetric monoidal category \mathbf{S}_Σ augmented with the Frobenius generators and equations in **Frob**.

Corollary 2.21. $\mathbf{S}_\Sigma + \mathbf{Frob}$ is a hypergraph category.

The term ‘hypergraph category’ should not be confused with the category of hypergraphs **Hyp**, which is not itself a hypergraph category. However, the category of *cospans* of hypergraphs is such a category.

Proposition 2.22. $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is a hypergraph category.

Proof. A Frobenius structure can be defined on $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ for each $n \in \mathbb{N}$ as follows:

$$\begin{array}{ll}
 n \bullet n := n + n \rightarrow n \leftarrow n & \bullet n := 0 \rightarrow n \leftarrow n \\
 n \square n := n \rightarrow n \leftarrow n + n & n \square \bullet := n \rightarrow n \leftarrow 0
 \end{array}
 \quad \square$$

We make use of the components of the above definition in order to define a PROP morphism from $\mathbf{S}_\Sigma + \mathbf{Frob}$ into $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$. Since \mathbf{S}_Σ is freely generated, these PROP morphisms can be defined solely on generators.

Definition 2.23 [BGK⁺22a, Sec. 3.2]. Let $\llbracket - \rrbracket_\Sigma : \mathbf{S}_\Sigma \rightarrow \mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ be a PROP morphism

defined on generators as $\llbracket m \square \phi \square n \rrbracket_\Sigma := m \rightarrow \boxed{\begin{array}{c} \vdots m \vdots \phi \vdots n \vdots \\ \bullet \quad \bullet \end{array}} \leftarrow n$.

Definition 2.24. Let $[-]_\Sigma : \mathbf{Frob} \rightarrow \mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ be a PROP morphism defined as in Proposition 2.22.

Definition 2.25. Let the PROP morphism $\langle\langle - \rangle\rangle_\Sigma : \mathbf{S}_\Sigma + \mathbf{Frob} \rightarrow \mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ be the copairing of $\llbracket - \rrbracket_\Sigma$ and $[-]_\Sigma$.

The PROP morphism $\langle\langle - \rangle\rangle_\Sigma$ maps Frobenius terms to cospans of hypergraphs. To show that $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is suitable for reasoning about Frobenius terms, this mapping must be full

and faithful; i.e. it must be a one-to-one mapping between Frobenius terms and isomorphism classes of cospans.

Theorem 2.26 [BGK⁺22a, Thm. 4.1]. *There is an isomorphism of PROPs $\mathbf{S}_\Sigma + \mathbf{Frob} \cong \mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$.*

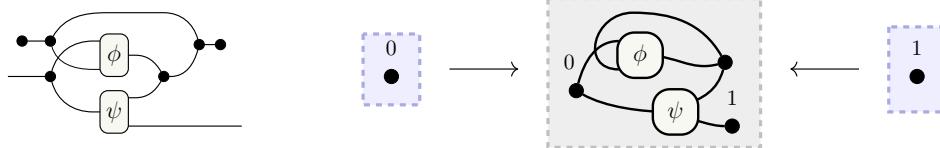
Proof (Sketch). Since $\mathbf{S}_\Sigma + \mathbf{Frob}$ is a coproduct in the category of PROPs, this can be shown by proving that $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ satisfies the universal property of the coproduct: given a coloured PROP \mathbb{A} and PROP morphisms $\mathbf{S}_\Sigma \rightarrow \mathbb{A}$ and $\mathbf{Frob} \rightarrow \mathbb{A}$, there exists a unique morphism $u: \mathbf{Csp}_D(\mathbf{Hyp}_\Sigma) \rightarrow \mathbb{A}$ as below:

$$\begin{array}{ccccc} \mathbf{S}_{\Sigma,C} & \xrightarrow{\llbracket - \rrbracket_\Sigma} & \mathbf{Csp}_D(\mathbf{Hyp}_\Sigma) & \xleftarrow{\llbracket - \rrbracket_\Sigma} & \mathbf{Frob} \\ & \searrow f & \downarrow u & \swarrow g & \\ & & \mathbb{A} & & \end{array}$$

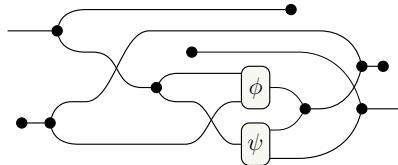
All the PROP morphisms involved are identity-on-objects, so all that is required to show the existence of u is to show that any morphism in $\mathbf{Csp}_{D_C}(\mathbf{Hyp}_{C,\Sigma})$ can be expressed as a composition of components either in the image of $\llbracket - \rrbracket_\Sigma$ or $\llbracket - \rrbracket_\Sigma$. \square

Rather than giving the formal construction required for the above proof, it is more instructive to provide a concrete example.

Example 2.27. Consider the following term and its cospan interpretation:



This cospan can be assembled into the form shown in Figure 6; by following the vertex maps, one can verify that this is indeed isomorphic to the original cospan. The outermost components correspond to terms in \mathbf{Frob} and the innermost to a term in \mathbf{S}_Σ .



This term is equal to the original term by the Frobenius equations.

This result means that any two terms in $\mathbf{S}_\Sigma + \mathbf{Frob}$ which are equal by the Frobenius equations can be mapped to isomorphic cospans of hypergraphs.

Example 2.28. Recall the following terms in \mathbf{Frob} from Example 2.6, which we showed were equal by the Frobenius equations.

$$\begin{array}{c} \text{Diagram 1: } \text{Hypergraph term} \\ = \\ \text{Diagram 2: } \text{Hypergraph term} \end{array}$$

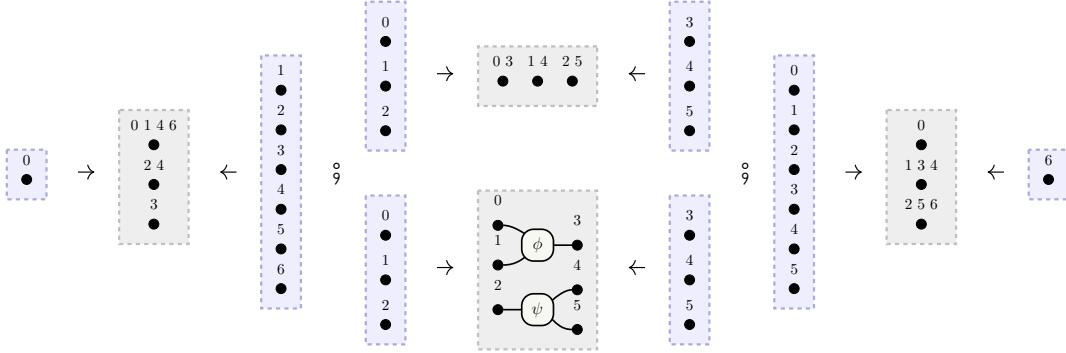
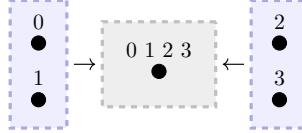


Figure 6: The cospan of Example 2.27 in the form of Theorem 2.26

By the isomorphism of Theorem 2.26, these two terms map to the same cospan of hypergraphs:



All of the Frobenius structure collapses into one vertex, much like when we considered the correspondence between Frobenius terms and finite sets.

3. HYPERGRAPHS FOR TRACED CATEGORIES

In the previous section we summarised the results of [BGK⁺22a] showing that every cospan of hypergraphs in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ corresponds to a single Frobenius term. In this paper we are concerned with *traced* terms, a more restrictive setting: we lack the ability to fork and join wires, and enforce that outputs of boxes may only be connected to inputs of boxes.

Definition 3.1 (Symmetric traced monoidal category [JSV96, Sec. 2]). A *symmetric traced monoidal category* (STMC) is a symmetric monoidal category \mathbf{C} equipped with a family of functions $\text{Tr}_{A,B}^X(-): \mathbf{C}(X \otimes A, X \otimes B) \rightarrow \mathbf{C}(A, B)$ for any objects A, B and X satisfying the axioms of STMCs:

$$\begin{aligned}
 \text{id}_X \otimes f ; g ; \text{id}_X \otimes h &= f ; \text{Tr}_{B,C}^X(g) && \text{(Tightening)} \\
 \text{Tr}_{A,B}^X(f ; g \otimes \text{id}_B) &= \text{Tr}_{A,B}^Y(g \otimes \text{id}_A ; f) && \text{(Sliding)} \\
 \text{Tr}_{A,B}^X(\text{Tr}_{X \otimes A, X \otimes B}^Y(f)) &= \text{Tr}_{A,B}^{X \otimes Y}(f) && \text{(Vanishing)} \\
 \text{Tr}_{A,B}^X(f) \otimes g &= \text{Tr}_{A \otimes C, B \otimes D}^X(f \otimes g) && \text{(Superposing)} \\
 \text{Tr}_{A,B}^Y(g \otimes \text{id}_A ; f) &= \text{id}_X && \text{(Yanking)}
 \end{aligned}$$

The trace is represented string diagrammatically by joining output wires to input wires:

$$\text{Tr}_{m,n}^x \left(\begin{smallmatrix} x & - & f & - & x \\ m & - & \boxed{f} & - & n \end{smallmatrix} \right) \stackrel{\text{def}}{=} \begin{smallmatrix} & \curvearrowleft & \\ & f & \\ & \curvearrowright & \end{smallmatrix}$$

When drawn in this manner the equations of STMCs can be elegantly expressed in the language of PROPs as shown in Figure 7. Much like regular STMCs, this notation is sound and complete.

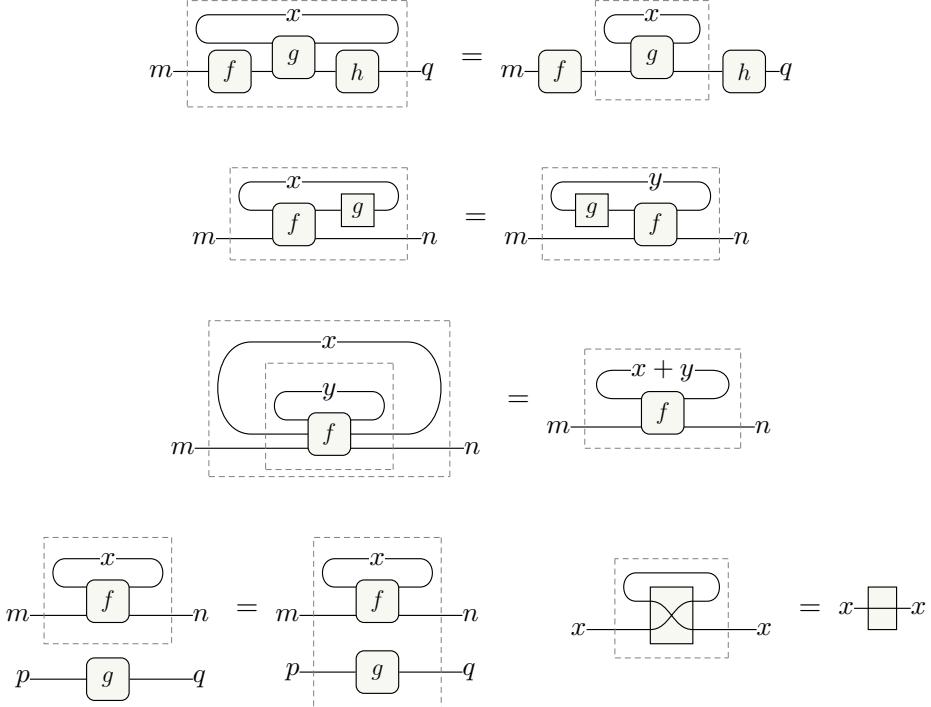


Figure 7: Equations that hold in any *symmetric traced monoidal category*, expressed string diagrammatically in the language of PROPs

$$\begin{array}{ccc} x \curvearrowright x & = & x - x \\ & & x \curveleftarrow x & = & x - x \end{array}$$

Figure 8: Equations that hold in any *compact closed category*, expressed string diagrammatically in the language of PROPs

Definition 3.2. For a monoidal signature Σ , let \mathbf{T}_Σ be the STMC freely generated over Σ . For a set of equations \mathcal{E} , let $\mathbf{T}_{\Sigma, \mathcal{E}}$ be defined as $\mathbf{T}_\Sigma / \mathcal{E}$.

Theorem 3.3 [Kis14, Cor. 6.14]. *Given two terms $f, g \in \mathbf{T}_\Sigma$, $f = g$ by axioms of STMCs if and only if their string diagrams are isomorphic.*

3.1. Compact closed categories. For $\text{Csp}_D(\text{Hyp}_\Sigma)$ to be suitable for reasoning about traced terms, it must necessarily have a trace. Fortunately, the links between the trace and Frobenius structure is very well-known, and can be expressed using the notion of another type of category.

Definition 3.4 (Compact closed category [KL80, Sec. 1]). A *compact closed category* (CCC) is a symmetric monoidal category in which every object X has a *dual* X^* equipped with morphisms called the *unit* $\eta: I \rightarrow X^* \otimes X$ and the *counit* $\varepsilon: X \otimes X^* \rightarrow I$, satisfying the ‘snake equations’: $\varepsilon \otimes \text{id}_{X^*} ; \text{id}_X \otimes \eta = \text{id}_X$ and $\text{id}_X \otimes \eta ; \varepsilon \otimes \text{id}_{X^*} = \text{id}_{X^*}$

When thinking string diagrammatically, a dual to an object can be thought of as a wire ‘running backwards’. However, in this paper we are only concerned with categories in which

objects are *self-dual*: any object X is equal to X^* . For the self-dual case, the unit and counit are drawn string diagrammatically as \square_n^n ('cup') and $n\square_n^n$ ('cap') respectively. When drawing the snake equations string using the cup and cap as in Figure 8, the reasoning behind their name becomes apparent.

This graphical notation is suggestive of links between traced and compact closed categories, and this is no accident.

Proposition 3.5 [JSV96, Prop. 3.1]. *Any compact closed category has a trace.*

This trace is called the *canonical trace*; for the self-dual case, it is constructed as follows:

$$\text{Tr}_{m,n}^x \left(\begin{smallmatrix} x & \\ m & f & n \\ & x \end{smallmatrix} \right) := \text{Diagram}$$

So to use the Frobenius results in the traced setting, we just need to show that $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is compact closed; this is also a well-known result.

Lemma 3.6 [RSW05, Prop. 2.8]. *Every hypergraph category is self-dual compact closed.*

Proof. The cup is defined as $\square_n^n := \bullet \bullet \square_n^n$ and the cap as $n\square_n^n := \square_n^n \bullet \bullet$. The snake equations follow by applying the Frobenius equation and unitality:

$$\text{Diagram} = \text{Diagram} = \dots \quad \text{Diagram} = \text{Diagram} = \dots \quad \square$$

Corollary 3.7. $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is compact closed.

Corollary 3.8. $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ has a trace.

A STMC freely generated over a signature faithfully embeds into a CCC generated over the same signature, mapping the trace in the former to the canonical trace in the latter.

Lemma 3.9. \mathbf{T}_Σ is a subcategory of $\mathbf{S}_\Sigma + \mathbf{Frob}$.

Proof. Since \mathbf{Hyp}_Σ is compact closed, it has a (canonical) trace. For \mathbf{T}_Σ to be a subcategory of \mathbf{H}_Σ , every morphism of the former must also be a morphism on the latter. Since the two categories are freely generated (with the trace constructed through the Frobenius generators in the latter), all that remains is to check that every morphism in \mathbf{T}_Σ is a unique morphism in \mathbf{H}_Σ , i.e. the equations of **Frob** do not merge any together. This is trivial since the equations do not apply to the construction of the canonical trace. \square

Definition 3.10. Let $[-]_\Sigma^\mathbf{T}: \mathbf{T}_\Sigma \rightarrow \mathbf{S}_\Sigma + \mathbf{Frob}$ be the inclusion functor of Lemma 3.9.

