




Rewriting Modulo Traced Comonoid Structure

Dan R. Ghica   

University of Birmingham, UK

George Kaye   

University of Birmingham, UK

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.

2012 ACM Subject Classification Theory of computation → Equational logic and rewriting

Keywords and phrases symmetric traced monoidal categories, string diagrams, graph rewriting, comonoid structure, double pushout rewriting

Digital Object Identifier 10.4230/LIPIcs.FSCD.2023.14

Related Version *Full Version*: <https://arxiv.org/abs/2302.09631>

Funding This work was supported by the Engineering and Physical Sciences Research Council [grant EP/V001612/1].

Acknowledgements Thanks to Chris Barrett for helpful comments.

1 Introduction

String diagrams constitute a useful and elegant conceptual bridge between term rewriting and graph rewriting. We will not reprise here, for lack of space, the by now impressive body of theoretical and applied work for and with string diagrams but will take it as a given. The survey [30] is a suitable starting point into the literature.

The purpose of this paper is to support reasoning (via graph rewrite) in *traced categories with a comonoid structure* but without a monoid structure. Prior art on this topic exists [21, 10], but it is based on the framework of “framed point graphs” which requires rewriting modulo so called *wire homeomorphisms*. This style of rewriting is awkward and is increasingly considered as obsolete as compared to more recent work on rewriting with hypergraphs [4, 5, 6], possibly modulo *Frobenius* structure. The variation to just a (co)monoid structure (“half a Frobenius”) has been studied as well [13, 27].

The study of rewriting for traced categories with a comonoid structure is motivated by an important application, *dataflow categories* [8, 9, 15], which represent a categorical foundation for the semantics of digital circuits [14]. It is also technically challenging, as it falls in a gap between compact closed structures constructible via Frobenius and symmetric



© Dan R. Ghica and George Kaye;

licensed under Creative Commons License CC-BY 4.0

8th International Conference on Formal Structures for Computation and Deduction (FSCD 2023).

Editors: Marco Gaboardi and Femke van Raamsdonk; Article No. 14; pp. 14:1–14:21

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

14:2 Rewriting Modulo Traced Comonoid Structure

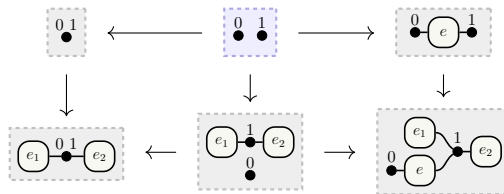
monoidal categories (without trace) so “off the shelf” solutions cannot be currently used. In fact the gap between the kind of semantic models which use an underlying compact closed structure and those which use a traced monoidal structure is significant: the former have a *relational* nature with subtle 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.

A key feature of compact closed categories is that the Cartesian product, if it exists, is degenerate and identified with the co-product. 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.

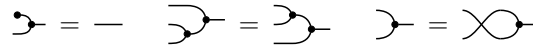
Contributions. This paper makes two distinct technical contributions. The first is to show that one subclass of cospans of hypergraphs (“partial monogamous”) are fully complete for traced terms (Corollary 35), and another class (“partial left-monogamous”) are fully complete for traced comonoid terms (Theorem 46). The challenge 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: $\text{Tr} \left(\begin{array}{|c|} \hline \square \\ \hline \end{array} \right) = \bigcirc$, 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, which graphically corresponds to “tracing a forking wire”: $\text{Tr} \left(\begin{array}{|c|} \hline \square \\ \hline \end{array} \right) = \begin{array}{c} \curvearrowright \\ \curvearrowleft \end{array}$.

The key step in performing double pushout (DPO) rewriting is identifying a *pushout complement*: 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, every pushout complement is valid [4] whereas when rewriting with symmetric monoidal structure exactly one pushout complement is valid [5]. For the traced case some pushout complements are valid and some are not. The second contribution is to characterise the valid pushout complements as “traced boundary complements” (Definition 58).

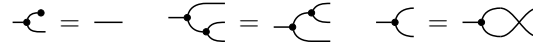
This is best illustrated with an example in which there is a pushout complement that is valid in a Frobenius setting because it uses the monoid structure, but it is not valid neither in a traced, nor even in a traced comonoid setting. Imagine we have a rule $\langle \text{---}, \text{---} \rangle$ and a term $\begin{array}{|c|} \hline \square \\ \hline \end{array}$, and rewrite it as follows.



This corresponds to the term rewrite $\begin{array}{|c|} \hline \square \\ \hline \end{array} = \begin{array}{c} \square \\ \curvearrowright \\ \curvearrowleft \end{array} = \begin{array}{c} \square \\ \curvearrowright \\ \square \\ \curvearrowleft \end{array}$, 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 [5] is too restrictive as it enforces that any matching must be mono: this prevents matchings such as $\begin{array}{|c|} \hline \square \\ \hline \end{array}$ in $\begin{array}{c} \curvearrowright \\ \square \\ \curvearrowleft \end{array}$. 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 (Theorems 65 and 69).



■ **Figure 1** Equations $\mathcal{E}_{\text{CMon}}$ of a commutative monoid.



■ **Figure 2** Equations $\mathcal{E}_{\text{CComon}}$ of a commutative comonoid.

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* [26] (“categories of *PRO*ducts and *PER*mutations”), which have natural numbers as objects and addition as tensor product.

► **Definition 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} . We write $\mathbf{S} := \mathbf{S}_\emptyset$ for the SMC with terms constructed solely from identities and symmetries.

► **Remark 2.** One can also define a *multi-sorted* SMT, in which wires can be of multiple colours. For brevity, we will only consider the single-sorted case, but the results generalise easily using the results of [4, 5].

While one could reason in \mathbf{S}_Σ using the one-dimensional categorical term language, it is more intuitive to reason with *string diagrams* [19, 30], 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 $m\text{-}\boxed{\phi}\text{-}n$, the identity id_n as $n\text{-}\boxed{\text{id}}\text{-}n$, and the symmetry $\sigma_{m,n}$ as $m\text{-}\boxed{\sigma}\text{-}n$. Composite terms will be illustrated as wider boxes $m\text{-}\boxed{f}\text{-}n$ to distinguish them from generators: then (diagrammatic order) composition $m\text{-}\boxed{f}\text{-}n \ ; \ n\text{-}\boxed{g}\text{-}p$ is defined as horizontal juxtaposition $m\text{-}\boxed{f}\text{-}\boxed{g}\text{-}p$ and tensor $m\text{-}\boxed{f}\text{-}n \otimes p\text{-}\boxed{g}\text{-}q$ as vertical juxtaposition $m\text{-}\boxed{f}\text{-}n \text{ over } p\text{-}\boxed{g}\text{-}q$.

The graphical notation clearly illustrates 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.

► **Example 3.** The monoidal theory of *special commutative Frobenius algebras* is defined as $(\Sigma_{\text{Frob}}, \mathcal{E}_{\text{Frob}})$ where $\Sigma_{\text{Frob}} := \{ \boxed{\mu}, \boxed{\eta}, \boxed{\nu}, \boxed{\epsilon} \}$ and the equations of $\mathcal{E}_{\text{Frob}}$ are listed in Figures 1–3. We write $\mathbf{Frob} := \mathbf{S}_{\Sigma_{\text{Frob}}, \mathcal{E}_{\text{Frob}}}$.

Reasoning equationally using string diagrams is certainly attractive as a pen-and-paper method, but for larger systems it quickly becomes intractible 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; instead, we require a combinatorial representation. Fortunately, this has been well studied recently, first with *string graphs* [10, 21] and later with *hypergraphs* [4, 5, 6], a generalisation of regular graphs in which edges can be the source or target of an arbitrary number of vertices. In this paper we are concerned with the latter.

14:4 Rewriting Modulo Traced Comonoid Structure

■ **Figure 3** Equations $\mathcal{E}_{\text{Frob}}$ of a *special commutative Frobenius algebra*, in addition to those in Figures 1 and 2.

Hypergraphs are formally defined as objects in a functor category.

► **Definition 4** (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 \mathbf{Hyp} maps \star to a set of vertices, and each (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. 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. Hypergraph morphisms can be used to *label* hypergraphs according to a signature.

► **Definition 5** (Slice category [25]). *For a category \mathbf{C} and an object $C \in \mathbf{C}$, the slice category \mathbf{C}/C is the category with objects the morphisms of \mathbf{C} with target C , and where a morphism $(f: X \rightarrow C) \rightarrow (f': X' \rightarrow C)$ is a morphism $g: X \rightarrow X' \in \mathbf{C}$ such that $f' \circ g = f$.*

► **Definition 6** (Hypergraph signature [4]). *For a given monoidal signature Σ , its corresponding hypergraph signature $\llbracket \Sigma \rrbracket$ is the hypergraph with a single vertex 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 $j < \text{cod}(\phi)$, $s_i(e_\phi) = t_j(e_\phi) = v$.*

► **Definition 7** (Labelled hypergraph [4]). *For a monoidal signature Σ , let the category \mathbf{Hyp}_Σ be defined as the slice category $\mathbf{Hyp}/\llbracket \Sigma \rrbracket$.*

While (labelled) hypergraphs may have dangling vertices, they do not have *interfaces* specifying the order of inputs and outputs. These can be provided using *cospans*.

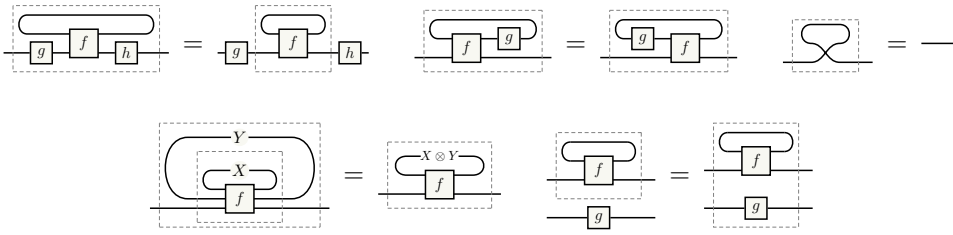
► **Definition 8** (Categories of cospans [5]). *For a finitely cocomplete category \mathbf{C} , a cospan from $X \rightarrow Y$ is a pair of arrows $X \rightarrow A \leftarrow Y$. A cospan morphism $(X \xrightarrow{f} A \xleftarrow{g} Y) \rightarrow (X \xrightarrow{h} B \xleftarrow{k} Y)$ is a morphism $\alpha: A \rightarrow B \in \mathbf{C}$ such that $\alpha \circ f = h$ and $\alpha \circ g = k$.*

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. 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} with unit 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 vertices in the graph at the apex.

► **Definition 9** (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 a main graph pick out the vertices in the interface: to assign an order to these vertices some more categorical machinery is required.



■ **Figure 4** Equations that hold in any *symmetric traced monoidal category*.

$$\begin{matrix} X \\ \curvearrowright \\ X \end{matrix} = X - X \quad \begin{matrix} \curvearrowleft \\ X \end{matrix} = X - X$$

■ **Figure 5** Equations that hold in any *compact closed category*.

► **Theorem 10** ([4], 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 $\mathbf{Csp}_F(\mathbf{C})$ whose arrows $m \rightarrow n$ are isomorphism classes of \mathbf{C} cospans $Fm \rightarrow C \leftarrow Fn$.

► **Definition 11.** Let \mathbb{F} be the PROP with morphisms $m \rightarrow n$ the functions between finite sets $[m] \rightarrow [n]$.

► **Definition 12** ([4]). Let $D: \mathbb{F} \rightarrow \mathbf{Hyp}_\Sigma$ be the faithful, coproduct-preserving functor that sends each object $m \in \mathbb{F}$ to the discrete hypergraph $m \in \mathbf{Hyp}_\Sigma$ and each morphism to the induced homomorphism of discrete hypergraphs.

From this we define the category $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ with objects *discrete cospans of hypergraphs*. Since the legs of each cospan are discrete hypergraphs containing some number of vertices, the objects of this category can be viewed as natural numbers, making this another PROP.

3 Hypergraphs for traced categories

We wish to use the hypergraph framework for a setting with a *trace*.

► **Definition 13** (Symmetric traced monoidal category [20, 16]). 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 listed in Figure 4.

In string diagrams, the trace is represented by joining output wires to input wires:

$$\text{Tr}_{A,B}^X \left(\begin{matrix} X \\ \boxed{f} \\ X \end{matrix} \right) \stackrel{\text{def}}{=} \begin{matrix} \boxed{f} \\ \text{---} \\ A \end{matrix} \text{---} B$$

Traced monoidal categories are not the only kind of category in which wires can “bend”.

► **Definition 14** (Compact closed category). 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 $\boxed{\text{---}}_X^{X^*}$ (“cup”) and the counit $\boxed{\text{---}}_{X^*}^X$ (“cap”) satisfying the equations of CCCs listed in Figure 5.

Dual objects are conventionally drawn as wires flowing the “other way”, but in this paper this is not necessary as all categories will be *self-dual*: any object X is isomorphic to X^* .

14:6 Rewriting Modulo Traced Comonoid Structure

$$\begin{array}{c} X \otimes Y \\ X \otimes Y \end{array} \rightarrow X \otimes Y = \begin{array}{c} X \\ Y \\ X \\ Y \end{array} \begin{array}{c} \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{array} X \\
 X \otimes Y \leftarrow \begin{array}{c} X \otimes Y \\ X \otimes Y \end{array} = \begin{array}{c} X \\ Y \\ X \\ Y \end{array} \begin{array}{c} \leftarrow \\ \leftarrow \\ \leftarrow \\ \leftarrow \end{array} X \\
 \bullet \leftarrow X \otimes Y = \begin{array}{c} \bullet \\ \bullet \end{array} \begin{array}{c} \leftarrow X \\ \leftarrow Y \end{array} \quad X \otimes Y \bullet = \begin{array}{c} X \\ Y \end{array} \begin{array}{c} \bullet \\ \bullet \end{array}$$

■ **Figure 6** Equations $\mathcal{E}_{\mathbf{Hyp}}$ of a hypergraph category, in addition to those in Figures 1–3.

► **Proposition 15** (Canonical trace ([20], Prop. 3.1)). *Any CCC has a trace $\text{Tr}_{A,B}^X \left(\begin{array}{c} X \\ A \end{array} \square \begin{array}{c} X \\ B \end{array} \right)$ called the canonical trace, defined for the self-dual case as*

$$\left(\begin{array}{c} \square \\ X \end{array} \otimes A \begin{array}{c} \square \\ A \end{array} \right) ; \left(X \begin{array}{c} \square \\ X \end{array} \otimes \begin{array}{c} X \\ A \end{array} \begin{array}{c} \square \\ B \end{array} \right) ; \left(\begin{array}{c} X \\ X \end{array} \begin{array}{c} \square \\ \square \end{array} \otimes B \begin{array}{c} \square \\ B \end{array} \right).$$

The category of interfaced hypergraphs as defined in the previous section already contains the structure necessary to define a trace.