Corollary 3.11. $[-]_\Sigma^\mathbf{T}$ is faithful.

Proof. $[-]_\Sigma^\mathbf{T}$ is an inclusion functor. \square

Crucially, this mapping is not *full*: there are terms in a CCC that are not terms in a STMC, such as $\phi \square$. In order to identify a category of cospans of hypergraphs for traced terms we must restrict the cospans of hypergraphs in $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ in some way.

3.2. Properties of hypergraph cospans. In [BGK⁺22b], it is shown that terms in a (non-traced) symmetric monoidal category are interpreted via a faithful functor into a sub-PROP of $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ named $\mathbf{MACsp}_D(\mathbf{Hyp}_\Sigma)$, containing only the ‘monogamous acyclic’ cospans. We will examine these conditions of monogamicity and acyclicity, and show how these can be adapted to the traced case.

Considered informally, *monogamy* means that every node has exactly one ‘in’ and ‘out’ connection, be it to an edge or an interface. This corresponds to the fact that wires in symmetric monoidal categories cannot arbitrarily fork or join.

Definition 3.12. For a hypergraph $F \in \mathbf{Hyp}$, the *degree* of a node $v \in F_\star$ is a tuple (i, o) where i is the number of hyperedges with v as a target, and o is the number of hyperedges with v as a source.

Definition 3.13 (Monogamy [BGK⁺22b, Def. 13]). For a cospan of hypergraphs $m \xrightarrow{f} F \xleftarrow{g} n$ in $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$, let $\mathbf{in}(F)$ be the image of f and let $\mathbf{out}(F)$ be the image of g . The cospan $m \xrightarrow{f} F \xleftarrow{g} n$ is *monogamous* if f and g are mono and, for all nodes v , the degree of v is:

$$\begin{array}{ll} (0, 0) & \text{if } v \in \mathbf{in}(F) \wedge v \in \mathbf{out}(F) \\ (1, 0) & \text{if } v \in \mathbf{out}(F) \\ (0, 1) & \text{if } v \in \mathbf{in}(F) \\ (1, 1) & \text{otherwise} \end{array}$$

The second condition on cospans in $\mathbf{MACsp}_D(\mathbf{Hyp}_\Sigma)$ is that all hypergraphs are *acyclic*, as in symmetric monoidal terms all wires must flow from left to right.

Definition 3.14 (Predecessor [BGK⁺22b, Def. 18]). A hyperedge e is a *predecessor* of another hyperedge e' if there exists a node v such that v is a target of e and a source of e' .

Definition 3.15 (Path [BGK⁺22b, Def. 19]). A *path* between two hyperedges e and e' is a sequence of hyperedges e_0, \dots, e_{n-1} such that $e = e_0$, $e' = e_{n-1}$, and for each $i < n - 1$, e_i is a predecessor of e_{i+1} . A subgraph H is the *start* or *end* of a path if it contains a node in the sources of e or the targets of e' respectively.

Since nodes are single-element subgraphs, one can also talk about paths from nodes.

Definition 3.16 (Acyclicity [BGK⁺22b, Def. 20]). A hypergraph F is acyclic if there is no path from a node to itself. A cospan $m \rightarrow F \leftarrow n$ is acyclic if F is.

Cospans of hypergraphs with these properties form a sub-PROP of $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$.

Lemma 3.17 [BGK⁺22b, Lems. 15-17]. *The following statements hold:*

- (1) *identities and symmetries are monogamous;*
- (2) *monogamicity is preserved by composition; and*
- (3) *monogamicity is preserved by tensor.*

Proof. For (1), identities and symmetries are monogamous by definition, as they are constructed from discrete hypergraphs, in which every node is an input and an output. For (2), as pushouts along monos in \mathbf{Hyp}_Σ are monos, the legs of the composition of monogamous cospans must also be mono; moreover as each node in the outputs of the first cospan is merged with exactly one node in the inputs of the second, only nodes of degree at most $(1, 1)$ can be created. For (3), the degree of each node is the same as in the original two cospans, and the coproduct of monos is mono. \square

Corollary 3.18. *There is a sub-PROP of $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$, named $\mathbf{MACsp}_D(\mathbf{Hyp}_\Sigma)$, containing only the monogamous acyclic cospans of hypergraphs.*

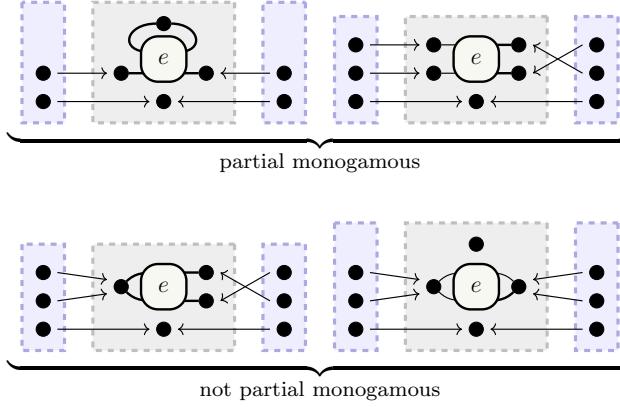


Figure 9: Examples of cospans that are and are not partial monogamous

These two conditions are enough to establish a correspondence between monogamous acyclic cospans of hypergraphs and symmetric monoidal terms.

Theorem 3.19 [BGK⁺22b, Cor. 26]. *There is an isomorphism $\mathbf{S}_\Sigma \cong \mathbf{MACsp}_D(\mathbf{Hyp}_\Sigma)$.*

We want to show a similar result for *traced* terms. Clearly, to model trace the acyclicity condition must be dropped. For the most part, monogamy also applies to the traced case: wires cannot arbitrarily fork and join. There is one nuance: the trace of the identity. This is depicted as a closed loop $\text{Tr}^1 \left(\square \right) = \bigcirc$, and one might think that it can be discarded, i.e. $\bigcirc = \square$. This is *not* always the case, such as in the category of finite dimensional vector spaces [Has97, Sec. 6.1].

These closed loops must be represented in the hypergraph framework: there is a natural representation as a lone node disconnected from either interface. In fact, this is exactly how the canonical trace applied to an identity is interpreted in $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$. This means we need a weaker version of monogamy, which we dub *partial monogamy*.

Definition 3.20 (Partial monogamy). For a cospan $m \xrightarrow{f} F \xleftarrow{g} n \in \mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$, let $\text{in}(F)$ be the image of f and let $\text{out}(F)$ be the image of g . The cospan $m \xrightarrow{f} F \xleftarrow{g} n$ is *partial monogamous* if f and g are mono and, for all nodes $v \in F_*$, the degree of v is

$$\begin{array}{ll} (0,0) & \text{if } v \in \text{in}(F) \wedge v \in \text{out}(F) \\ (1,0) & \text{if } v \in \text{out}(F) \\ & \quad (0,1) \text{ if } v \in \text{in}(F) \\ & \quad (0,0) \text{ or } (1,1) \text{ otherwise} \end{array}$$

Intuitively, partial monogamy means that each node has either exactly one ‘in’ and one ‘out’ connection to an edge or to an interface, or none at all.

Example 3.21. Examples of cospans that are and are not partial monogamous are shown in Figure 9.

In order to establish a correspondence between cospans of partial monogamous hypergraphs and traced terms, the former need to be assembled into a sub-PROP of $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$.

Lemma 3.22. *Identities and symmetries are partial monogamous.*

Proof. By Lemma 3.17, identities and symmetries are monogamous, so they must also be partial monogamous. \square

Lemma 3.23. *Given partial monogamous cospans $m \rightarrow F \leftarrow n$ and $n \rightarrow G \leftarrow p$, the composition $(m \rightarrow F \leftarrow n) ; (n \rightarrow G \leftarrow p)$ is partial monogamous.*

Proof. As with the proof for preservation of regular monogamicity by composition, the legs of a composed cospan must be mono as pushouts along monos are themselves mono.

To verify the degrees of nodes in the composition respect partial monogamy, recall that the only changes in nodes in the original two hypergraphs F and G when compared to the corresponding nodes in the composite is that i -th node in the outputs of F is merged with the i -th node in the inputs of G , and their degrees summed. The only freedoms permitted by partial monogamy over regular monogamy are:

- there can be cycles, but by definition of a path any node in a cycle must have degree $(1, 1)$, so cannot be in the image of the interfaces; and
- nodes with degree $(0, 0)$ can occur outside of the image of the interfaces, so these will be unaffected by composition as well.

So the nodes than can be altered by composition are precisely those permitted by regular monogamy, i.e. nodes with degree $(0, 0)$ in the inputs of F , and nodes with degree $(1, 0)$; as we already know that composition preserves the monogamicity degree conditions for these nodes, then it must also preserve the partial monogamy degree conditions. \square

Lemma 3.24. *Given partial monogamous cospans $m \rightarrow F \leftarrow n$ and $p \rightarrow G \leftarrow q$, the tensor $(m \rightarrow F \leftarrow n) \otimes (p \rightarrow G \leftarrow q)$ is partial monogamous.*

Proof. As with composition, tensor preserves monogamicity by Lemma 3.17, and as it does not affect the degree of nodes then it preserves partial monogamy as well. \square

As partial monogamicity is preserved by both forms of composition, the partial monogamous cospans themselves form a PROP.

Definition 3.25. Let $\text{PMCsp}_D(\mathbf{Hyp}_\Sigma)$ be the sub-PROP of $\text{Csp}_D(\mathbf{Hyp}_\Sigma)$ containing only the partial monogamous cospans of hypergraphs.

To show that $\text{PMCsp}_D(\mathbf{Hyp}_\Sigma)$ is a *traced* PROP, we must show that $\text{PMCsp}_D(\mathbf{Hyp}_\Sigma)$ has a trace. Although $\text{Csp}_D(\mathbf{Hyp}_\Sigma)$ already has a trace, we must make sure that this does not degenerate for cospans of partial monogamous hypergraphs.

Theorem 3.26. *The canonical trace is a trace on $\text{PMCsp}_D(\mathbf{Hyp}_\Sigma)$.*

Proof. Consider a partial monogamous cospan $x + m \xrightarrow{f+h} F \xleftarrow{g+k} x + n$; we must show that its trace $m \xrightarrow{h} F' \xleftarrow{k} n$ is partial monogamous. For each node $a \in x$, $f(a)$ and $g(a)$ are merged together in the traced graph, summing their degrees. If a node is in the image of h or k , this is also the case in the traced cospan. We consider the various cases:

- if $f(a) = g(a)$, then this node must have degree $(0, 0)$; the traced node will still have degree $(0, 0)$ and will no longer be in the interface;
- if $f(a)$ is also in the image of $n \rightarrow F$ and $g(a)$ is also in the image of $m \rightarrow F$, then both $f(a)$ and $g(a)$ have degree $(0, 0)$; the traced node will still have degree $(0, 0)$ and be in both interfaces of the traced cospan;
- if $f(a)$ is in the image of $n \rightarrow F$, then $f(a)$ has $(0, 0)$ and $g(a)$ has degree $(1, 0)$, so the traced node has degree $(1, 0)$ and is in the image of $n \rightarrow F'$;
- if $g(a)$ is in the image of $m \rightarrow F$, then the above applies in reverse; and
- if neither node is in the image of $m \rightarrow F$ and $n \rightarrow F$, then the traced node will have degree $(1, 1)$ and be in the image of no interface.

In all these cases, partial monogamy is preserved. \square

Crucially, while we leave $\text{PMCsp}_D(\mathbf{Hyp}_\Sigma)$ in order to construct the trace using the cup and cap, the resulting cospan *is* in $\text{PMCsp}_D(\mathbf{Hyp}_\Sigma)$.

3.3. From terms to graphs. Partial monogamous hypergraphs are the domain in which we will interpret terms in symmetric monoidal theories equipped with a trace. A (*traced*) *PROP morphism* is a strict (traced) symmetric monoidal functor between PROPs. For $\text{PMCsp}_D(\mathbf{Hyp}_\Sigma)$ to be suitable for reasoning with a traced category \mathbf{T}_Σ for some given signature, there must be a *full and faithful* PROP morphism $\mathbf{T}_\Sigma \rightarrow \text{PMCsp}_D(\mathbf{Hyp}_\Sigma)$.

We exploit the interplay between compact closed and traced categories in order to reuse the existing PROP morphisms from [BGK⁺22a] for the traced case.

To map from traced terms to cospans of hypergraphs, we translate them into $\mathbf{S}_\Sigma + \mathbf{Frob}$ using the inclusion functor $[-]_\Sigma^T$, then use the previously defined PROP morphism $\langle\langle - \rangle\rangle_\Sigma$ to map into $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$. A diagram showing the interaction of these PROP morphisms can be seen in Figure 11.

We need to show that $\langle\langle - \rangle\rangle_\Sigma \circ [-]_\Sigma^T$ is full and faithful when restricted to $\text{PMCsp}_D(\mathbf{Hyp}_\Sigma)$. Showing that it is faithful is simple: $[-]_\Sigma^T$ is faithful as it is an inclusion, and the components of $\langle\langle - \rangle\rangle_\Sigma$ are known to be faithful by the following result.

Proposition 3.27 [BGK⁺22a, Cor. 4.3]. $\llbracket - \rrbracket_\Sigma$ *is faithful*.

Proof (Sketch). This follows by combining the so-called 3-for-2 property [MS09, Thm. 3.3] with the fact that $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma) \cong \mathbf{S}_\Sigma + \mathbf{Frob}$ is a pushout in the category of small SMCs. \square

Proposition 3.28. $[-]_\Sigma$ *is faithful*.

Proof. By definition of $[-]_\Sigma$. \square

The more interesting proof is that of the fullness of $\langle\langle - \rangle\rangle_\Sigma \circ [-]_\Sigma^T$. Following the strategy of [BGK⁺22a], we will assemble an arbitrary partial monogamous cospan of hypergraphs into a form constructed of components mapped from \mathbf{T}_Σ .

Theorem 3.29 [BGK⁺22b, Thm. 25]. *A cospan $m \rightarrow F \leftarrow n$ is in the image of $\llbracket - \rrbracket_\Sigma$ if and only if $m \rightarrow F \leftarrow n$ is monogamous acyclic.*

Proof (Sketch). The only if direction follows by induction on morphisms on \mathbf{S}_Σ . The if direction is by induction on the hyperedges in F ; if there are no hyperedges then $m \rightarrow F \leftarrow n$ is in the image of a morphism containing only identities and symmetries, and for the inductive step each edge corresponds to the image of a generator in some larger term. \square

Theorem 3.30. *A cospan $m \rightarrow F \leftarrow n$ is in the image of $\langle\langle - \rangle\rangle_\Sigma \circ [-]_\Sigma^T$ if and only if it is partial monogamous.*

Proof. For the (\Rightarrow) direction, the generators of \mathbf{T}_Σ are mapped to monogamous cospans by $\langle\langle - \rangle\rangle_\Sigma \circ [-]_\Sigma^T$, and partial monogamy is preserved by composition (Lemma 3.23), tensor (Lemma 3.24), and trace (Theorem 3.26).

For the (\Leftarrow) direction, we show that any partial monogamous cospan $m \xrightarrow{f} F \xleftarrow{g} n$ is in the image of $\langle\langle - \rangle\rangle_\Sigma \circ [-]_\Sigma^T$ by constructing an isomorphic trace of cospans, in which each component under the trace is in the image of $\llbracket - \rrbracket_\Sigma$. The components of the new cospan are as follows:

- let L be the discrete hypergraph containing nodes with degree $(0, 0)$ that are not in the image of f or g ;
- let S and T be the discrete hypergraphs containing the source and target nodes of hyperedges in F respectively, with the ordering determined by some order e_1, e_2, \dots, e_n on the edges in F .
- let E be the hypergraph containing the nodes of $S + T$ and the hyperedges of F , in which the sources of edges of E are the nodes in S that are the sources of the edge in F , and the targets of an edge are the nodes in T that are the targets of the edge in F ; and
- let V be the discrete hypergraph containing all the nodes of F ; and

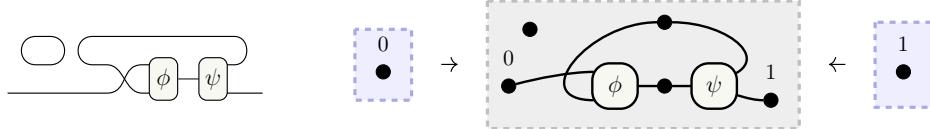
These parts can be composed to form the following composite:

$$L + T + m \xrightarrow{\text{id} + \text{id} + f} L + V \xleftarrow{\text{id} + \text{id} + g} L + S + n ; L + S + n \xrightarrow{\text{id}} L + E + n \xleftarrow{\text{id}} L + T + n$$

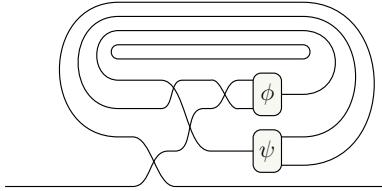
We take the trace of $L + T$ over this composite to obtain a cospan isomorphic to the original. The components of the composite under the trace are all monogamous acyclic so are in the image of $\llbracket - \rrbracket_\Sigma$ by Theorem 3.29; this means there is a term $f \in \mathbf{S}_\Sigma$ such that $\llbracket f \rrbracket_\Sigma$ is isomorphic to the original composite. The trace of f is in \mathbf{T}_Σ , so the trace of the composite is in the image of $\langle\langle - \rangle\rangle_\Sigma \circ \llbracket - \rrbracket_\Sigma^T$. \square

The large composite cospan may appear rather impenetrable at first glance; essentially, it is constructed by stacking up the edges in the cospan $\tilde{m} \rightarrow \tilde{E} \leftarrow \tilde{n}$, and joining up the targets to the appropriate sources by tracing them around and shuffling them to the correct source. The graph L contains any identity loops.

Example 3.31. Consider the following term and its cospan interpretation:



We assemble the latter into the composite cospan of Theorem 3.30 as shown in Figure 10. Both of the components under the trace correspond to terms in \mathbf{S}_Σ , so applying the trace to this produces a term in \mathbf{T}_Σ :



This term is equal to the original by string diagrammatic deformations.

This brings us to the first major result of this paper: that terms in a freely generated symmetric traced monoidal category are in a one-to-one correspondence with isomorphism classes of partial monogamous cospans of hypergraphs.

Corollary 3.32. $\mathbf{T}_\Sigma \cong \text{PMCsp}_D(\mathbf{Hyp}_\Sigma)$.

This means we can now freely translate between traced string diagram terms and partial monogamous cospans of hypergraphs. Crucially, terms that are equal by the axioms of STMCs are mapped to isomorphic cospans; to verify if two traced terms $\llbracket f \rrbracket$ and $\llbracket g \rrbracket$ are equal in \mathbf{T}_Σ , we can check if their hypergraph interpretations are isomorphic.

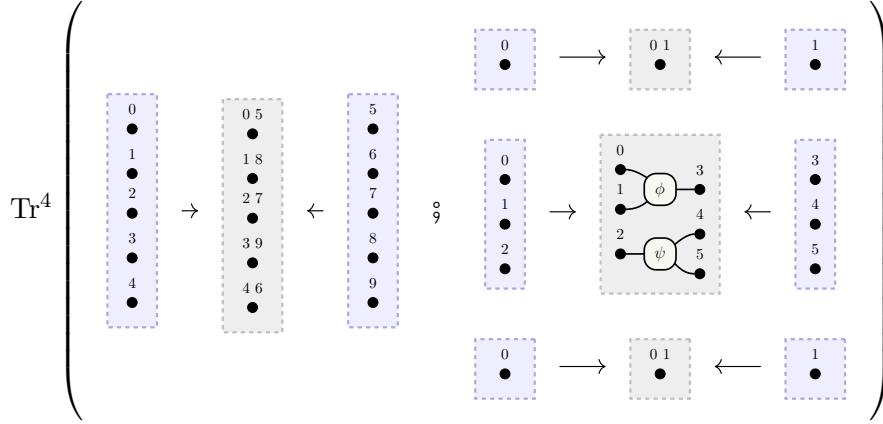


Figure 10: The cospan of Example 3.31 in the form of Theorem 3.30

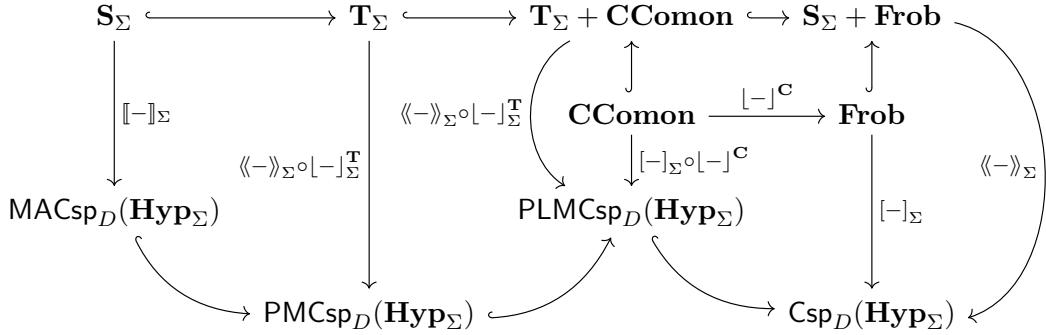
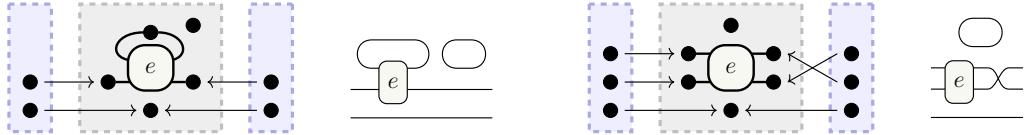


Figure 11: Interactions between categories of terms and hypergraphs

Example 3.33. The partial monogamous cospans from Figure 9 are shown below with their corresponding terms in \mathbf{T}_Σ .



Note that the position of the ‘closed loop’ could be moved anywhere in the term, but the hypergraph interpretation would be unchanged.

This result is also important for rewriting traced terms modulo some equational theory, in which the graph interpretations themselves will have to be rewritten. Corollary 3.32 will be instrumental in showing that any such rewriting system is sound and complete, that is to say a graph rewrite $\langle\langle \boxed{f} \rangle\rangle_\Sigma \xrightarrow{\mathbf{T}} \langle\langle \boxed{g} \rangle\rangle_\Sigma$ is valid if and only if $\boxed{f} = \boxed{g}$ under the equational theory.

4. HYPERGRAPHS FOR TRACED COMMUTATIVE COMONOID CATEGORIES

The trace is but one kind of structure we are interested in adding to symmetric monoidal terms. We also want to be able to *fork* and *eliminate* wires by adding a (*commutative*)

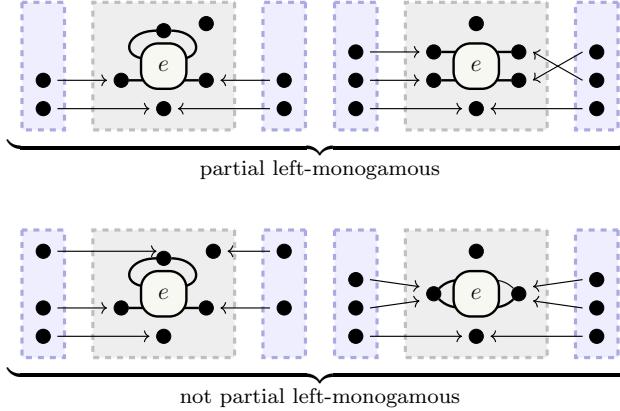


Figure 12: Examples of cospans that are and are not partial left-monogamous

comonoid structure; categories equipped with such a structure are also known as *gs-monoidal* (*garbage-sharing*) categories [FL23].

Definition 4.1. Let $(\Sigma_{\text{CCComon}}, \mathcal{E}_{\text{CCComon}})$ be the symmetric monoidal theory of *commutative comonoids*, with $\Sigma_{\text{CCComon}} := \{ \text{---} \bullet \text{---}, \text{---} \square \text{---} \}$ and $\mathcal{E}_{\text{CCComon}}$ defined as in Figure 3. We write $\text{CCComon} := \mathbf{S}_{\Sigma_{\text{CCComon}}, \mathcal{E}_{\text{CCComon}}}$.

From now on, we write ‘comonoid’ to mean ‘commutative comonoid’. There has already been work using hypergraphs for PROPs with a (co)monoid structure [FL23, MPZ23] but these consider *acyclic* hypergraphs: we must ensure that removing the acyclicity condition does not lead to any degeneracies.

Definition 4.2 (Partial left-monogamy). For a cospan $m \xrightarrow{f} H \xleftarrow{g} n$, we say it is *partial left-monogamous* if f is mono and, for all nodes $v \in H_*$, the degree of v is

$$\begin{cases} (0, m) & \text{if } v \text{ is in the image of } f \\ (0, m) \text{ or } (1, m) & \text{otherwise} \end{cases}$$

for some $m \in \mathbb{N}$.

Partial left-monogamy is a weakening of partial monogamy that allows nodes to have multiple ‘out’ connections, which represent the use of the comonoid structure to fork wires.

Example 4.3. Examples of cospans that are and are not partial left-monogamous are shown in Figure 12.

Remark 4.4. As with the nodes not in the interfaces with degree $(0, 0)$ in the vanilla traced case, the nodes not in the interface with degree $(0, m)$ allow for the interpretation of terms such as $\text{Tr}(\text{---} \bullet \text{---})$.

Lemma 4.5. *Identities and symmetries are partial left-monogamous.*

Proof. Again by Lemma 3.17, identities and symmetries are monogamous so they are also partial left-monogamous. \square

Lemma 4.6. *Given partial left-monogamous cospans $m \rightarrow F \leftarrow n$ and $n \rightarrow G \leftarrow p$, the composition $(m \rightarrow F \leftarrow n) ; (n \rightarrow G \leftarrow p)$ is partial left-monogamous.*

Proof. We only need to check the in-degree of nodes, as the out-degree can be arbitrary. Only the nodes in the image of $n \rightarrow G$ have their in-degree modified; they will gain the in-tentacles of the corresponding nodes in the image of $n \rightarrow F$. Initially the nodes in $n \rightarrow G$ must have in-degree 0 by partial monogamy. They can only gain at most one in-tentacle from nodes in $n \rightarrow F$ as each of these nodes has in-degree 0 or 1 and $n \rightarrow G$ is mono. So the composite graph is partial left-monogamous. \square

Lemma 4.7. *Given partial left-monogamous cospans $m \rightarrow F \leftarrow n$ and $p \rightarrow G \leftarrow q$, the tensor $(m \rightarrow F \leftarrow n) \otimes (p \rightarrow G \leftarrow q)$ is partial left-monogamous.*

Proof. The elements of the original graphs are unaffected so the degrees are unchanged. \square

Definition 4.8. Let $\text{PLMCsp}_D(\mathbf{Hyp}_\Sigma)$ be the sub-PROP of $\text{Csp}_D(\mathbf{Hyp}_\Sigma)$ containing only the partial left-monogamous cospans of hypergraphs.

Proposition 4.9. *The canonical trace is a trace on $\text{PLMCsp}_D(\mathbf{Hyp}_\Sigma)$.*

Proof. We must show that for any set of nodes in the image of $x + n \rightarrow K$ merged by the canonical trace, at most one of them can have in-degree 1. But this must be the case because any image in the image of $x + m \rightarrow K$ must have in-degree 0, and $x + m \rightarrow K$ is moreover mono so it cannot merge nodes in the image of $x + n \rightarrow K$. \square

This category can be equipped with a comonoid structure.

Definition 4.10. Let $[-]^C : \mathbf{CComon} \rightarrow \mathbf{Frob}$ be the obvious embedding of \mathbf{CComon} into \mathbf{Frob} , and let $[-]_\Sigma : \mathbf{T}_\Sigma + \mathbf{Comon} \rightarrow \mathbf{S}_\Sigma + \mathbf{Frob}$ be the copairing of $[-]_\Sigma^T$ and $[-]^C$.

As before, these PROP morphisms are summarised in Figure 11. To show that partial left-monogamy is the correct notion to characterise terms in a traced comonoid setting, it is necessary to ensure that the image of these PROP morphisms lands in $\text{PLMCsp}_D(\mathbf{Hyp}_\Sigma)$.

Lemma 4.11. *The image of $[-]_\Sigma \circ [-]^C$ is in $\text{PLMCsp}_D(\mathbf{Hyp}_\Sigma)$.*

Proof. By definition. \square

Corollary 4.12. *The image of $\langle\langle - \rangle\rangle_\Sigma \circ [-]_\Sigma$ is in $\text{PLMCsp}_D(\mathbf{Hyp}_\Sigma)$.*

To show the correspondence between $\mathbf{T}_\Sigma + \mathbf{CComon}$ and $\text{PLMCsp}_D(\mathbf{Hyp}_\Sigma)$, we use a similar strategy to the one of Theorem 3.30.

Lemma 4.13. *Given a discrete hypergraph $X \in \mathbf{Hyp}_\Sigma$, any cospan $m \xrightarrow{f} X \leftarrow n$ with f mono is in the image of $[-]_\Sigma \circ [-]^C$.*

This leads to a version of Corollary 3.32 for traced terms additionally equipped with a comonoid structure: every such term is in one-to-one correspondence with isomorphism classes of partial left-monogamous cospans of hypergraphs.

Theorem 4.14. $\mathbf{T}_\Sigma + \mathbf{CComon} \cong \text{PLMCsp}_D(\mathbf{Hyp}_\Sigma)$.

Proof. Since $\langle\langle - \rangle\rangle_\Sigma$ and $[-]^C$ are faithful, it suffices to show that a cospan $m \rightarrow F \leftarrow n$ in $\text{PLMCsp}_D(\mathbf{Hyp}_\Sigma)$ can be decomposed into a traced cospan in which every component under the trace is in the image of either $\langle\langle - \rangle\rangle_\Sigma$ or $[-]_\Sigma \circ [-]^C$. This is achieved by taking the

construction of Theorem 3.30 and allowing the first component to be partial left-monogamous; by Lemma 4.13 this is in the image of $[-]_\Sigma \circ [-]^C$. The remaining components remain in the image of $\llbracket - \rrbracket_\Sigma$. Subsequently, the entire traced cospan must be in the image of $\langle\langle - \rangle\rangle_\Sigma \circ \llbracket [-]^\mathbf{T} [-]^C \rrbracket$. \square

Much like with the result for traced terms and partial monogamous cospans of hypergraphs, this result is key for both performing rewriting on traced comonoid terms and for showing the soundness and completeness of such a rewriting system.