► **Definition 16** (Hypergraph category [11]). *A hypergraph category is a symmetric monoidal category in which each object X has a special commutative Frobenius structure in the sense of Example 3 satisfying the equations in Figure 6.*

► **Proposition 17** ([28]). *Any hypergraph category is self-dual compact closed.*

Proof. The cup is constructed as $\begin{array}{c} \square \\ \bullet \end{array} ; \begin{array}{c} \square \\ \bullet \end{array}$ and the cap as $\begin{array}{c} \square \\ \bullet \end{array} ; \begin{array}{c} \square \\ \bullet \end{array}$. ◀

A generic “hypergraph category” should not be confused with the category of hypergraphs \mathbf{Hyp} , which is not itself a hypergraph category. However, the category of *cospan*s of hypergraphs is such a category.

► **Proposition 18** ([7, 4]). $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ 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:

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

► **Corollary 19.** $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is compact closed.

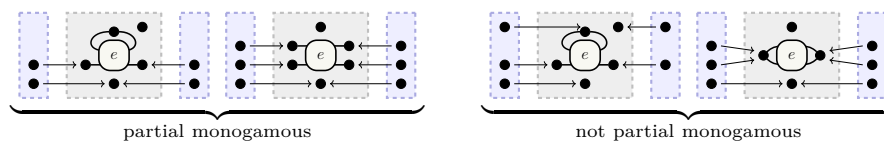
► **Corollary 20.** $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ has a trace.

This means that 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. However, this mapping is not *full*: there are terms in a CCC that are not terms in a STMC, such as $\begin{array}{c} \square \\ \oplus \end{array} \square$. So we must still restrict the cospan of hypergraphs in $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$ we use for *traced* terms.

3.1 Monogamy

In [3], it is shown that terms in a (non-traced) symmetric monoidal category are interpreted via a faithful functor into a sub-PROP of $\mathbf{Csp}_D(\mathbf{Hyp}_\Sigma)$. One condition on this sub-PROP is that all hypergraphs are *acyclic*. Clearly, to model trace this condition must be dropped.

However, there is also another condition known as *monogamy*: informally, this means that every vertex has exactly one “in” and “out” connection, be it to an edge or an interface. For the most part, this condition 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



■ **Figure 7** Examples of cospans that are and are not partial monogamous.

loop $\text{Tr}^1 \left(\begin{array}{|c|} \hline \oplus \\ \hline \end{array} \right) = \bigcirc$, 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 [15, Sec. 6.1]. These closed loops must be represented in the hypergraph framework: there is a natural representation as a lone vertex disconnected from either interface. In fact, this is exactly how the canonical trace applied to an identity is interpreted in $\text{Csp}_D(\mathbf{Hyp}_\Sigma)$.

► **Definition 21.** For a hypergraph $F \in \mathbf{Hyp}$, the degree of a vertex $v \in F_\star$ is a tuple (i, o) where i is the number of pairs (e, i) where e is a hyperedge with v as its i th target, and o is similarly the number of pairs (e, j) where e is a hyperedge with v as its j th target.

► **Definition 22.** For a cospan $m \xrightarrow{f} F \xleftarrow{g} n \in \text{Csp}_D(\mathbf{Hyp}_\Sigma)$, we say it is partial monogamous if f and g are mono and, for all nodes $v \in F_\star$, the degree of v is

$$\begin{array}{ll} (0, 0) & \text{if } v \in f_\star \wedge v \in g_\star \\ (1, 0) & \text{if } v \in g_\star \\ (0, 1) & \text{if } v \in f_\star \\ (0, 0) \text{ or } (1, 1) & \text{otherwise} \end{array}$$

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

► **Example 23.** Examples of cospans that are and are not partial monogamous are shown in Figure 7.

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

► **Theorem 24.** Let $m \rightarrow F \leftarrow n$, $n \rightarrow G \leftarrow p$, $p \rightarrow H \leftarrow q$ and $x + m \rightarrow K \leftarrow x + n$ be partial monogamous cospans in $\text{Csp}_D(\mathbf{Hyp}_\Sigma)$. Then,

- identities and symmetries are partial monogamous;
- $m \rightarrow F \leftarrow n \ ; \ n \rightarrow G \leftarrow p$ is partial monogamous;
- $m \rightarrow F \leftarrow n \otimes p \rightarrow H \leftarrow q$ is partial monogamous; and
- $\text{Tr}^x(x + m \rightarrow K \leftarrow x + n)$ is partial monogamous.

Proof. Since any monogamous hypergraph is also partial monogamous, the first three points hold due to [4, Prop.16], dropping the acyclicity condition. The final condition is routine by case analysis on the interfaces a vertex occurs in. ◀

► **Definition 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.

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 $\text{PMCsp}_D(\mathbf{Hyp}_\Sigma)$.

3.2 From terms to graphs

► **Definition 26.** For a SMT (Σ, \mathcal{E}) , let \mathbf{T}_Σ be the strict STMC freely generated over the generators in Σ . Let $\mathbf{T}_{\Sigma, \mathcal{E}}$ be \mathbf{T}_Σ quotiented by equations in \mathcal{E} .

A (traced) PROP morphism is a strict (traced) symmetric monoidal functor between PROPs. For $\text{PMCSp}_{FI}(\mathbf{Hyp}_\Sigma)$ to be suitable for reasoning with a traced category \mathbf{T}_Σ for some given signature, there must be a fully complete PROP morphism $\mathbf{T}_\Sigma \rightarrow \text{PMCSp}_{FI}(\mathbf{Hyp}_\Sigma)$: a full and faithful functor from terms to cospans of hypergraphs.

We exploit the interplay between compact closed and traced categories in order to reuse the existing PROP morphisms from [4] for the traced case. Since \mathbf{S}_Σ is freely generated, these PROP morphisms can be defined solely on generators.

► **Definition 27 ([4]).** Let $\llbracket - \rrbracket_\Sigma: \mathbf{S}_\Sigma \rightarrow \text{Csp}_D(\mathbf{Hyp}_\Sigma)$ be a PROP morphism defined as

$$\begin{aligned} \llbracket m \boxed{\phi} n \rrbracket_\Sigma &:= m \rightarrow \begin{array}{c} \text{---} \bullet \text{---} \\ \text{---} \bullet \text{---} \\ \text{---} \bullet \text{---} \\ \text{---} \bullet \text{---} \\ \text{---} \bullet \text{---} \end{array} \leftarrow n \\ \llbracket n \boxed{\square} n \rrbracket_\Sigma &:= n \xrightarrow{\text{id}} n \xleftarrow{\text{id}} n \quad \llbracket m \boxed{\boxtimes} n \rrbracket_\Sigma := m + n \xrightarrow{[\text{id}, \text{id}]} m + n \xleftarrow{[\text{id}, \text{id}]} n + m \end{aligned}$$

Let $[-]_\Sigma: \mathbf{Frob} \rightarrow \text{Csp}_D(\mathbf{Hyp}_\Sigma)$ be a PROP morphism defined as in Proposition 18. Then, let $\langle\langle - \rangle\rangle_\Sigma: \mathbf{S}_\Sigma + \mathbf{Frob} \rightarrow \text{Csp}_D(\mathbf{Hyp}_\Sigma)$ be the copairing of $\llbracket - \rrbracket_\Sigma$ and $[-]_\Sigma$.

► **Lemma 28.** Let $m \boxed{f} n$ be a term in \mathbf{T}_Σ . Then there exists at least one ${}^x \boxed{g} n \in \mathbf{S}_\Sigma$ such that $\text{Tr}^x (\boxed{g}) = \boxed{f}$.