5. GRAPH REWRITING

We have now shown that we can reason up to the axioms of symmetric traced categories with a comonoid structure using hypergraphs: string diagrams equal by topological deformations are translated into isomorphic cospans of hypergraphs. Already this is a useful tool to have for reasoning with string diagrams, but ultimately this only allows us to ‘move boxes and wires about’ while preserving the connectivity between them; the boxes themselves cannot be altered.

Reasoning with string diagrams becomes more interesting when performed modulo a *monoidal theory*, in which we have additional equations between terms. These equations can be used to replace certain patterns of boxes and wires with others, actually changing the make-up of a diagram. Some examples of useful monoidal theories will be examined in Section 6.

The process of translating one traced string diagram term into another is defined formally as *term rewriting*.

Definition 5.1 (Term rewriting). A *rewriting system* \mathcal{R} for a traced PROP \mathbf{T}_Σ consists of a set of *rewrite rules* $\langle i \xrightarrow{l} j, i \xrightarrow{r} j \rangle$. Given terms $m \xrightarrow{g} n$ and $m \xrightarrow{h} n$ in \mathbf{T}_Σ we write $\xrightarrow{g} \Rightarrow_{\mathcal{R}} \xrightarrow{h}$ if there exists rewrite rule $(i \xrightarrow{l} j, i \xrightarrow{r} j)$ in \mathcal{R} and $\xrightarrow{j} \xrightarrow{c} i$ in \mathbf{T}_Σ such that $\xrightarrow{g} = \begin{array}{c} l \\ \square \\ c \\ r \end{array}$ and $\xrightarrow{h} = \begin{array}{c} r \\ \square \\ c \\ l \end{array}$ by axioms of STMCs.

Term rewriting using string diagrams is convenient for pen-and-paper reasoning, but as we have already mentioned is difficult to automate. Now armed with hypergraph interpretations of string diagram terms, we can turn to *graph rewriting*.

Graph rewriting is a deeply studied field with a plethora of techniques. To tie in with our categorical motivations, we use the framework of *double pushout rewriting* (DPO rewriting), which has its roots in the 70s [EK76]. Rather than using the original presentation, we use an extension, known as *double pushout rewriting with interfaces* (DPOI rewriting) [BGK⁺17]. This definition is advantageous as confluence of graph rewriting using interfaces has been shown to be decidable [BGK⁺17, Cor. 20].

In DPO rewriting, rewrite rules are specified as pairs of morphisms from some shared interface graph. While the framework can be applied to all manner of graph-based structures, the definitions we present will be in terms of hypergraphs.

Definition 5.2 (DPO rule). Given interfaced hypergraphs $i \xrightarrow{a_1} L \xleftarrow{a_2} j$ and $i \xrightarrow{b_1} R \xleftarrow{b_2} j$, their *DPO rule* in \mathbf{Hyp}_Σ is a span $L \xleftarrow{[a_1, a_2]} i + j \xrightarrow{[b_1, b_2]} R$.

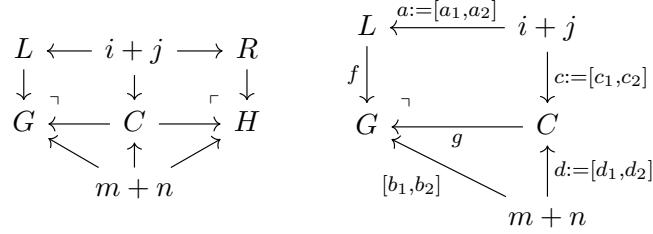


Figure 13: The DPO diagram and a pushout complement

A DPO rule makes up part of a larger diagram that represents the application of said rule in a larger context graph.

Definition 5.3 (DPO(I) rewriting). Let \mathcal{R} be a set of DPO rules. Then, for morphisms $G \leftarrow m + n$ and $H \leftarrow m + n$ in \mathbf{Hyp}_Σ , there is a rewrite $G \rightsquigarrow_{\mathcal{R}} H$ if there exist a rule $L \leftarrow i + j \rightarrow R \in \mathcal{R}$ and cospan $i + j \rightarrow C \leftarrow m + n \in \mathbf{Hyp}_\Sigma$ such that the diagram in the left of Figure 13 commutes.

The first thing to note is that the graphs in the DPO diagram have a *single* interface $G \leftarrow m + n$ instead of the cospans $m \rightarrow G \leftarrow n$ we are used to. Before performing DPO rewriting in \mathbf{Hyp}_Σ , the interfaces must be ‘folded’ into one.

Definition 5.4 [BGK⁺22b, Sec. 4.4]. Let $\lceil - \rceil: \mathbf{S}_\Sigma + \mathbf{Frob} \rightarrow \mathbf{S}_\Sigma + \mathbf{Frob}$ be defined as having action $m - \boxed{f} - n \mapsto \overbrace{\boxed{f}}^m - n$.

Note that the result of applying $\lceil - \rceil$ is no longer a valid traced term due to the input wire bending round to the output. This is not an issue for the purpose of rewriting traced terms, as long as we ‘unfold’ the interfaces once rewriting is completed. When viewed through the lens of hypergraphs, the distinction is even less important.

Proposition 5.5 [BGK⁺22a, Prop. 4.8]. For $m - \boxed{f} - n$ in $\mathbf{S}_\Sigma + \mathbf{Frob}$, if $\langle\langle - \boxed{f} - \rangle\rangle_\Sigma = m \xrightarrow{i} F \xleftarrow{o} n$ then $\lceil \langle\langle - \boxed{f} - \rangle\rangle_\Sigma \rceil$ is isomorphic to $0 \rightarrow F \xleftarrow{i+o} m + n$.

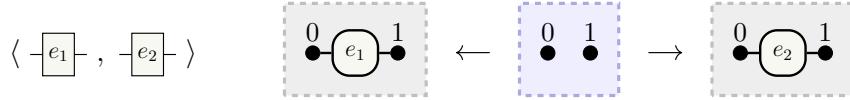
In order to apply a given DPO rule $L \leftarrow i + j \rightarrow R$ in some larger graph $m \rightarrow G \leftarrow n$, a morphism $L \rightarrow G$ must first be identified. The next step is to ‘cut out’ the components of L that exist in G , using what is known as a *pushout complement*. The pushout complement is a key part of DPO rewriting as it defines the rewriting context for a given rule in a larger term.

Definition 5.6 (Pushout complement). Let $i + j \rightarrow L \rightarrow G \rightarrow m + n$ be morphisms in \mathbf{Hyp}_Σ ; their *pushout complement* is an object C with morphisms $i + j \rightarrow C \rightarrow G$ such that $L \rightarrow G \leftarrow C$ is a pushout and the right diagram in Figure 13 commutes.

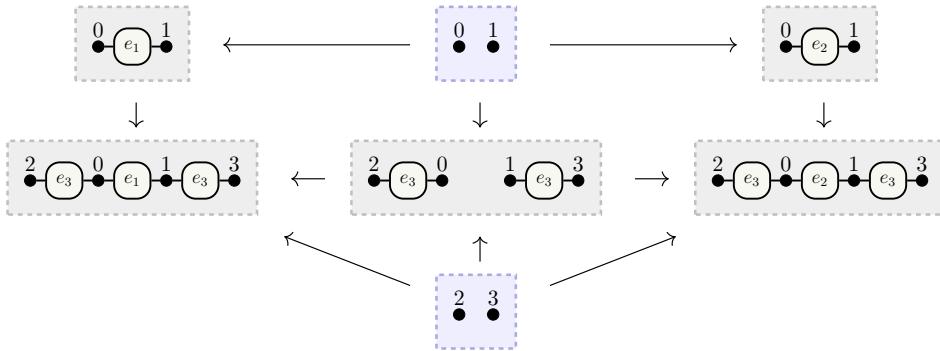
Given a rule $L \leftarrow i + j \rightarrow R$ and morphism $L \rightarrow G$, a pushout complement $i + j \rightarrow C \rightarrow G$ is effectively C is ‘ G with L cut out of it’.

Once a pushout complement is computed, the pushout of $C \leftarrow i + j \rightarrow R$ can be performed to obtain the completed rewrite H . Put simply, the pushout $L \rightarrow H \leftarrow R$ pastes R into C along the dangling wires left by cutting L out of H .

Example 5.7. Consider the following DPO rule:



Now consider the term $\boxed{-e_3-} \boxed{e_1} \boxed{-e_3-}$; a DPO rewrite is performed as follows:

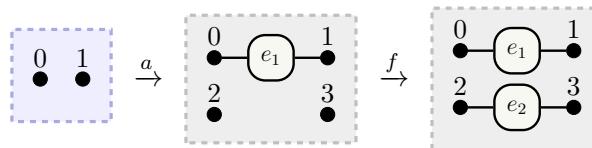


As one would expect, the resulting hypergraph is the interpretation of term $\boxed{-e_3-} \boxed{e_2} \boxed{-e_3-}$.

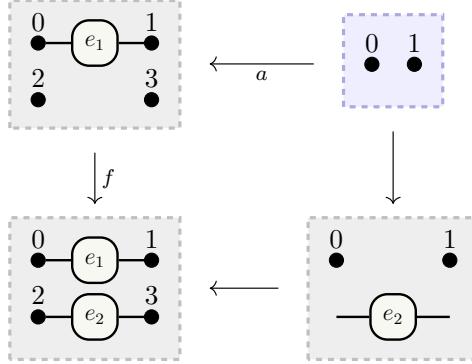
A pushout complement may not exist for a given rule and matching; there are two conditions that must be satisfied for this to be the case. The first ensures that all the sources and targets of a hyperedge are present in a candidate context.

Definition 5.8 (No-dangling-hyperedges condition [CMR⁺97, Prop. 3.3.4]). Given morphisms $i + j \xrightarrow{a} L \xrightarrow{f} G$ in \mathbf{Hyp}_Σ , they satisfy the *no-dangling* condition if, for every hyperedge not in the image of f , each of its source and target nodes is either not in the image of f or are in the image of $f \circ a$.

Example 5.9. The following morphisms do not satisfy the no-dangling-hyperedges condition.



To obtain the pushout complement we ‘cut out’ any vertices in the rightmost graph which are in the image of f but not the image of $f \circ a$, as the latter are the interfaces of the rule. However, if we cut out the vertices labelled 2 and 3, the edge e_2 will be left with ‘dangling’ tentacles connected to no vertices, a malformed hypergraph.

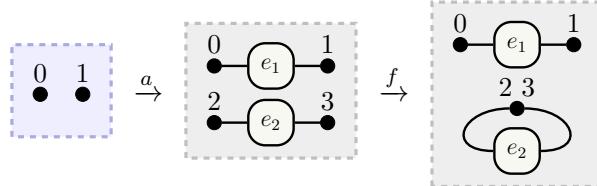


This illustrates why every node in the image of f must also be in the image of $f \circ a$.

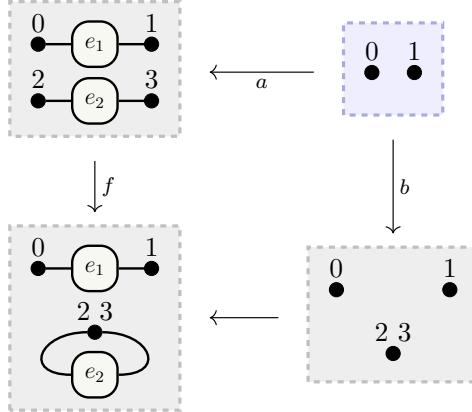
The second condition enforces that merging of vertices is well-defined.

Definition 5.10 (No-identification condition [CMR⁺97, Prop. 3.3.4]). Given morphisms $i + j \xrightarrow{a} L \xrightarrow{f} G$ in \mathbf{Hyp}_Σ , they satisfy the *no-identification* condition if any two distinct elements merged by f are also in the image of $f \circ a$.

Example 5.11. The following morphisms do not satisfy the no-identification condition.



When trying to construct a pushout complement, the edge e_2 will be removed. However, since vertices 2 and 3 are not mapped from the rule interfaces, there is no reason that a pushout would glue them together so that they are merged in the final graph. Therefore no pushout complement can exist.



Although the above diagram may look reasonable, it is *not* a pushout; the pushout can only merge vertices that are in the image of b .

With these two conditions, we can establish when pushout complements exist for a pair of hypergraph homomorphisms. If there is a pushout complement, then there is an opportunity for a rewrite.

Proposition 5.12 [CMR⁺97, Prop. 3.3.4]. *Morphisms $i + j \rightarrow L \rightarrow G$ have at least one pushout complement if and only if they satisfy the no-dangling and no-identification conditions.*

Definition 5.13. Given a partial monogamous cospan $i \rightarrow L \leftarrow j$, a morphism $L \rightarrow G$ is called a *matching* if it has at least one pushout complement.

In certain settings, known as *adhesive categories* [LS04], it is possible to be more precise about the number of pushout complements for a given matching and rewrite rule.

Proposition 5.14 [LS04, Lem. 15]. *In an adhesive category, pushout complements of $i + j \xrightarrow{a} L \rightarrow G$ are unique if they exist and a is mono.*

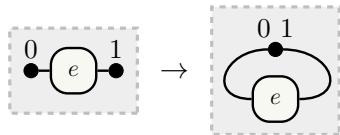
Proposition 5.15 [LS05, Prop. 3.5]. **Hyp_Σ** is adhesive.

A given pushout complement uniquely determines the rewrite performed, so it might seem advantageous to always have exactly one. However, when writing modulo traced comonoid structure there are settings where having multiple pushout complements is beneficial.

5.1. Rewriting with traced structure. In the Frobenius case considered in [BGK⁺22a], all valid pushout complements correspond to a valid rewrite, but this is not the case for the traced monoidal case. In the symmetric monoidal case considered in [BGK⁺22b], pushout complements that correspond to a valid rewrite in the non-traced symmetric monoidal case are identified as *boundary complements*. As with monogamy, we will weaken this definition to find one suitable for the traced case.

Definition 5.16 (Traced boundary complement). A pushout complement as in Definition 5.6 is called a *traced boundary complement* if c_1 and c_2 are mono and $j + m \xrightarrow{[c_2, d_1]} C \xleftarrow{[d_2, c_1]} n + i$ is a partial monogamous cospan.

Unlike [BGK⁺22b], we do not enforce that the matching is mono, as this cuts off potential rewrites in the *traced* setting, such as a matching inside a loop:



The definition of a traced boundary complement leads to a definition of double pushout rewriting for traced terms.

Definition 5.17 (Traced DPO). For morphisms $G \leftarrow m + n$ and $H \leftarrow m + n$ in **Hyp_Σ**, there is a traced rewrite $G \rightsquigarrow_{\mathcal{R}} H$ if there exists a rule $L \leftarrow i + j \rightarrow G \in \mathcal{R}$ and cospan $i + j \rightarrow C \leftarrow n + m \in \mathbf{Hyp}_{\Sigma}$ such that the diagram in Definition 5.3 commutes and $i + j \rightarrow C \rightarrow G$ is a traced boundary complement.

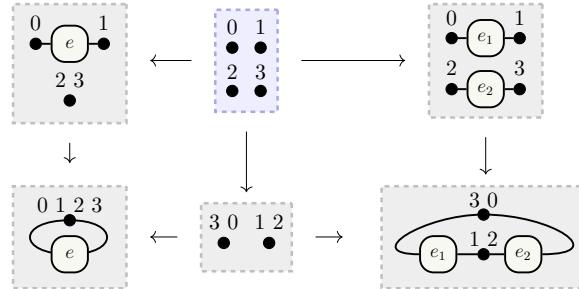
Some intuition on the construction of traced boundary complements may be required: this will be provided through a lemma and some examples.

Lemma 5.18. Consider the DPO diagram as in Figure 13, and let $i + j \rightarrow C \rightarrow G$ be a traced boundary complement. Let $v \in i$ and $w_0, w_1, \dots, w_k \in j$ such that $f(a_1(v)) = f(a_2(w_0))$, $f(a_1(v)) = f(a_2(w_1))$ and so on. Then either (1) there exists exactly one w_l not in the image of d_1 such that $c_1(v) = c_2(w_l)$; (2) $c_1(v)$ is in the image of d_1 ; or (3) $c_1(v)$ has degree $(1, 0)$. The same also holds for $w \in j$, with the interface map as d_2 and the degree as $(0, 1)$.

Proof. Since $c_1(v)$ is in the image of c , it must have either degree $(0, 0)$ or $(1, 0)$ by partial monogamy. For it to have degree $(0, 0)$, it must either be in the image of one of c_2 or d_2 . In the case of the former, this means that there must be a w_l such that $c_1(v) = c_2(w_l)$, and only one such node as c_2 is mono, so (1) is satisfied. In the case of the latter, (2) is immediately satisfied. Otherwise, (3) is satisfied. The proof for the flipped case is exactly the same. \square

Often there can be valid rewrites in the realm of graphs that are non-obvious in the term language. This is because we are rewriting modulo *yanking*.

Example 5.19. Consider the rule $\langle \overline{\quad}, \begin{array}{c} -e_1- \\ -\boxed{e}- \\ -e_2- \end{array} \rangle$. The interpretation of this as a DPO rule in a valid traced boundary complement is illustrated below.



This corresponds to a valid term rewrite:

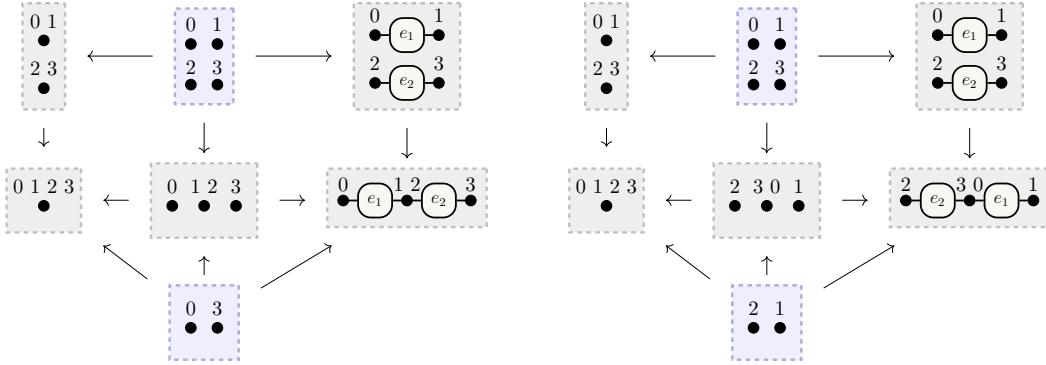
$$\text{Diagram showing a term rewrite: } \text{rule} = \text{left side} = \text{middle side} = \text{right side}$$

The diagram consists of four parts connected by equals signs. The first part is a traced boundary complement with a single input wire labeled e . The second part is a traced boundary complement where the input wire e is split into two wires, one going up and one going down, which then meet at a crossing point. The third part is a traced boundary complement where the input wire e is split into two wires, one going up and one going down, which then meet at a crossing point, and the output wire is labeled e_1 and e_2 . The fourth part is a traced boundary complement where the input wire e is split into two wires, one going up and one going down, which then meet at a crossing point, and the output wires are labeled e_1 and e_2 .

Note that applying yanking is required in the term setting because the traced wire is flowing from right to left, whereas applying the rule requires all wires flowing left to right.

Unlike regular boundary complements, traced boundary complements need not be unique. However, this is not a problem since all pushout complements can be enumerated given a rule and matching [HJKS11].

Example 5.20. Consider the rule $\langle \overline{\quad}, \begin{array}{c} -e_1- \\ -\boxed{e}- \\ -e_2- \end{array} \rangle$. Below are two valid traced boundary complements involving a matching of this rule.



Once again, these derivations arise through yanking:

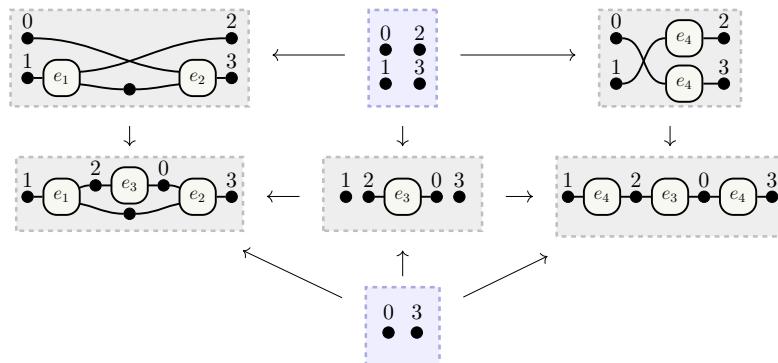
$$\begin{aligned}
 & \text{---} = \text{---} = \text{---} = \text{---} = -e_1 - e_2 - \\
 & \text{---} = \text{---} = \text{---} = \text{---} = -e_2 - e_1 -
 \end{aligned}$$

The diagram shows two separate DPO derivations. Each derivation starts with a horizontal line segment, followed by a crossing loop. This is then followed by a sequence of nodes labeled e_1 and e_2 , enclosed in dashed boxes. Finally, the sequence ends with a minus sign and the labels e_1 or e_2 . The two derivations are connected by arrows indicating they are parallel processes.

Rewriting modulo yanking also eliminates another foible of rewriting modulo (non-traced) symmetric monoidal structure. In the SMC case, the image of the matching must be ‘convex’: any path between nodes must also be captured by a matching. This is not necessary in the traced case.

Example 5.21. Consider the rule $\langle \text{---} , \text{---} \rangle$ and the term $\text{---} \text{---} \text{---}$.

Although it is not immediately obvious, there is in fact a matching of the former in the latter. Performing the DPO procedure yields the following:



In a non-traced setting this is an invalid rule, but it is possible with yanking.

$$\begin{array}{ccc}
 \text{Diagram 1: } & \text{Diagram 2: } & \\
 \text{Diagram 3: } & & \text{Diagram 4: } \\
 & & \text{Diagram 5: }
 \end{array}$$

The diagrams show the equivalence of string rewriting rules involving yanking. Diagram 1 shows $e_1 - e_3 - e_2$ equating to a diagram where e_3 is yanked over e_1 and e_2 . Diagram 3 shows a more complex yanking pattern involving multiple strands and dashed boxes. Diagram 4 shows the result of further simplification, and Diagram 5 shows the final simplified form $e_4 - e_3 - e_4$.

We are almost ready to show the soundness and completeness of this DPO rewriting system with respect to term rewriting, but first we prove some lemmas to be used in the final result. The first is a decomposition lemma, akin to that used in [BGK⁺22b, Lem. 24].

Lemma 5.22 (Traced decomposition). *Given partial monogamous cospans $m \xrightarrow{d_1} G \xleftarrow{d_2} n$ and $i \xrightarrow{a_1} L \xleftarrow{a_2} j$, and a morphism $L \xrightarrow{f} G$ such that $i + j \rightarrow L \rightarrow G$ satisfies the no-dangling and no-identification conditions, then there exists a partial monogamous cospan $j + m \xrightarrow{[c_2, d_1]} C \xleftarrow{[c_1, d_2]} i + n$ such that $m \rightarrow G \leftarrow n$ can be factored as*

$$\text{Tr}^i \left(\begin{array}{c} i \xrightarrow{a_1} L \xleftarrow{a_2} j \\ \otimes \\ m \rightarrow m \leftarrow m \end{array} ; j + m \xrightarrow{[c_2, d_1]} C \xleftarrow{[c_1, d_2]} i + n \right) \quad (5.1)$$

where $j + m \xrightarrow{c_2, d_1} C \xleftarrow{c_1, d_2} i + n$ is a traced boundary complement.

Proof. Let $i + j \xrightarrow{[c_1, c_2]} C \xleftarrow{[d_1, d_2]} m + n$ be defined as a traced boundary complement of $i + j \xrightarrow{[a_1, a_2]} L \xrightarrow{f} G$, which exists as the no-dangling and no-identification condition is satisfied. We assign names to the various cospans in play, and reason string diagrammatically:

$$\begin{array}{ll}
 i - \boxed{l} - j := i \rightarrow L \leftarrow j & \boxed{\hat{l}} - \boxed{i} - j := 0 \rightarrow L \leftarrow i + j \\
 m - \boxed{c} - n := j + m \xrightarrow{[c_2, d_1]} C \xleftarrow{[c_1, d_2]} i + n & j - \boxed{\hat{c}} - n := i + j \xrightarrow{[c_1, c_2]} C \xleftarrow{[d_1, d_2]} m + n \\
 m - \boxed{g} - n := m \rightarrow G \leftarrow n & \boxed{\hat{g}} - \boxed{m} - n := 0 \rightarrow G \leftarrow m + n
 \end{array}$$

Note that the cospans in the left column are partial monogamous by definition of rewrite rules and traced boundary complements. We will show that \boxed{g} can be decomposed into a form using the two cospans \boxed{l} and \boxed{c} , along with identities.

By using the compact closed structure of $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$, we have the following:

$$m - \boxed{g} - n = \boxed{\hat{g}} - \boxed{m} - n \quad j - \boxed{\hat{c}} - n = j - \boxed{c} - n \quad \boxed{\hat{l}} - \boxed{i} - j = \boxed{l} - \boxed{i} - j$$

Since G is the pushout of $L \xleftarrow{[a_1, a_2]} i + j \xrightarrow{[c_1, c_2]} C$ and pushout is cospan composition, we also have that $\hat{g}^m_n = \hat{l}^m_n \hat{c}^m_n$. Putting this all together we can show that

$$\begin{aligned} m - \boxed{g} - n &= \begin{array}{c} m \\ \boxed{\hat{g}} \\ n \end{array} = m \curvearrowright \boxed{\hat{l}} \boxed{\hat{c}} \curvearrowright n \\ &= \begin{array}{c} m \\ \boxed{l} \boxed{\hat{c}} \\ n \end{array} = \begin{array}{c} m \\ \boxed{l} \boxed{c} \\ n \end{array} = m - \boxed{l} - \boxed{c} - n \end{aligned}$$

Since the ‘loop’ is constructed in the same manner as the canonical trace on $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ (and is therefore identical in the graphical notation), this is a term in the form of (5.1). \square

The second result relates to how cospans of hypergraphs need to be ‘bent’ using $\Gamma \dashv$ in order to be used in DPO rewriting. We write $\Gamma \dashv [\mathcal{R}]_\Sigma^\mathbf{T}$ for the pointwise map $(-\boxed{l}-, -\boxed{r}-) \mapsto (\Gamma \dashv \boxed{l}, \Gamma \dashv \boxed{r})$.

Lemma 5.23. *Let \boxed{c}^p_q be a term in $\mathbf{S}_\Sigma + \mathbf{Frob}$; if*

$$\langle\langle \boxed{c} \rangle\rangle_{\Sigma, \mathcal{C}} = m + n \xrightarrow{[f_1, f_2]} F \xleftarrow{[g_1, g_2]} p + q$$

then

$$\langle\langle \boxed{c} \rangle\rangle_{\Sigma, \mathcal{C}} = p + m \xrightarrow{[g_1, f_1]} F \xleftarrow{[f_2, g_2]} n + q.$$

Proof. By definition of cups and caps in $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$. \square

We now show the main result of this section: that traced DPO rewriting on partial monogamous cospans of hypergraphs coincides exactly with term rewriting. That is to say, a term rewrite $\boxed{f} \Rightarrow_{\mathcal{R}} \boxed{g}$ is valid if and only if the hypergraph interpretation of \boxed{f} can be rewritten using traced DPO to the hypergraph interpretation of \boxed{g} .

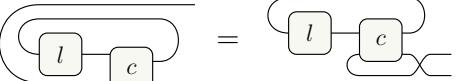
Theorem 5.24. *Let \mathcal{R} be a rewriting system on \mathbf{T}_Σ . Then,*

$$m - \boxed{g} - n \Rightarrow_{\mathcal{R}} m - \boxed{h} - n \text{ if and only if } \langle\langle \Gamma \dashv \boxed{g} \rangle\rangle_\Sigma \rightsquigarrow \langle\langle \Gamma \dashv [\mathcal{R}]_\Sigma^\mathbf{T} \rangle\rangle_\Sigma \langle\langle \Gamma \dashv \boxed{h} \rangle\rangle_\Sigma.$$

Proof. First the (\Rightarrow) direction. If $\boxed{g} \Rightarrow_{\mathcal{R}} \boxed{h}$ then we have $\boxed{g} = \boxed{l} \boxed{c}$ and $\boxed{r} \boxed{c} = \boxed{h}$; we must derive the DPO diagram in \mathbf{Hyp}_Σ . First we give

names to the following cospans:

$$\begin{aligned}
 0 \rightarrow L \leftarrow i + j &:= \langle\langle \Gamma \left[- \boxed{l} - \right]_{\Sigma}^{\mathbf{T}} \rangle\rangle_{\Sigma} = \langle\langle \overbrace{\boxed{l}}^{\text{---}} \rangle\rangle_{\Sigma} \\
 0 \rightarrow R \leftarrow i + j &:= \langle\langle \Gamma \left[- \boxed{r} - \right]_{\Sigma}^{\mathbf{T}} \rangle\rangle_{\Sigma} = \langle\langle \overbrace{\boxed{r}}^{\text{---}} \rangle\rangle_{\Sigma} \\
 0 \rightarrow G \leftarrow m + n &:= \langle\langle \Gamma \left[- \boxed{g} - \right]_{\Sigma}^{\mathbf{T}} \rangle\rangle_{\Sigma} = \langle\langle \overbrace{\boxed{g}}^{\text{---}} \quad l \quad c \rangle\rangle_{\Sigma} \\
 0 \rightarrow H \leftarrow m + n &:= \langle\langle \Gamma \left[- \boxed{h} - \right]_{\Sigma}^{\mathbf{T}} \rangle\rangle_{\Sigma} = \langle\langle \overbrace{\boxed{h}}^{\text{---}} \quad r \quad c \rangle\rangle_{\Sigma}
 \end{aligned}$$

Moving into $\mathbf{S}_{\Sigma} + \mathbf{Frob}$, we have that  ; so by

functoriality $\langle\langle \Gamma \left[- \boxed{g} - \right]_{\Sigma}^{\mathbf{T}} \rangle\rangle_{\Sigma} = \langle\langle \overbrace{\boxed{g}}^{\text{---}} \rangle\rangle_{\Sigma} ; \langle\langle \overbrace{\boxed{c}}^{\text{---}} \rangle\rangle_{\Sigma}$, i.e. $0 \rightarrow G \leftarrow m + n = 0 \rightarrow L \leftarrow i + j ; i + j \rightarrow C \leftarrow m + n$.

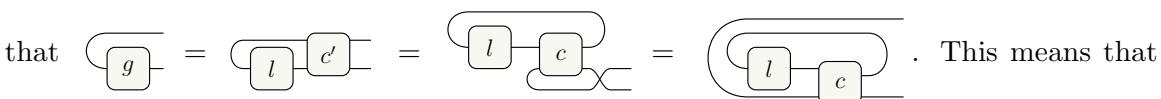
Cospan composition is by pushout, so $L \rightarrow G \leftarrow C$ is a pushout. Using the same reasoning, $R \rightarrow G \leftarrow C$ is also a pushout; this gives us the DPO diagram. All that remains is to check that the aforementioned pushouts are traced boundary complements; this follows by Lemma 5.23 as $\langle\langle \boxed{c} \rangle\rangle_{\Sigma}$ is partial monogamous.