► **Proposition 29.** There exists a faithful PROP morphism $[-]_\Sigma^{\mathbf{T}}: \mathbf{T}_\Sigma \rightarrow \mathbf{S}_\Sigma + \mathbf{Frob}$.

Proof. Lemma 28 is used to isolate a term in \mathbf{S}_Σ . The corresponding term in $\mathbf{S}_\Sigma + \mathbf{Frob}$ is then the canonical trace of this term. There may be many such terms in \mathbf{S}_Σ , but the canonical trace being a trace means that any possible outcomes post-trace are all equal. The equations of \mathbf{Frob} do not merge any morphisms since the only use of the generators of \mathbf{Frob} is in the canonical trace, to which the Frobenius equations do not apply. ◀

A summary of these PROP morphisms is shown in Figure 8.

Before progressing to the main theorem, we must show a result about terms in \mathbf{S} : terms constructed from just symmetries and identities. There is a correspondence between such terms and bijective functions.

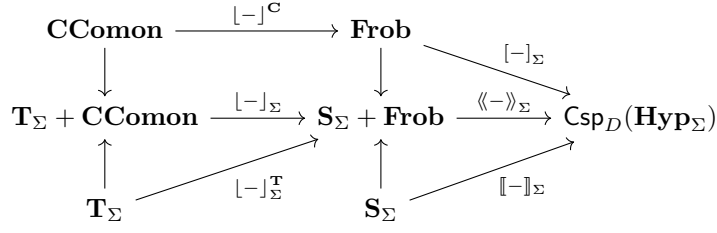
► **Definition 30.** Let \mathbb{P} be the sub-PROP of \mathbb{F} containing only the bijective functions.

► **Lemma 31.** $\mathbf{S} \cong \mathbb{P}$.

► **Lemma 32.** Given a monogamous cospan $m \xrightarrow{f} m \xleftarrow{g} m$, there exists a unique term $m \boxed{h} m \in \mathbf{S}$ up to the axioms of SMCs such that $\llbracket \boxed{h} \rrbracket_\Sigma = m \xrightarrow{f} m \xleftarrow{g} m$.

Proof. Since the cospan is monogamous, f and g are mono. As the cospan is also discrete, there exists a (unique) bijective function $h': [m] \rightarrow [m]$ such that $h'(i) = j$ if $f(i) = g(j)$. By Lemma 31, there is a corresponding term $m \boxed{h} m \in \mathbf{S}$ that is unique up to SMC axioms: a simple induction shows that $\llbracket \boxed{h} \rrbracket_\Sigma = m \xrightarrow{f} m \xleftarrow{g} m$. ◀

Cospans of the form above are used in order to reconstruct a term in \mathbf{T}_Σ given a cospan of partial monogamous hypergraphs, showing that partial monogamy characterises the image of $\langle\langle - \rangle\rangle_\Sigma \circ [-]_\Sigma^{\mathbf{T}}$.



■ **Figure 8** The various PROP morphisms at play.

► **Theorem 33.** *A cospan $m \rightarrow F \leftarrow n$ is in the image of $\llbracket - \rrbracket_\Sigma \circ [-]_\Sigma^\mathbf{T}$ if and only if it is partial monogamous.*

Proof (Sketch). The (\Rightarrow) direction is by induction on the structure of the term. For the (\Leftarrow) direction, a cospan isomorphic to the original cospan can be constructed from which a term in \mathbf{T}_Σ can be read off. Informally, this cospan is

$$\mathrm{Tr}^{x+n}(x + n + m \rightarrow V \leftarrow m + x + n \ ; \ m + x + n \rightarrow L + E + n \leftarrow x + n + n) \quad (1)$$

where V is all the vertices in F , L is the vertices with degree $(0, 0)$ not in the image of the interfaces, and E is the all the hyperedges in F , “stacked” in some arbitrary order. The first component corresponds to a term in \mathbf{S} by Lemma 32, and the stack of edges to a tensor of generators in \mathbf{T}_Σ . ◀

This shows that $\llbracket - \rrbracket_\Sigma \circ [-]_\Sigma^\mathbf{T}$ is a *full* mapping from \mathbf{T}_Σ to $\mathrm{PMCSp}_D(\mathbf{Hyp}_\Sigma)$. It remains to show that it is faithful: every term in \mathbf{T}_Σ has a *unique* cospan of hypergraphs up to isomorphism. By definition, $[-]_\Sigma^\mathbf{T}$ is faithful, so we only need to consider $\llbracket - \rrbracket_\Sigma$.

► **Proposition 34** ([4]). $\llbracket - \rrbracket_\Sigma$ and $[-]_\Sigma$ are faithful.

► **Corollary 35.** $\mathbf{T}_\Sigma \cong \mathrm{PMCSp}_{FI}(\mathbf{Hyp}_\Sigma)$.

4 Hypergraphs for traced commutative comonoid categories

We are interested in another element of structure in addition to the trace: the ability to *copy* and *discard* wires. This is known as a (*commutative*) *comonoid structure*: categories equipped with such a structure are also known as *gs-monoidal* (*garbage-sharing*) categories [13].

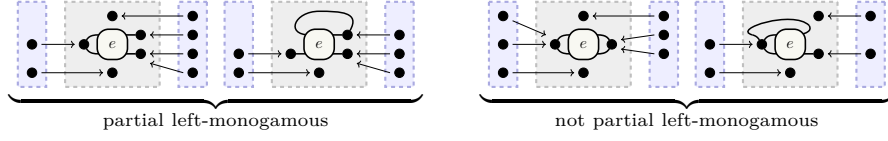
► **Definition 36.** Let $(\Sigma_{\mathbf{CComon}}, \mathcal{E}_{\mathbf{CComon}})$ be the symmetric monoidal theory of commutative comonoids, with $\Sigma_{\mathbf{CComon}} := \{ \begin{array}{c} \square \\ \leftarrow \end{array}, \begin{array}{c} \square \\ \rightarrow \end{array} \}$ and $\mathcal{E}_{\mathbf{CComon}}$ defined as in Figure 2. We write $\mathbf{CComon} := \mathbf{S}_{\Sigma_{\mathbf{CComon}}, \mathcal{E}_{\mathbf{CComon}}}$.

From now on, we write “comonoid” to mean “commutative comonoid”. There has already been work using hypergraphs for PROPs with a (co)monoid structure [13, 27] but these consider *acyclic* hypergraphs: we must ensure that removing the acyclicity condition does not lead to any degeneracies.

► **Definition 37** (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 $v \in H_\star$, the degree of v is $(0, m)$ if $v \in f_\star$ and $(0, m)$ or $(1, m)$ otherwise, for some $m \in \mathbb{N}$.*

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

14:10 Rewriting Modulo Traced Comonoid Structure



■ **Figure 9** Examples of cospans that are and are not partial left-monogamous.

► **Example 38.** Examples of cospans that are and are not partial left-monogamous are shown in Figure 9.

► **Remark 39.** As with the vertices not in the interfaces with degree $(0, 0)$ in the vanilla traced case, the vertices not in the interface with degree $(0, m)$ allow for terms such as $\text{Tr} \left(\begin{array}{c} \square \\ \square \end{array} \right)$.

► **Lemma 40.** Let $m \rightarrow F \leftarrow n$, $n \rightarrow G \leftarrow p$, $p \rightarrow H \leftarrow q$ and $x + m \rightarrow K \leftarrow x + n$ be partial left-monogamous cospans. Then,

- identities and symmetries are partial left-monogamous;
- $m \rightarrow F \leftarrow n \circledast n \rightarrow G \leftarrow p$ is partial left-monogamous;
- $m \rightarrow F \leftarrow n \otimes p \rightarrow H \leftarrow q$ is partial left-monogamous; and
- $\text{Tr}^x (x + m \rightarrow K \leftarrow x + n)$ is partial left-monogamous.