Now for the (\Leftarrow) direction: we assume we have a traced DPO rewrite, so there exist cospans $0 \rightarrow L \leftarrow i + j, 0 \rightarrow R \leftarrow i + j, i + j \rightarrow C \leftarrow m + n$ as above such that the DPO diagram commutes and $i + j \rightarrow C \rightarrow G$ is a traced boundary complement. We must show

that $- \boxed{g} - = \boxed{l} \boxed{c}$ and $- \boxed{h} - = \boxed{r} \boxed{c}$.

We have that $0 \rightarrow G \leftarrow m + n = 0 \rightarrow L \leftarrow i + j ; i + j \xrightarrow{[c_1, c_2]} C \xleftarrow{[d_1, d_2]} m + n$ as cospan composition is by pushout. Let $j \boxed{c'} m \boxed{n}$ be the term in $\mathbf{S}_{\Sigma} + \mathbf{Frob}$ such that $\langle\langle \boxed{c'} \rangle\rangle_{\Sigma} = i + j \xrightarrow{[c_1, c_2]} C \xleftarrow{[d_1, d_2]} m + n$, which exists as $\langle\langle - \rangle\rangle_{\Sigma}$ is full.

The cospan $j + m \xrightarrow{[c_2, d_1]} C \xleftarrow{[c_1, d_2]} i + n$ is partial monogamous because $i + j \rightarrow C \rightarrow G$ is a traced boundary complement. Let $m \boxed{c} n$ be the term in $\mathbf{S}_{\Sigma} + \mathbf{Frob}$ such that $\langle\langle \boxed{c} \rangle\rangle_{\Sigma} = j + m \xrightarrow{[c_2, d_1]} C \xleftarrow{[c_1, d_2]} i + n$; by Lemma 5.23, we have $\langle\langle \overbrace{\boxed{c}}^{\text{---}} \rangle\rangle_{\Sigma} = i + j \xrightarrow{[c_1, c_2]} C \xleftarrow{[d_1, d_2]} m + n$.

So we have that $\langle\langle \Gamma - \boxed{g} - \rangle\rangle_{\Sigma} = \langle\langle \Gamma - \boxed{l} - \rangle\rangle_{\Sigma} ; \langle\langle \boxed{c} \rangle\rangle_{\Sigma}$; by fullness we derive that .

$\Gamma - \boxed{g} - = \boxed{l} \boxed{c} \boxed{c'}$ so ‘unfolding’ the interface gives us $- \boxed{g} - = \boxed{l} \boxed{c}$.

Since $\langle\langle \boxed{c} \rangle\rangle_{\Sigma}$ is partial monogamous, \boxed{c} is in \mathbf{T}_{Σ} . As the trace in \mathbf{T}_{Σ} is the

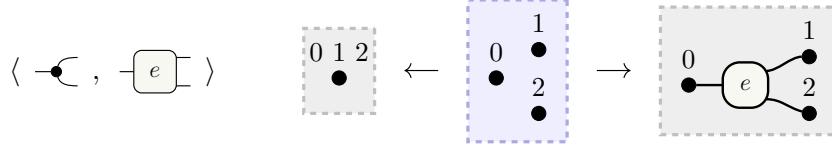
canonical trace, the entire term is in \mathbf{T}_Σ , completing the proof. The same procedure holds for rewriting from the other direction. \square

5.2. Rewriting with traced comonoid structure. It is straightforward to adapt the results for rewriting with partial monogamous cospans to those for rewriting with partial left-monogamous cospans. In this case, the definition of traced boundary complement is too restrictive, so must be weakened to permit more valid rewrites.

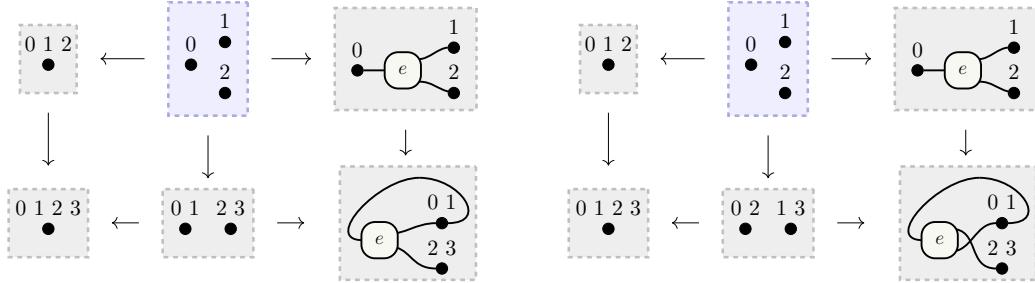
Definition 5.25 (Traced left-boundary complement). For partial left-monogamous cospans $i \xrightarrow{a_1} L \xleftarrow{a_2} j$ and $n \xrightarrow{b_1} G \xleftarrow{b_2} m \in \mathbf{Hyp}_\Sigma$, a pushout complement as in Definition 5.16 is called a *traced left-boundary complement* if c_2 is mono and $j + m \xrightarrow{[c_2, d_1]} C \xleftarrow{[c_1, d_2]} i + n$ is a partial left-monogamous cospan.

Definition 5.26 (Traced comonoid DPO). For morphisms $G \leftarrow m + n$ and $H \leftarrow m + n$ in \mathbf{Hyp}_Σ , there is a traced comonoid rewrite $G \rightsquigarrow_{\mathcal{R}} H$ if there exists a rule $L \leftarrow i + j \rightarrow G \in \mathcal{R}$ and cospan $i + j \rightarrow C \leftarrow n + m \in \mathbf{Hyp}_\Sigma$ such that the diagram in Definition 5.3 commutes and $i + j \rightarrow C \rightarrow G$ is a traced left-boundary complement.

Example 5.27. As with traced DPO, there may be multiple valid traced comonoid DPO rewrites for a given rule and instance in a larger graph. Consider the following rule and its interpretation.



Two valid rewrites are as follows:



The first rewrite is the ‘obvious’ one, but the second also holds by cocommutativity:

$$\text{Diagram showing two equivalent configurations of curved arrows and a box labeled 'e'}$$

To show that traced comonoid rewriting is sound and complete with respect to traced comonoid term rewriting we follow the same procedure as for the traced setting.

Lemma 5.28 (Traced comonoid decomposition). *Given partial left-monogamous cospans $m \xrightarrow{d_1} G \xleftarrow{d_2} n$ and $i \xrightarrow{a_1} L \xleftarrow{a_2} j$, along with a morphism $L \xrightarrow{f} G$ such that $i + j \rightarrow L \rightarrow G$*

satisfies the no-dangling and no-identification conditions, then there exists a partial left-monogamous cospan $j + m \xrightarrow{[c_2, d_1]} C \xleftarrow{[c_1, d_2]} i + n$ such that $m \rightarrow G \leftarrow n$ can be factored as

$$\text{Tr}^i \left(\begin{array}{c} i \xrightarrow{a_1} L \xleftarrow{a_2} j \\ \otimes \\ m \rightarrow m \leftarrow m \end{array} ; j + m \xrightarrow{[c_2, d_1]} C \xleftarrow{[c_1, d_2]} i + n \right)$$

where $j + m \xrightarrow{c_1, d_2} C \xleftarrow{c_2, d_1} i + n$ is a traced left-boundary complement.

Proof. As Lemma 5.22, but with partial left-monogamous cospans. \square

Traced comonoid decomposition is then used in exactly the same role as traced decomposition in the previous section.

Theorem 5.29. Let \mathcal{R} be a rewriting system on $\mathbf{T}_\Sigma + \mathbf{CComon}$. Then,

$$\boxed{g} \Rightarrow_{\mathcal{R}} \boxed{h} \quad \text{if and only if} \quad \langle\!\langle \Gamma \left[\boxed{g} \right]_\Sigma \rangle\!\rangle_\Sigma \rightsquigarrow \langle\!\langle \Gamma \left[\boxed{h} \right]_\Sigma \rangle\!\rangle_\Sigma.$$

Proof. As Theorem 5.24, but with traced left-boundary complements and traced comonoid decomposition (Lemma 5.28). \square

This means that not only can we perform rewriting on traced terms modulo some equational theory using partial monogamous cospans and traced boundary complements; we can do the same for traced comonoid terms using partial left-monogamous cospans and traced left-boundary complements.

6. CASE STUDIES

6.1. Cartesian structure. One important class of categories with a traced comonoid structure are *traced Cartesian*, or *dataflow*, categories [CS90, Has97]. These categories are interesting because any traced Cartesian category has a fixpoint operator [Has97, Thm. 3.1].

Definition 6.1 (Cartesian category [Fox76]). A monoidal category is *Cartesian* if its tensor is given by the Cartesian product.

As a result of this, the unit is a terminal object in any Cartesian category, and any object has a comonoid structure. Cartesian categories are settings in which morphisms can be *copied* and *discarded*. These two operations are more clearly illustrated when viewed through the lens of a monoidal theory.

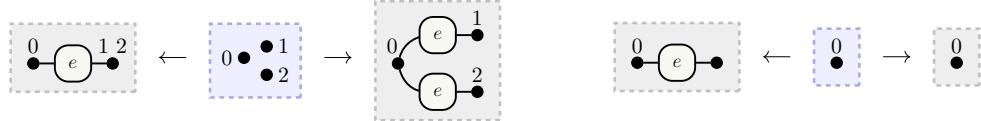
Definition 6.2. For a given PROP \mathbf{T}_{Σ_C} with a comonoid structure, the traced SMT $(\Sigma_{\mathbf{Cart}_C}, \mathcal{E}_{\mathbf{Cart}_C})$ is defined with $\Sigma_{\mathbf{Cart}_C} := \Sigma_C$ and $\mathcal{E}_{\mathbf{Cart}_C}$ as the equations in Figure 14.

The hypergraph interpretations of these rules for a generator e are shown in Figure 15.

Remark 6.3. The combination of Cartesian equations with the underlying compact closed structure of $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ may prompt alarm bells, as a compact closed category in which the tensor is the Cartesian product is trivial. However, it is important to note that $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is *not* subject to these equations: it is only a setting for performing graph rewrites.

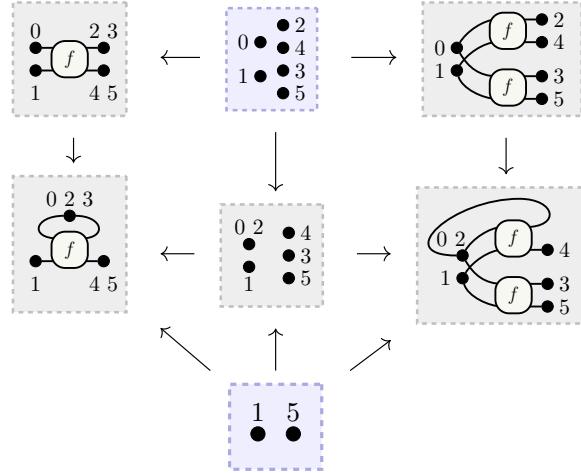
Reasoning about fixpoints can be performed using the *unfolding* rule, which holds in any traced Cartesian category.

$$m - \boxed{f} \bullet_n = m \bullet_n \begin{array}{c} f \\ \swarrow \searrow \end{array} \quad m - \boxed{f} \bullet = \bullet$$

Figure 14: Equations of the monoidal theory $\mathbf{Cart}_{\mathcal{C}}$, for an arbitrary generator f Figure 15: Interpretations of equations in $\mathbf{Cart}_{\mathcal{C}}$ for an arbitrary generator e .

$$m - \begin{array}{c} x \\ \swarrow \searrow \end{array} \boxed{f} - n = m - \begin{array}{c} x \\ \swarrow \searrow \end{array} \boxed{f} - \bullet_n = m - \begin{array}{c} x \\ \swarrow \searrow \end{array} \boxed{f} - \bullet_n = m - \begin{array}{c} x \\ \swarrow \searrow \end{array} \boxed{f} - n$$

In the syntactic setting, this requires the application of multiple equations: the two counitality equations followed by the copy equation and optionally some axioms of STMCs for housekeeping. However, if we interpret this in the hypergraph setting, the comonoid equations are absorbed into the notation so only one rewrite rule needs to be applied.



The dual notion of traced *cocartesian* categories [Bai76] are also important in computer science: a trace in a traced cocartesian category corresponds to *iteration* in the context of *control flow*. The details of this section could also be applied to the cocartesian case by flipping all the directions and working with partial *right-monogamous* cospans.

However, attempting to combine the product and coproduct approaches for settings with a *biproduct* would simply yield the category $\mathbf{Csp}_D(\mathbf{Hyp}_{\Sigma})$, a hypergraph category (Proposition 2.22) subject to the Frobenius equations in Figure 4. A category with biproducts is not necessarily subject to such equations, so this would not be a suitable approach.

6.2. Digital circuits. As mentioned above, traced Cartesian categories are useful for reasoning in settings with fixpoint operators. One such setting is that of *sequential digital circuits* built from primitive logic gates: in [GKS24], such circuits are modelled as morphisms in a STMC. Here, the trace models a feedback loop, and the comonoid structure represents forking wires. We are interested in using graph rewriting to implement an *operational semantics* for sequential circuits.

Definition 6.4. Let \mathbf{V} be the set $\{\perp, t, f, \top\}$ with a lattice structure defined by $t \sqcup f = \top$ and $t \sqcap f = \perp$.

The elements of \mathbf{V} are *values* that flow through wires in a circuit. The t and f values are the traditional true and false values, the \perp value represents *no information* (a *disconnected wire*) and the \top value represents *both* true and false at once (a *short circuit*).

Notation 6.5. For $m \in \mathbb{N}$, we write elements of \mathbf{V}^m with an overline, e.g. $\bar{v} \in \mathbf{V}^3 := \text{tft}$.

Values are one part of the signature for sequential circuits.

Definition 6.6 (Gate-level signature). Let the set Σ_{CCirc} of *combinational gate-level circuit generators* be defined as $\{\text{AND}, \text{OR}, \text{NOT}, \text{Fork}, \text{Join}, \text{Delay}, \text{Reset}\}$ and the set Σ_{SCirc} of *sequential gate-level circuit generators* be defined as $\{\text{t-}, \text{f-}, \text{T-}, \text{-D}\}$.

The *combinational* generators are components that model functions; they are respectively AND, OR and NOT gates along with *structural* constructs for introducing, forking, joining and eliminating wires. Combinational circuits are drawn with light blue backgrounds $m-\boxed{f}-n$; these circuits *always* produce the same outputs given the same inputs.

Notation 6.7. The structural generators are defined for wires of width 1, but versions for arbitrary widths can be easily derived by axioms of STMCs. In diagrams, these are drawn the same as their single-bit counterparts: $\boxed{\bullet}-n$, $n-\boxed{\bullet}-n$, $n-\boxed{\circlearrowleft}-n$ and $n-\boxed{\circlearrowright}$.

Sequential generators are components that model state: *instantaneous values* and a delay of one unit of time. Sequential circuits are drawn with green backgrounds $m-\boxed{f}-n$; these are circuits where the outputs may differ depending on past states.

The intended interpretation of the value generators is that they produce the relevant value on the first cycle of execution, followed by the disconnected \perp value after that. Subsequently, there is no sequential \perp value generator; it is instead modelled by the combinational $\boxed{\bullet}-n$, as it will *always* emit \perp .

The delay component is the opposite: initially it outputs \perp but on subsequent cycles it will output the input to the previous cycle. What a ‘cycle’ is can differ depending on application; the most obvious interpretation is a D flipflop in a clocked circuit, but it could also model the inertial delay on wires.

Notation 6.8. We write $\boxed{v}-$ for an arbitrary value $v \in \mathbf{V}$; note that this could also include \perp . For $\bar{v} \in \mathbf{V}^m$ we collapse multiple value and delay generators into one as $\boxed{\bar{v}}-m$ and

$m-\boxed{\text{D}}-m$, and write $m-\boxed{\bar{v}}-m := m-\boxed{\text{D}}-\boxed{\bar{v}}-m$ for a *register*.

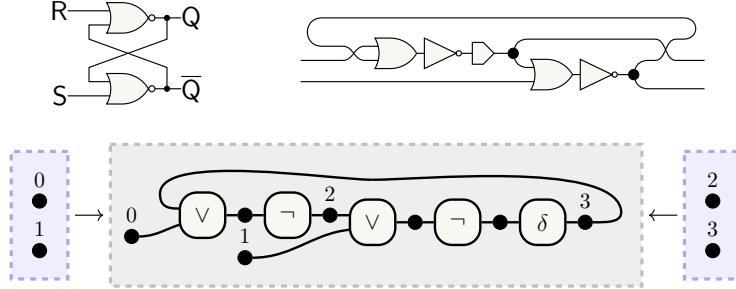


Figure 16: An SR NOR latch, a possible construction of said latch in \mathbf{SCirc}_Σ , and the hypergraph interpretation of this construction

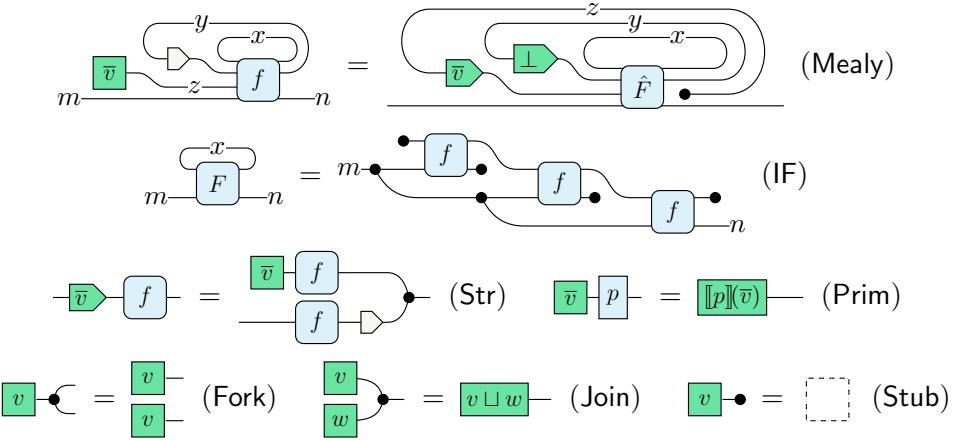


Figure 17: Equations for reducing sequential circuits

Example 6.9 (SR Latch [GKS24]). An example of a sequential circuit component is an *SR NOR latch*, illustrated in Figure 16; a NOR gate is constructed as $\begin{array}{c} \text{---} \\ \text{g} \end{array} := \begin{array}{c} \text{---} \\ \text{g} \end{array} \rightarrow \begin{array}{c} \text{---} \\ \text{g} \end{array}$.

SR latches are used to hold state: when the *S* input ('set') is pulsed true then the *Q* output is held at true until the *R* input ('reset') is true. The state is held because there are delays in the gates and the wires; one of the feedback loops between the two NOR gates will 'win'. In \mathbf{SCirc}_Σ this is modelled by using a different number of delay generators on the wires between the top and the bottom of the latch, as shown in Figure 16. The interpretation of this implementation as a cospan of hypergraphs is also depicted in Figure 16, where \vee is the interpretation of the OR gate, \neg is the interpretation of the NOT gate, and δ is the interpretation of the delay.

An operational semantics for sequential circuits is defined in terms of equations showing how a circuit transforms an input value into an output value. On sticking point is *non-delay-guarded feedback*; this is tackled by applying the Kleene fixpoint theorem and *iterating* a circuit multiple times until a fixpoint is reached.

Definition 6.10. Let the set $\mathcal{E}_{\mathbf{SCirc}}$ of *gate-level circuit equations* be those listed in Figure 17, where $\begin{array}{c} \text{---} \\ \text{g} \end{array}$ is one of the logic gates and $\llbracket \text{---} \rrbracket$ maps them to the corresponding truth table.

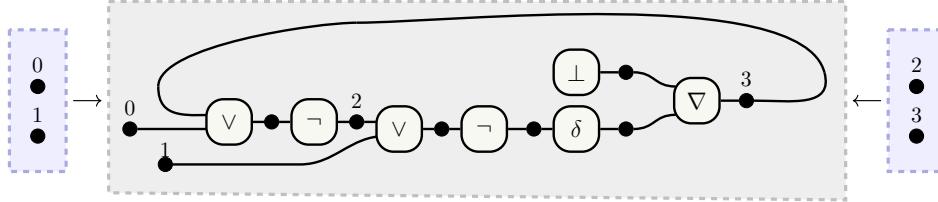


Figure 18: The result of rewriting the SR NOR latch using the Mealy rule

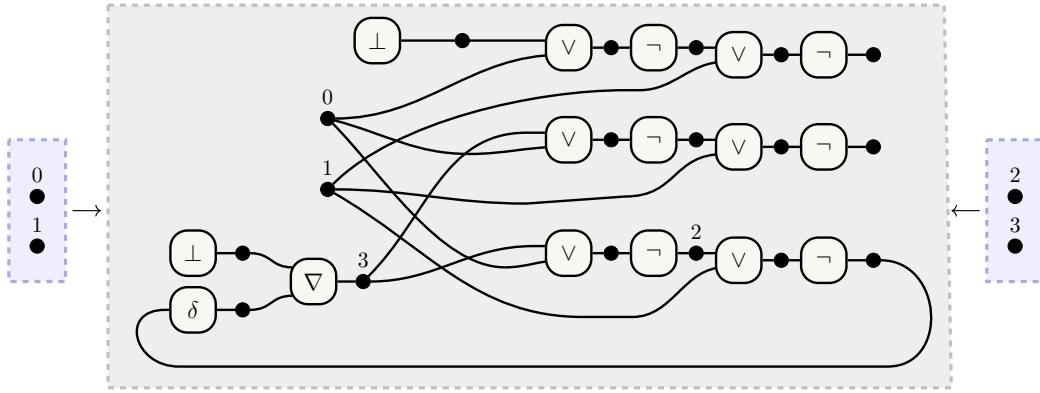


Figure 19: The result of rewriting Figure 18 using the IF rule

The aim of the operational semantics is to transform a circuit of the form $m \xrightarrow{\bar{v}} f \xrightarrow{} n$ into one of the form $m \xrightarrow{\bar{w}} G \xrightarrow{\bar{w}} n$; this shows how a circuit processes inputs, producing a new internal state and outputs.

First the circuit $m \xrightarrow{\bar{f}} n$ must be assembled into a special form known as ‘Mealy form’ by applying the Mealy rule followed by the IF rule. Rewriting modulo comonoid structure is useful here since this procedure can create multiple forks and stubs, which can clutter up a term; here the connectivity is much clearer.

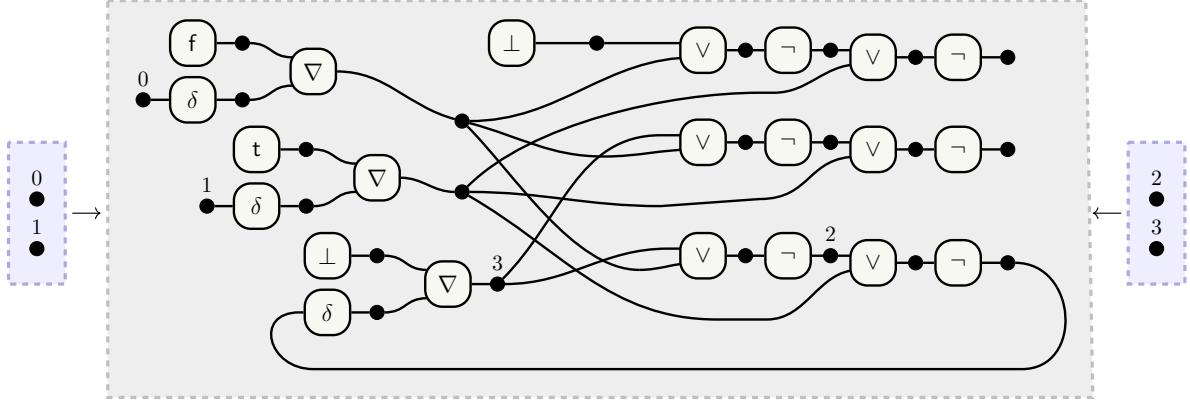
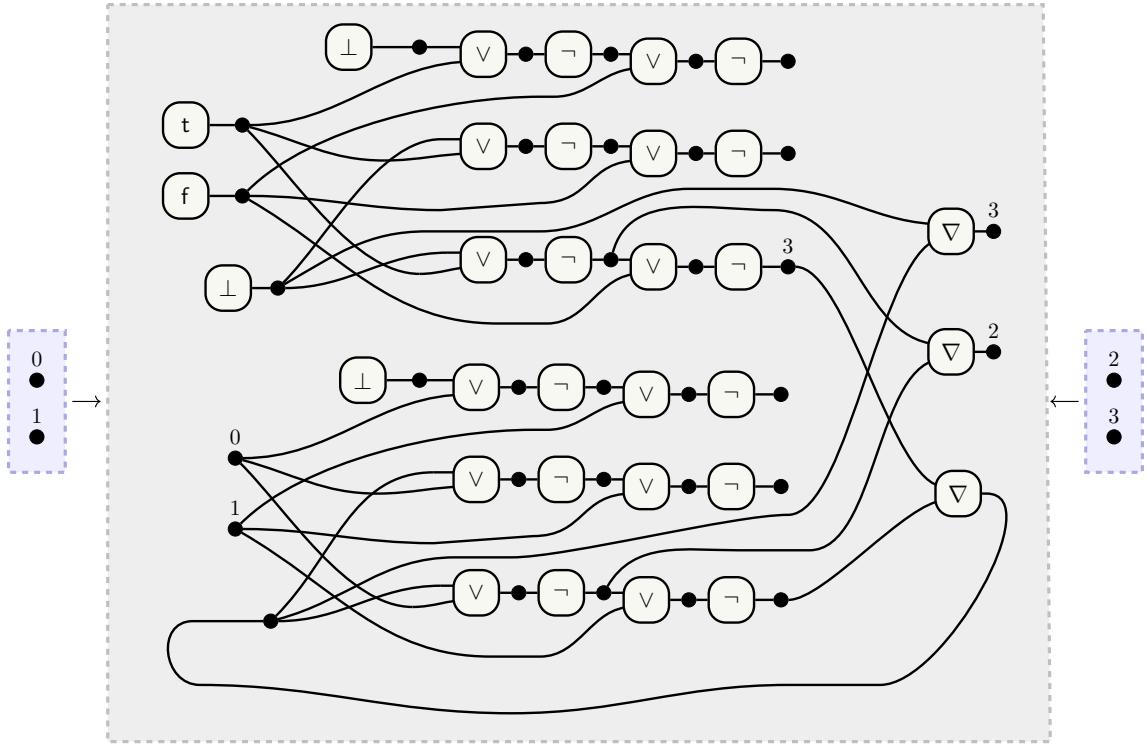
Once the circuit is in this form, it has a ‘combinational core’ traced by delay-guarded feedback. Values can now be applied to it and the Str rule used to copy the core into a ‘now’ copy and a ‘later’ copy. The ‘now’ copy computes the outputs and the transition for the current cycle of execution. To determine the exact values, the Fork, Join, Stub and Prim equations can be applied to reduce the core to values.

Example 6.11. The rewriting procedure discussed above is applied to the SR NOR latch in Figures 18-23.

7. CONCLUSION, RELATED AND FURTHER WORK

We have shown how previous work on rewriting string diagrams modulo Frobenius [BGK⁺22a] and symmetric monoidal [BGK⁺22b] structure using hypergraphs can also be adapted for rewriting modulo traced comonoid structure using a setting between the two.

Graphical languages for traced categories have seen many applications, such as to illustrate cyclic lambda calculi [Has97], or to reason graphically about programs [SJ99]. The

Figure 20: Applying Figure 19 to the input values ft Figure 21: Applying Str to Figure 20

presentation of traced categories as *string diagrams* has existed since the 90s [JS91, JSV96]; a soundness and completeness theorem for traced string diagrams, folklore for many years but only proven for certain signatures [Sel11], was finally shown in [Kis14]. Combinatorial languages predate even this, having existed since at least the 80s in the guise of *flowchart schemes* [St90, CS90, CS94]. These diagrams have also been used to show the completeness of finite dimensional vector spaces [HHP08] with respect to traced categories and, when equipped with a dagger, Hilbert spaces [Sel12].

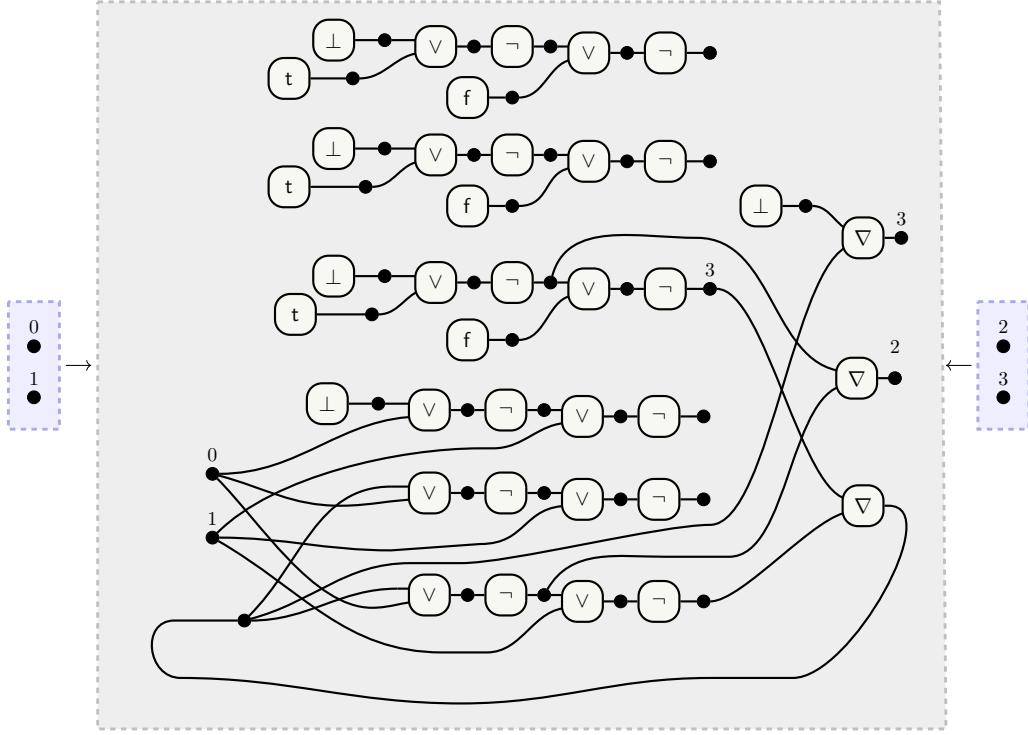
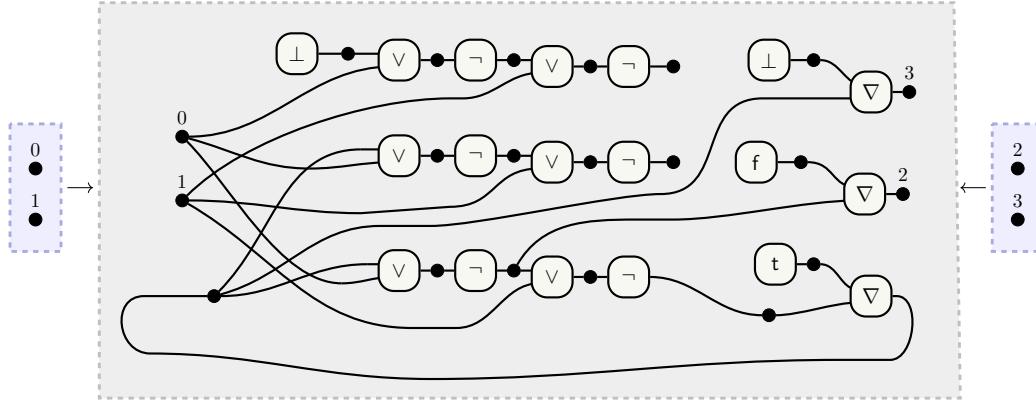


Figure 22: Applying Fork to Figure 21

Figure 23: Applying Fork, Prim_I and Stub to Figure 22

We are not just concerned with diagrammatic languages as a standalone concept: we are interested in performing *graph rewriting* with them to reason about monoidal theories. This has been studied in the context of traced categories before using *string graphs* [Kis12, DK13]. We have instead opted to use the *hypergraph* framework of [BGK⁺22a, BGK⁺22b, BGK⁺22c] instead, as it allows rewriting modulo *yanking*, is more extensible for rewriting modulo comonoid structure, and one does not need to awkwardly reason modulo wire homeomorphisms.