► **Definition 41.** 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.

This category can be equipped with a comonoid structure.

► **Definition 42.** Let $[-]^\mathbf{C}: \mathbf{CComon} \rightarrow \mathbf{Frob}$ 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 $[-]_\Sigma^\mathbf{T}$ and $[-]^\mathbf{C}$.

As before, these PROP morphisms are summarised in Figure 8. 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 $\text{PLMCsp}_D(\mathbf{Hyp}_\Sigma)$.

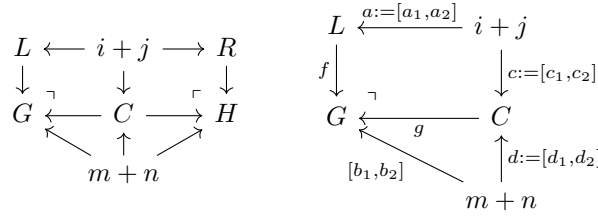
► **Lemma 43.** The image of $[-]_\Sigma \circ [-]^\mathbf{C}$ is in $\text{PLMCsp}_D(\mathbf{Hyp}_\Sigma)$.

► **Corollary 44.** The image of $\langle\langle - \rangle\rangle_\Sigma \circ [-]_\Sigma$ is in $\text{PLMCsp}_D(\mathbf{Hyp}_\Sigma)$.

► **Lemma 45.** Given a partial left-monogamous cospan $m \xrightarrow{f} m \xleftarrow{g} n$, there exists a unique term $m \xrightarrow{h} n \in \mathbf{CComon}$ up to the axioms of SMCs and comonoids such that $\left[\begin{array}{c} \square \\ \square \end{array} \right]_\Sigma^\mathbf{C} = m \xrightarrow{f} m \xleftarrow{g} n$.

► **Theorem 46.** $\mathbf{T}_\Sigma + \mathbf{CComon} \cong \text{PLMCsp}_{FI}(\mathbf{Hyp}_\Sigma)$.

Proof. Since $\langle\langle - \rangle\rangle_\Sigma$ and $[-]_\Sigma^\mathbf{C}$ are faithful, it suffices to show that a cospan $m \rightarrow F \leftarrow n$ in $\text{PLMCsp}_D(\mathbf{Hyp}_\Sigma)$ can be decomposed in such a way that each component is in the image of either $\langle\langle - \rangle\rangle_\Sigma \circ [-]_\Sigma^\mathbf{T}$ or $[-]_\Sigma \circ [-]^\mathbf{C}$. This is achieved by taking the construction of Theorem 33 and allowing the first component to be partial left-monogamous; by Lemma 45 a term in \mathbf{CComon} can be retrieved from this. ◀



■ **Figure 10** The DPO diagram and a pushout complement.

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 hypergraphs. However, to reason about a *monoidal theory* with extra equations we must actually rewrite the components of the graph. In the syntactic realm this is performed with *term rewriting*.

► **Definition 47** (Term rewriting). A rewriting system \mathcal{R} for a traced PROP \mathbf{T}_Σ consists of a set of rewrite rules $\langle i \text{---} \boxed{l} \text{---} j, i \text{---} \boxed{r} \text{---} j \rangle$. Given terms $m \text{---} \boxed{g} \text{---} n$ and $m \text{---} \boxed{h} \text{---} n$ in \mathbf{T}_Σ we write $\boxed{g} \Rightarrow_{\mathcal{R}} \boxed{h}$ if there exists rewrite rule $(i \text{---} \boxed{l} \text{---} j, i \text{---} \boxed{r} \text{---} j)$ in \mathcal{R} and $\boxed{c} \text{---}^j \text{---}^i$ in \mathbf{T}_Σ such that $\boxed{g} = \boxed{l} \text{---} \boxed{c}$ and $\boxed{h} = \boxed{r} \text{---} \boxed{c}$ by axioms of STMCs.

The equivalent for graphs is *graph rewriting*. A common framework is that of *double pushout rewriting* (DPO rewriting); we use an extension, known as *double pushout rewriting with interfaces* (DPOI rewriting).

► **Definition 48** (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 $L \xleftarrow{[a_1, a_2]} i + j \xrightarrow{[b_1, b_2]} R$.

► **Definition 49** (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 $i + j \rightarrow C \leftarrow n + m \in \mathbf{Hyp}_\Sigma$ such that diagram in the left of Figure 10 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 50** ([5]). Let $\lceil - \rceil : \mathbf{S}_\Sigma + \mathbf{Frob} \rightarrow \mathbf{S}_\Sigma + \mathbf{Frob}$ be defined as having action $m \text{---} \boxed{f} \text{---} n \mapsto \boxed{f} \text{---}^m \text{---}^n$.

Note that the result of applying $\lceil - \rceil$ is not in the image of $\langle \langle - \rangle \rangle_\Sigma \circ \lceil - \rceil_\Sigma^{\mathbf{T}}$ any more. This is not an issue, so long as we “unfold” the interfaces once rewriting is completed.

► **Proposition 51** ([4, Prop. 4.8]). If $\langle \langle m \text{---} \boxed{f} \text{---} n \rangle \rangle_\Sigma = m \xrightarrow{i} F \xleftarrow{o} n$ 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 .

14:12 Rewriting Modulo Traced Comonoid Structure

► **Definition 52** (Pushout complement). *Let $i + j \rightarrow L \rightarrow G \rightarrow m + n$ be morphisms in \mathbf{Hyp}_Σ . Then the pushout complement of these morphisms is an object C with morphisms $i + j \rightarrow C \rightarrow G$ such that $L \rightarrow G \leftarrow C$ is a pushout and the diagram on the right of Figure 10 commutes.*

Given a rule $L \leftarrow i + j \rightarrow R$ and morphism $L \rightarrow G$, a pushout complement $i + j \rightarrow C \rightarrow G$ represents the context of a valid rewrite step. Once a pushout complement is computed, the pushout of $C \leftarrow i + j \rightarrow R$ can be performed to obtain the completed rewrite H . However, a pushout complement may not exist for a given rule and matching.

► **Definition 53** ([4], Def. 3.16). *Let $i + j \xrightarrow{a} L \xrightarrow{f} G$ be morphisms in \mathbf{Hyp}_Σ . The morphisms satisfy the no-dangling condition if, for every hyperedge not in the image of f , each of its source and target vertices is either not in the image of f or are in the image of $f \circ a$. The morphisms satisfy the no-identification condition if any two distinct elements merged by f are also in the image of $f \circ a$.*

► **Proposition 54** ([4], Prop. 3.17). *The 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 55**. *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* [23], it is possible to be more precise about the number of pushout complements for a given matching and rewrite rule.

► **Proposition 56** ([23]). *In an adhesive category, pushout complements of $i + j \xrightarrow{a} L \rightarrow G$ are unique if they exist and a is mono.*

► **Proposition 57** ([24]). *\mathbf{Hyp}_Σ is adhesive.*

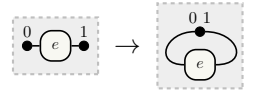
A given pushout complement uniquely determines the rewrite performed, so it might seem advantageous to always have exactly one. However, when writing modulo traced comonoid structure there are settings where having multiple pushout complements is beneficial.

5.1 Rewriting with traced structure

While in the Frobenius case considered in [4], all valid pushout complements correspond to a valid rewrite, this is not the case for the traced monoidal case. In [5], pushout complements that correspond to a valid rewrite in the non-traced symmetric monoidal case are identified as *boundary complements*. We will use a weakening of this definition.