As mentioned during the case studies, there are still elements of the rewriting framework that are somewhat informal. One such issue involves defining rewrite spans for arbitrary subgraphs: this is hard to do at a general level because the edges must be concretely specified in DPO rewriting. However, if we performed rewriting with *hierarchical hypergraphs* [AGSZ23], in which edges can have hypergraphs as labels, we could ‘compress’ the subgraph into a single edge that can be rewritten: this is future work.

In regular PROP notation, wires are annotated with numbers in order to avoid drawing multiple wires in parallel: when interpreted as hypergraphs a node is created for each wire, and simple diagrams can quickly get very large. The results of [BGK⁺22b] also extend to the multi-sorted case, in which nodes are labelled in addition to wires. We could use this in combination with the *strictifiers* of [WGZ23]: these are additional generators for transforming buses of wires into thinner or thicker ones. This could drastically reduce the number of elements in a hypergraph, which is ideal from a computational point of view. Work has already begun on implementing the rewriting system for digital circuits using these techniques.

Acknowledgements. Thanks to Chris Barrett for comments on earlier versions of this paper, and to the anonymous reviewers for their helpful insights.

REFERENCES

- [AC04] Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.*, pages 415–425, July 2004. doi:[10.1109/LICS.2004.1319636](https://doi.org/10.1109/LICS.2004.1319636).
- [AGSZ23] Mario Alvarez-Picallo, Dan Ghica, David Sprunger, and Fabio Zanasi. Functorial String Diagrams for Reverse-Mode Automatic Differentiation. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual Conference on Computer Science Logic (CSL 2023)*, volume 252 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:20, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:[10.4230/LIPIcs.CSL.2023.6](https://doi.org/10.4230/LIPIcs.CSL.2023.6).
- [Bai76] E. S. Bainbridge. Feedback and generalized logic. *Information and Control*, 31(1):75–96, May 1976. doi:[10.1016/S0019-9958\(76\)90390-9](https://doi.org/10.1016/S0019-9958(76)90390-9).
- [BE15] John C. Baez and Jason Erbele. Categories in Control. *Theory and Applications of Categories*, 30(24):836–881, May 2015. doi:[10.48550/arXiv.1405.6881](https://doi.org/10.48550/arXiv.1405.6881).
- [BGK⁺17] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Paweł Sobociński, and Fabio Zanasi. Confluence of Graph Rewriting with Interfaces. In Hongseok Yang, editor, *Programming Languages and Systems*, Lecture Notes in Computer Science, pages 141–169, Berlin, Heidelberg, 2017. Springer. doi:[10.1007/978-3-662-54434-1_6](https://doi.org/10.1007/978-3-662-54434-1_6).
- [BGK⁺22a] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Paweł Sobociński, and Fabio Zanasi. String Diagram Rewrite Theory I: Rewriting with Frobenius Structure. *Journal of the ACM*, 69(2):14:1–14:58, March 2022. doi:[10.1145/3502719](https://doi.org/10.1145/3502719).
- [BGK⁺22b] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Paweł Sobociński, and Fabio Zanasi. String diagram rewrite theory II: Rewriting with symmetric monoidal structure. *Mathematical Structures in Computer Science*, 32(4):511–541, April 2022. doi:[10.1017/S0960129522000317](https://doi.org/10.1017/S0960129522000317).
- [BGK⁺22c] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Paweł Sobociński, and Fabio Zanasi. String diagram rewrite theory III: Confluence with and without Frobenius. *Mathematical Structures in Computer Science*, 32(7):1–41, June 2022. doi:[10.1017/S0960129522000123](https://doi.org/10.1017/S0960129522000123).
- [BP22] Guillaume Boisseau and Robin Piedeleu. Graphical Piecewise-Linear Algebra. In *Foundations of Software Science and Computation Structures: 25th International Conference, FOSSACS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2–7, 2022, Proceedings*, pages 101–119, Berlin, Heidelberg, April 2022. Springer-Verlag. doi:[10.1007/978-3-030-99253-8_6](https://doi.org/10.1007/978-3-030-99253-8_6).

- [BPSZ19] Filippo Bonchi, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. Graphical Affine Algebra. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, June 2019. doi:10.1109/LICS.2019.8785877.
- [BS22] Guillaume Boisseau and Paweł Sobociński. String Diagrammatic Electrical Circuit Theory. *Electronic Proceedings in Theoretical Computer Science*, 372:178–191, November 2022. arXiv: 2106.07763, doi:10.4204/EPTCS.372.13.
- [BSZ14] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. A Categorical Semantics of Signal Flow Graphs. In Paolo Baldan and Daniele Gorla, editors, *CONCUR 2014 – Concurrency Theory*, Lecture Notes in Computer Science, pages 435–450, Berlin, Heidelberg, 2014. Springer. doi: 10.1007/978-3-662-44584-6_30.
- [BSZ15] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. Full Abstraction for Signal Flow Graphs. *ACM SIGPLAN Notices*, 50(1):515–526, January 2015. doi:10.1145/2775051.2676993.
- [BSZ17] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. Interacting Hopf algebras. *Journal of Pure and Applied Algebra*, 221(1):144–184, January 2017. doi:10.1016/j.jpaa.2016.06.002.
- [CMR⁺97] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe. Algebraic approaches to graph transformation ? part i: Basic concepts and double pushout approach. In *Handbook of Graph Grammars and Computing by Graph Transformation*, pages 163–245. World Scientific, February 1997. doi:10.1142/9789812384720_0003.
- [CŞ90] Virgil Emil Căzănescu and Gheorghe Ștefănescu. Towards a New Algebraic Foundation of Flowchart Scheme Theory. *Fundamenta Informaticae*, 13(2):171–210, January 1990. doi:10.3233/FI-1990-13204.
- [CŞ94] Virgil Emil Căzănescu and Gheorghe Ștefănescu. Feedback, Iteration, and Repetition. In *Mathematical Aspects of Natural and Formal Languages*, volume Volume 43 of *World Scientific Series in Computer Science*, pages 43–61. World Scientific, October 1994. doi:10.1142/9789814447133_0003.
- [DK13] Lucas Dixon and Aleks Kissinger. Open-graphs and monoidal theories. *Mathematical Structures in Computer Science*, 23(2):308–359, April 2013. doi:10.1017/S0960129512000138.
- [EK76] Hartmut Ehrig and Hans-Jörg Kreowski. Parallelism of manipulations in multidimensional information structures. In Antoni Mazurkiewicz, editor, *Mathematical Foundations of Computer Science 1976*, Lecture Notes in Computer Science, pages 284–293, Berlin, Heidelberg, 1976. Springer. doi:10.1007/3-540-07854-1_188.
- [FL23] Tobias Fritz and Wendong Liang. Free gs-Monoidal Categories and Free Markov Categories. *Applied Categorical Structures*, 31(2):21, April 2023. doi:10.1007/s10485-023-09717-0.
- [Fox76] Thomas Fox. Coalgebras and cartesian categories. *Communications in Algebra*, 4(7):665–667, January 1976. doi:10.1080/00927877608822127.
- [FS19] Brendan Fong and David I. Spivak. Hypergraph categories. *Journal of Pure and Applied Algebra*, 223(11):4746–4777, November 2019. doi:10.1016/j.jpaa.2019.02.014.
- [FSR16] Brendan Fong, Paweł Sobociński, and Paolo Rapisarda. A categorical approach to open and interconnected dynamical systems. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS ’16, pages 495–504, New York, NY, USA, July 2016. Association for Computing Machinery. doi:10.1145/2933575.2934556.
- [GK23] Dan R. Ghica and George Kaye. Rewriting Modulo Traced Comonoid Structure. In Marco Gaboardi and Femke van Raamsdonk, editors, *8th International Conference on Formal Structures for Computation and Deduction (FSCD 2023)*, volume 260 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:21, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSCD.2023.14.
- [GKS24] Dan R. Ghica, George Kaye, and David Sprunger. A Fully Compositional Theory of Sequential Digital Circuits: Denotational, Operational and Algebraic Semantics. (arXiv:2201.10456), January 2024. doi:10.48550/arXiv.2201.10456.
- [Has97] Masahito Hasegawa. Recursion from cyclic sharing: Traced monoidal categories and models of cyclic lambda calculi. In Philippe de Groote and J. Roger Hindley, editors, *Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, pages 196–213, Berlin, Heidelberg, 1997. Springer. doi:10.1007/3-540-62688-3_37.

- [Has03] Masahito Hasegawa. The Uniformity Principle on Traced Monoidal Categories. *Electronic Notes in Theoretical Computer Science*, 69:137–155, February 2003. doi:10.1016/S1571-0661(04)80563-2.
- [HHP08] Masahito Hasegawa, Martin Hofmann, and Gordon Plotkin. Finite Dimensional Vector Spaces Are Complete for Traced Symmetric Monoidal Categories. In Arnon Avron, Nachum Dershowitz, and Alexander Rabinovich, editors, *Pillars of Computer Science: Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, Lecture Notes in Computer Science, pages 367–385. Springer, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-78127-1_20.
- [HJKS11] Marvin Heumüller, Salil Joshi, Barbara König, and Jan Stückrath. Construction of Pushout Complements in the Category of Hypergraphs. *Electronic Communications of the EASST*, 39(0), September 2011. doi:10.14279/tuj.eceasst.39.647.
- [JS91] André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in Mathematics*, 88(1):55–112, 1991. doi:10.1016/0001-8708(91)90003-P.
- [JSV96] André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):447–468, April 1996. doi:10.1017/S0305004100074338.
- [Kis12] Aleks Kissinger. *Pictures of Processes: Automated Graph Rewriting for Monoidal Categories and Applications to Quantum Computing*. PhD thesis, University of Oxford, March 2012. arXiv:1203.0202, doi:10.48550/arXiv.1203.0202.
- [Kis14] Aleks Kissinger. Abstract Tensor Systems as Monoidal Categories. In Claudia Casadio, Bob Coecke, Michael Moortgat, and Philip Scott, editors, *Categories and Types in Logic, Language, and Physics: Essays Dedicated to Jim Lambek on the Occasion of His 90th Birthday*, Lecture Notes in Computer Science, pages 235–252. Springer, Berlin, Heidelberg, 2014. doi:10.1007/978-3-642-54789-8_13.
- [KL80] G. M. Kelly and M. L. Laplaza. Coherence for compact closed categories. *Journal of Pure and Applied Algebra*, 19:193–213, December 1980. doi:10.1016/0022-4049(80)90101-2.
- [Law63] F. William Lawvere. Functorial semantics of algebraic theories. *Proceedings of the National Academy of Sciences*, 50(5):869–872, November 1963. doi:10.1073/pnas.50.5.869.
- [LS04] Stephen Lack and Paweł Sobociński. Adhesive Categories. In Igor Walukiewicz, editor, *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science, pages 273–288. Springer, Berlin, Heidelberg, 2004. doi:10.1007/978-3-540-24727-2_20.
- [LS05] Stephen Lack and Paweł Sobociński. Adhesive and quasiadhesive categories. *RAIRO - Theoretical Informatics and Applications*, 39(3):511–545, July 2005. doi:10.1051/ita:2005028.
- [Mac65] Saunders MacLane. Categorical algebra. *Bulletin of the American Mathematical Society*, 71(1):40–106, 1965. doi:10.1090/S0002-9904-1965-11234-4.
- [MPZ23] Aleksandar Milosavljevic, Robin Piedeleu, and Fabio Zanasi. String Diagram Rewriting Modulo Commutative (Co)monoid Structure. (arXiv:2204.04274), March 2023. doi:10.48550/arXiv.2204.04274.
- [MS09] John MacDonald and Laura Scull. Amalgamations of Categories. *Canadian Mathematical Bulletin*, 52(2):273–284, June 2009. doi:10.4153/CMB-2009-030-5.
- [Pen71] Roger Penrose. Applications of negative dimensional tensors. In *Combinatorial Mathematics and Its Applications*, pages 221–244. Academic Press, 1971.
- [PZ21] Robin Piedeleu and Fabio Zanasi. A String Diagrammatic Axiomatisation of Finite-State Automata. *Foundations of Software Science and Computation Structures*, 12650:469–489, March 2021. doi:10.1007/978-3-030-71995-1_24.
- [RSW05] R. Rosebrugh, N. Sabadini, and R. F. C. Walters. Generic commutative separable algebras and cospans of graphs. *Theory and Applications of Categories*, 15:164–177, 2005.
- [Sel11] Peter Selinger. A Survey of Graphical Languages for Monoidal Categories. In Bob Coecke, editor, *New Structures for Physics*, Lecture Notes in Physics, pages 289–355. Springer, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-12821-9_4.
- [Sel12] Peter Selinger. Finite dimensional Hilbert spaces are complete for dagger compact closed categories. *Logical Methods in Computer Science*, 8(3), August 2012. doi:10.2168/LMCS-8(3:6)2012.

- [SJ99] Ralf Schweimeier and Alan Jeffrey. A Categorical and Graphical Treatment of Closure Conversion. *Electronic Notes in Theoretical Computer Science*, 20:481–511, January 1999. doi:10.1016/S1571-0661(04)80090-2.
- [WGZ23] Paul Wilson, Dan Ghica, and Fabio Zanasi. String Diagrams for Non-Strict Monoidal Categories. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual Conference on Computer Science Logic (CSL 2023)*, volume 252 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:19, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2023.37.
- [Zan15] Fabio Zanasi. *Interacting Hopf Algebras: The Theory of Linear Systems*. PhD thesis, University of Lyon, October 2015. doi:10.48550/arXiv.1805.03032. arXiv:1805.03032.
- [St90] Gheorghe Ștefănescu. Feedback Theories (A Calculus for Isomorphism Classes of Flowchart Schemes). *Romanian Journal of Pure and Applied Mathematics*, 35(1):73–79, 1990.