► **Definition 58** (Traced boundary complement). *A pushout complement as in Definition 52 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 [5], 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:



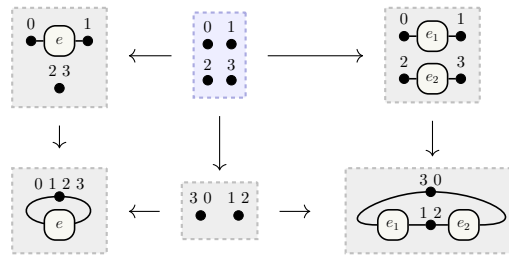
► **Definition 59** (Traced DPO). *For morphisms $G \leftarrow m + n$ and $H \leftarrow m + n$ in \mathbf{Hyp}_Σ , there is a traced rewrite $G \rightsquigarrow_{\mathcal{R}} H$ if there exists a rule $L \leftarrow i + j \rightarrow G \in \mathcal{R}$ and cospan $i + j \rightarrow C \leftarrow n + m \in \mathbf{Hyp}_\Sigma$ such that diagram in Definition 49 commutes and $i + j \rightarrow C$ 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.

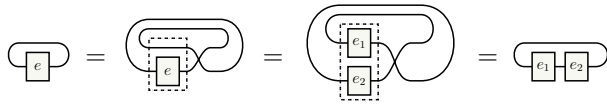
► **Lemma 60.** *In a traced boundary complement, let $v \in i$ and w_0, w_1, \dots, w_k such that $f(a_1(v)) = f(a_2(w_0))$, $f(a_1(v)) = f(a_2(w_1))$ and so on. Then either (1) there exists exactly one w_i not in the image of d_1 such that $c_1(v) = c_2(w_i)$; (2) $c_1(v)$ is in the image of d_1 ; 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 $(0, 1)$.*

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



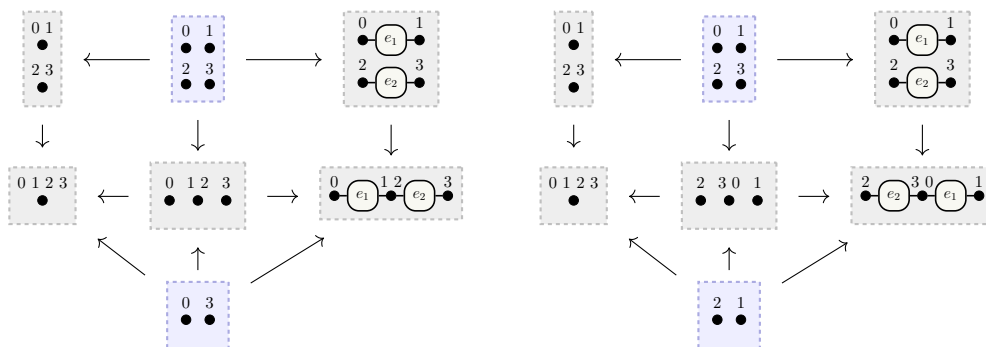
This corresponds to a valid term rewrite:



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.

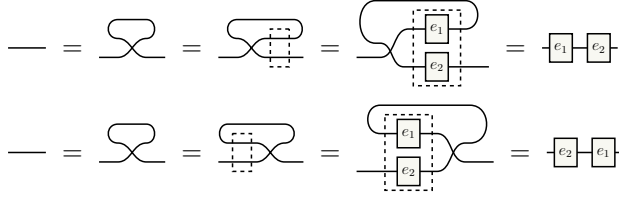
Unlike regular boundary complements, traced boundary complements need not be unique. However, this is not a problem since all pushout complements can be enumerated given a rule and matching [18].

► **Example 62.** Consider the rule $\langle \boxed{\quad}, \begin{array}{c} \boxed{e_1} \\ \boxed{e_2} \end{array} \rangle$. Below are two valid traced boundary complements involving a matching of this rule.



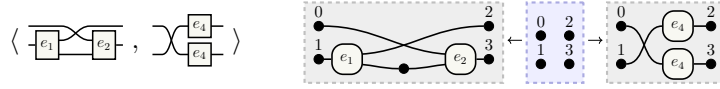
14:14 Rewriting Modulo Traced Comonoid Structure

Once again, these derivations arise through yanking:

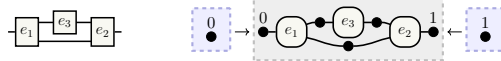


Rewriting modulo yanking also eliminates another foible of rewriting modulo (non-traced) symmetric monoidal structure. In the SMC case, the image of the matching must be *convex*: any path between vertices must also be captured. This is not necessary in the traced case.

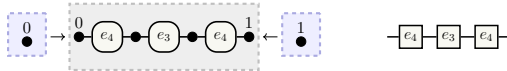
► **Example 63.** Consider the following rewrite rule and its interpretation.



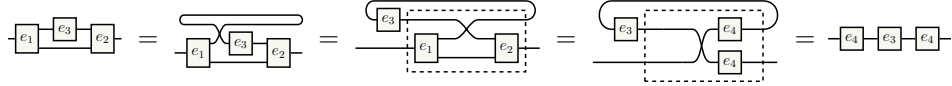
Now consider the following term and interpretation:



Although it is not obvious in the original string diagram, 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! However, it is possible with yanking.



We are almost ready to show the soundness and completeness of this DPO rewriting system. The final prerequisite is a decomposition lemma, akin to a similar result in [4].

► **Lemma 64 (Traced decomposition).** *Given partial monogamous cospans $m \xrightarrow{d_1} G \xleftarrow{d_2} n$ and $i \xrightarrow{a_1} L \xleftarrow{a_2} j$, 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 $j + m \xrightarrow{[c_2, d_1]} C \xleftarrow{[c_1, d_2]} i + n$ such that $m \rightarrow G \leftarrow n$ can be factored as*

$$\mathrm{Tr}^i \left(\begin{array}{c} i \xrightarrow{a_1} L \xleftarrow{a_2} j \\ \otimes \\ m \rightarrow m \leftarrow m \end{array} ; j + m \xrightarrow{[c_2, d_1]} C \xleftarrow{[c_1, d_2]} i + n \right) \quad (2)$$

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

We write $\ulcorner [\mathcal{R}]_{\Sigma}^{\mathbf{T}\urcorner}$ for the pointwise map $(\ulcorner l \urcorner, \ulcorner r \urcorner) \mapsto (\ulcorner l \urcorner]_{\Sigma}^{\mathbf{T}\urcorner}, \ulcorner r \urcorner]_{\Sigma}^{\mathbf{T}\urcorner}$.

► **Theorem 65.** *Let \mathcal{R} be a rewriting system on \mathbf{T}_{Σ} . Then, $m \ulcorner g \urcorner \Rightarrow_{\mathcal{R}} m \ulcorner h \urcorner$ if and only if $\llcorner \ulcorner \ulcorner g \urcorner \urcorner \rrcorner \gg_{\Sigma} \rightsquigarrow \llcorner \ulcorner \ulcorner h \urcorner \urcorner \rrcorner \gg_{\Sigma}$.*

Proof. (\Rightarrow) follows by defining cospans corresponding each part of Definition 47 and composing them together: since composition of cospans is by pushout, the DPO diagram can be recovered and the pushouts checked to be traced boundary complements. (\Leftarrow) follows by using Lemma 64 and the fullness of $\llcorner - \gg_{\Sigma}$ to obtain the pieces of Definition 47. ◀

5.2 Rewriting with traced comonoid structure

To extend rewriting with traced structure to the comonoid case, the traced boundary complement conditions need to be weakened to the case of *left-monogamous* cospans.

► **Definition 66** (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 58 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.*

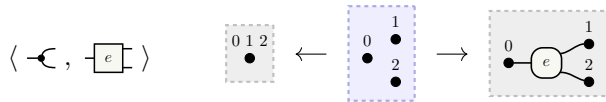
► **Definition 67** (Traced comonoid DPO). *For morphisms $G \leftarrow m + n$ and $H \leftarrow m + n$ in \mathbf{Hyp}_Σ , there is a traced comonoid rewrite $G \rightsquigarrow_{\mathcal{R}} H$ if there exists a rule $L \leftarrow i + j \rightarrow G \in \mathcal{R}$ and cospan $i + j \rightarrow C \leftarrow n + m \in \mathbf{Hyp}_\Sigma$ such that diagram in Definition 49 commutes and $i + j \rightarrow C \rightarrow G$ is a traced left-boundary complement.*

► **Lemma 68** (Traced comonoid decomposition). *Lemma 64 holds when all cospans are partial left-monogamous and $j + m \xrightarrow{[c_2, d_1]} C \xleftarrow{[c_1, d_2]} i + n$ is a traced left-boundary complement.*

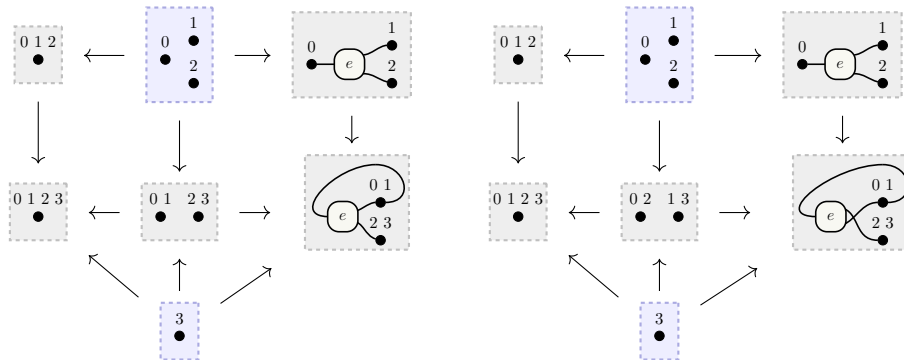
► **Theorem 69.** *Let \mathcal{R} be a rewriting system on $\mathbf{T}_\Sigma + \mathbf{CComon}$. Then, $\boxed{g} \Rightarrow_{\mathcal{R}} \boxed{h}$ in $\mathbf{T}_\Sigma + \mathbf{CComon}$ if and only if $\langle\langle \ulcorner \boxed{g} \urcorner \rfloor_\Sigma \urcorner \rangle\rangle_\Sigma \rightsquigarrow \langle\langle \ulcorner \boxed{h} \urcorner \rfloor_\Sigma \urcorner \rangle\rangle_\Sigma$.*

Proof. As Theorem 65, but with traced left-boundary complements. ◀

► **Example 70.** As with the traced case, there may be multiple valid rewrites given a particular interface. The comonoid structure adds more possibilities, as there are the equations of commutative comonoids to consider. Consider the following rule and its interpretation.



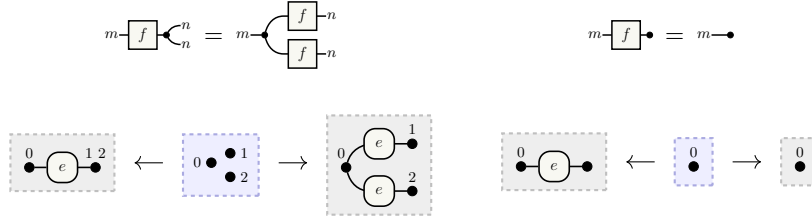
Two valid rewrites are as follows:



The first rewrite is the “obvious” one, but the second also holds by cocommutativity:



14:16 Rewriting Modulo Traced Comonoid Structure



■ **Figure 11** Equations of the monoidal theory $\mathbf{Cart}_{\mathcal{C}}$, where $m\boxed{f}_n$ is an arbitrary morphism in \mathcal{C} , and the interpretations of these equations as rewrite rules for an arbitrary generator e .

6 Case studies

6.1 Cartesian structure

One important class of categories with a traced comonoid structure are *traced Cartesian*, or *dataflow*, categories [8, 15]. These categories are interesting because any traced Cartesian category admits a fixpoint operator [15, Thm. 3.1].

► **Definition 71** (Cartesian category [12]). *A monoidal category is Cartesian if its tensor is given by the Cartesian product.*

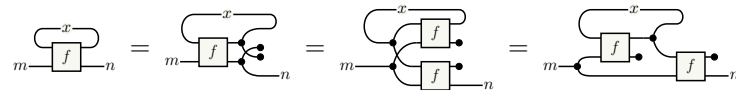
As a result of this, the unit is a terminal object in any Cartesian category, and any object has a comonoid structure. Cartesian categories are settings in which objects can be *copied* and *discarded*. These two operations are more clearly illustrated when viewed through the lens of a monoidal theory.

► **Definition 72.** *For a given base PROP $\mathbf{T}_{\Sigma_{\mathcal{C}}}$ with a comonoid structure, the monoidal theory $(\Sigma_{\mathbf{Cart}_{\mathcal{C}}}, \mathcal{E}_{\mathbf{Cart}_{\mathcal{C}}})$ is defined with $\Sigma_{\mathbf{Cart}_{\mathcal{C}}} := \Sigma_{\mathcal{C}}$ and $\mathcal{E}_{\mathbf{Cart}_{\mathcal{C}}}$ as the equations in Figure 11.*

Note that as the equations in $\mathcal{E}_{\mathbf{Cart}_{\mathcal{C}}}$ are parameterised over any morphism $m\boxed{f}_n$, a separate DPO rewrite rule is required for every combination of generators as in Figure 11. However, as is the case in the next section, it is often possible to characterise the copying behaviour through a finite number of equations.

► **Remark 73.** 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.



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 the copy equation needs to be applied.

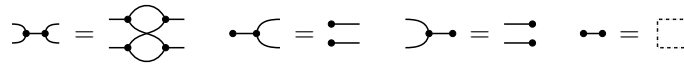
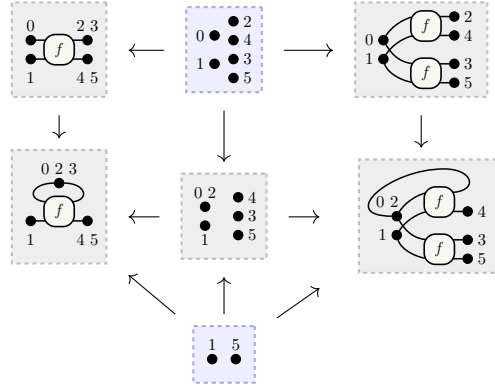


Figure 12 Equations $\mathcal{E}_{\text{Bialg}}$ of a *bialgebra*, in addition to those in Figures 1 and 2.



The dual notion of traced *cocartesian* categories [2] 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 18) subject to the Frobenius equations in Figure 3. 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 *digital circuits* built from primitive logic gates: in [14], digital circuits are modelled as morphisms in a STMC. Here, the trace models a feedback loop, and the comonoid structure represents forking wires. The semantics of digital circuits can be expressed as a monoidal theory [14, Sec. 6].

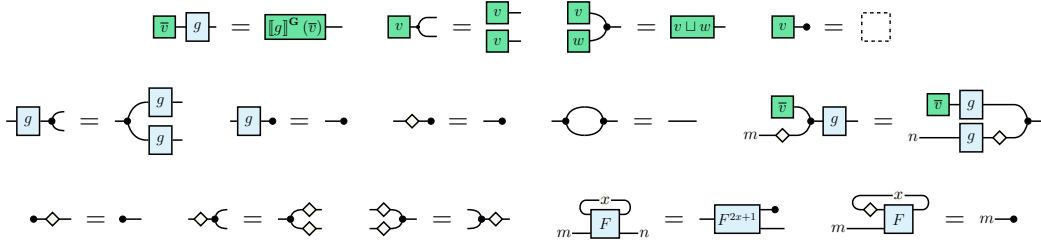
► **Definition 74** (Gate-level circuits). *Let the monoidal theory of gate-level sequential circuits be defined as $(\Sigma_{\text{SCirc}}, \mathcal{E}_{\text{SCirc}})$, where*

$$\Sigma_{\text{SCirc}} := \{ \boxed{\text{AND}}, \boxed{\text{OR}}, \boxed{\text{NOT}}, \boxed{\bullet}, \boxed{\leftarrow}, \boxed{\rightarrow}, \boxed{\text{t}}, \boxed{\text{f}}, \boxed{\text{b}}, \boxed{\text{delay}} \}$$

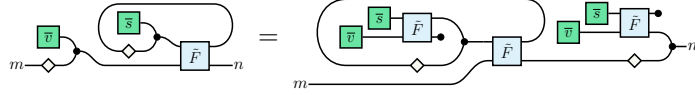
and the equations of $\mathcal{E}_{\text{SCirc}}$ are listed in Figures 1, 2, 12, and 13, where $\llbracket - \rrbracket^{\mathbf{G}}$ maps gates to the corresponding truth table, \sqcup is the join in a lattice structure on $\{\bullet, \text{t}, \text{f}, \text{b}\}$, and $m\text{-}\boxed{F^n}_n$ is defined inductively as $\boxed{F^0} := \boxed{F}$ and $\boxed{F^{k+1}} := \text{trace}(\boxed{F^k}, \boxed{F})$.

The generators in Σ_{SCirc} are, respectively: AND, OR and NOT gates; constructs for introducing, forking, joining and stubbing wires; *values* representing a true signal, a false signal, and both signals at once; and a delay of one unit of time. Note that while the equations in $\mathcal{E}_{\text{SCirc}}$ contain those of a commutative comonoid, they do *not* explicitly contain the general Cartesian equations: instead, these are derived from smaller equations.

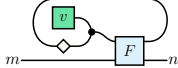
14:18 Rewriting Modulo Traced Comonoid Structure



■ **Figure 13** The equations of $\mathcal{E}_{\text{SCirc}}$, from the monoidal theory of gate-level sequential circuits.



■ **Figure 14** The cycle equation, which is derivable from the equations in $\mathcal{E}_{\text{SCirc}}$.

Using graph rewriting, we can sketch out an *operational semantics* for sequential circuits. For the interests of brevity, we will only consider circuits of the form : circuits with no “non-delay-guarded feedback” in which the registers of the circuit have been isolated from a core containing only “blue” (*combinational*) components, which models a function. Any sequential circuit can be translated into such a form by the equational theory.

We can “apply” such a circuit to an input as shown in the left-hand side of Figure 14; the equations in $\mathcal{E}_{\text{SCirc}}$ can be used to derive the right-hand side. The four equations in the top row of Figure 13 can then be repeatedly applied to reduce the two “new” cores down to values, representing the output and new state of the circuit.

When the circuits are interpreted as hypergraphs and the equations as rewrites, a computer could perform this sequence of rewrites in order to evaluate circuits in a step-by-step manner.

7 Conclusion, related and further work

We have shown how the frameworks for rewriting string diagrams modulo Frobenius [4] and symmetric monoidal [5] structure using hypergraphs can also be adapted for rewriting modulo traced comonoid structure, by using hypergraphs that sit between the two settings.

Graphical languages for traced categories have seen many applications, such as to illustrate cyclic lambda calculi [15], or to reason graphically about programs [29]. The presentation of traced categories as *string diagrams* has existed since the 90s [19, 20]; a soundness and completeness theorem for traced string diagrams, folklore for many years but only proven for certain signatures [30], was finally shown in [22]. Combinatorial languages predate even this, having existed since at least the 80s in the guise of *flowchart schemes* [33, 8, 9]. These diagrams have also been used to show the completeness of finite dimensional vector spaces [17] with respect to traced categories and, when equipped with a dagger, Hilbert spaces [31].

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* [21, 10]. We have instead opted to use the *hypergraph* framework of [4, 5, 6] 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* [1], 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 vertex is created for each wire, and simple diagrams can quickly get very large. The results of [5] also extend to the multi-sorted case, in which vertices are labelled in addition to wires. We could use this in combination with the *strictifiers* of [32]: 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.

References

- 1 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.
- 2 E. S. Bainbridge. Feedback and generalized logic. *Information and Control*, 31(1):75–96, May 1976. doi:10.1016/S0019-9958(76)90390-9.
- 3 Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Paweł Sobociński, and Fabio Zanasi. Rewriting modulo symmetric monoidal structure. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16*, pages 710–719, New York, NY, USA, July 2016. Association for Computing Machinery. doi:10.1145/2933575.2935316.
- 4 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.
- 5 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.
- 6 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*, pages 1–41, June 2022. doi:10.1017/S0960129522000123.
- 7 A. Carboni and R. F. C. Walters. Cartesian bicategories I. *Journal of Pure and Applied Algebra*, 49(1):11–32, November 1987. doi:10.1016/0022-4049(87)90121-6.
- 8 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.
- 9 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.
- 10 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.
- 11 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.

- 12 Thomas Fox. Coalgebras and cartesian categories. *Communications in Algebra*, 4(7):665–667, January 1976. doi:10.1080/00927877608822127.
- 13 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.
- 14 Dan R. Ghica, George Kaye, and David Sprunger. A compositional theory of digital circuits. *CoRR*, abs/2201.10456, February 2023. doi:10.48550/arXiv.2201.10456.
- 15 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.
- 16 Masahito Hasegawa. On traced monoidal closed categories. *Mathematical Structures in Computer Science*, 19(2):217–244, April 2009. doi:10.1017/S0960129508007184.
- 17 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.
- 18 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.
- 19 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.
- 20 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.
- 21 Aleks Kissinger. *Pictures of Processes: Automated Graph Rewriting for Monoidal Categories and Applications to Quantum Computing*. PhD thesis, University of Oxford, March 2012. doi:10.48550/arXiv.1203.0202.
- 22 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.
- 23 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, Berlin, Heidelberg, 2004. Springer. doi:10.1007/978-3-540-24727-2_20.
- 24 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.
- 25 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.
- 26 Saunders MacLane. Categorical algebra. *Bulletin of the American Mathematical Society*, 71(1):40–106, 1965. doi:10.1090/S0002-9904-1965-11234-4.
- 27 Aleksandar Milosavljevic, Robin Piedeleu, and Fabio Zanasi. String Diagram Rewriting Modulo Commutative (Co)monoid Structure. *CoRR*, abs/2204.04274, March 2023. doi:10.48550/arXiv.2204.04274.
- 28 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.
- 29 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.

- 30 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.
- 31 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.
- 32 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.
- 33 Gheorghe Ștefănescu. Feedback Theories (A Calculus for Isomorphism Classes of Flowchart Schemes). *Romanian Journal of Pure and Applied Mathematics*, 35(1):73–79, 1990.