CAMBRIDGE
UNIVERSITY PRESS

PAPER

String diagram rewrite theory II: Rewriting with symmetric monoidal structure

Filippo Bonchi[1], Fabio Gadducci[1]* , Aleks Kissinger[2], Pawel Sobocinski[3] and Fabio Zanasi[4]

[1]University of Pisa, Pisa, Italy, [2]University of Oxford, Oxford OX1 2JD, UK, [3]Tallinn University of Technology, Tallinn, Estonia and [4]University College London, London WC1E 6BT, UK
*Corresponding author. Email: fabio.gadducci@unipi.it

Abstract

Symmetric monoidal theories (SMTs) generalise algebraic theories in a way that make them suitable to express resource-sensitive systems, in which variables cannot be copied or discarded at will. In SMTs, traditional tree-like terms are replaced by *string diagrams*, topological entities that can be intuitively thought of as diagrams of wires and boxes. Recently, string diagrams have become increasingly popular as a graphical syntax to reason about computational models across diverse fields, including programming language semantics, circuit theory, quantum mechanics, linguistics, and control theory. In applications, it is often convenient to implement the equations appearing in SMTs as *rewriting rules*. This poses the challenge of extending the traditional theory of term rewriting, which has been developed for algebraic theories, to string diagrams. In this paper, we develop a mathematical theory of string diagram rewriting for SMTs. Our approach exploits the correspondence between string diagram rewriting and double pushout (DPO) rewriting of certain graphs, introduced in the first paper of this series. Such a correspondence is only sound when the SMT includes a *Frobenius algebra* structure. In the present work, we show how an analogous correspondence may be established for arbitrary SMTs, once an appropriate notion of DPO rewriting (which we call *convex*) is identified. As proof of concept, we use our approach to show termination of two SMTs of interest: Frobenius semi-algebras and bialgebras.

Keywords: String diagram; symmetric monoidal category; double-pushout rewriting

1. Introduction

The study of algebraic theories and their role in modelling computing systems (Behrisch et al. 2012; Hyland and Power 2007) is a recurring theme of John Power's research, and the subject of some of his most influential contributions. In a series of articles (Garner and Power 2018; Lack and Power 2009; Power 1999, 2005), he and his coauthors developed an enriched category theoretic generalisation of Lawvere theories and explored their applications, particularly in the study of computational effects of programming languages. Whereas monads provide a powerful theory for principled and compositional definitions of *denotational* semantics, as pioneered by Moggi (1991), algebraic theories are particularly useful (Power 2004, 2006*a*,*b*) in the development of formal and principled approaches to *operational* semantics, as shown in a series of articles as part of a long running and productive collaboration with Gordon Plotkin (Hyland et al. 2002; Plotkin and Power 2001*a*,*b*, 2002, 2003, 2004).

   There have been several efforts to generalise the notion of algebraic theory in general, and that of Lawvere theory in particular. Especially after the work of Lack (2004), the theory of PROPs (Mac

CrossMark

Lane 1965) – a particularly simple family of symmetric strict monoidal categories – has been advanced as a categorical tool for the study of algebraic theories, and PROPs have been applied in several parts of computer science (Baez and Erbele 2015; Bonchi et al. 2017*b*,*d*, 2019; Coecke and Duncan 2011; Coecke and Kissinger 2017; Ghica et al. 2017; Jacobs et al. 2019; Sadrzadeh et al. 2013; Zanasi 2015). The notion of algebraic theory here is that of symmetric monoidal theory (SMT), with the essential difference being that the underlying assumption of Cartesianity is discarded. Indeed, PROPs generalise Lawvere theories, since the latter are nothing but Cartesian PROPs. The correspondence is well-behaved enough to extend to *presentations* of theories: indeed, it has been since long understood (Fox 1976) that any presentation of a Lawvere theory can be seen as a presentation of a SMT (Bonchi et al. 2018*b*). PROPs are more general and can be used to capture partial and relational theories (Bonchi et al. 2017*c*; Corradini and Gadducci 2002; Liberti et al. 2021; Zanasi 2016). Overall, it appears that symmetric monoidal structure is the axiomatic baseline for many pertinent examples.

One of the driving motivations for the development of rewriting theory has been the desire to implement aspects of algebraic theories. For example, the word problem for a (presentation of an) algebraic theory is decidable if one can orient the equations $l = r$, obtaining rewriting rules $l \to r$, and prove confluence and termination of the resulting rewriting system. In this way, one obtains normal forms. To decide whether two terms are judged equal by such an algebraic theory, it suffices to rewrite both until no more redexes are found, then compare the results: they are equal precisely when they rewrite to the same normal form. However, classical rewriting theory has been developed for ordinary terms, which are intimately connected with classical algebraic theories.

The big question driving the theoretical contributions of this paper is 'how does one implement algebraic theories captured by PROPs?'. If we take rewriting as an answer, then the rewriting has to be done up-to the axioms of symmetric strict monoidal categories.

Traditional terms enjoy a pleasantly simple structure: their syntactic decomposition may be represented as trees. Analogously, the structure of terms of symmetric monoidal categories may be represented as a particular family of string diagrams. However, there is an underlying problem: trees are combinatorial objects that are elements of a classical, inductive data type, with well-understood and efficient algorithms that are exploited for rewriting. On the other hand, string diagrams have traditionally been considered as topological entities (Joyal and Street 1991). Our first task is therefore to understand string diagrams as combinatorial objects. In the prequel (Bonchi et al. 2022) to this paper, we showed a close connection between string diagrams over a signature Σ and the category of discrete cospans of hypergraphs with Σ-typed edges. Nevertheless, the correspondence is not an isomorphism: for isomorphic cospans to be equated as string diagrams, they must be considered up to an underlying special Frobenius structure. While examples of such theories abound, here we consider mere symmetric monoidal structure. In Theorem 25, we characterise those cospans that arise via this correspondence, and in Proposition 27 we extend this characterisation to the multi-coloured case. The cospans of interest are those whose underlying hypergraph is acyclic and satisfies an additional technical condition that we refer to as monogamy. Checking both is algorithmically simple enough.

Having identified a satisfactory combinatorial representation leads us to the actual mechanism of rewriting, which is an adaptation of the DPO approach (Corradini et al. 1997). The first modification of classical DPO is forced on us by the fact that we are rewriting cospans of hypergraphs, hence the rewriting has to be done in a way that respects the interfaces. Similar approaches have been considered previously in the literature on graph rewriting, though, the most notable example being Ehrig and König (2004). Second, and more seriously, the general mechanism of DPO rewriting is not sound for mere symmetric monoidal structure. The reason for this has already been highlighted in the previous paragraph: the correspondence between string diagrams and cospans of hypergraphs works when string diagrams are considered modulo the axioms of symmetric monoidal categories as well as those of the Frobenius structure. Given that we do not want

to assume the presence of Frobenius, we must suitably restrict the DPO mechanism. We introduce two technical modifications: first, legal pushout complements are restricted to a variant we call boundary complements in order to preserve monogamy and acyclicity of the resulting cospan, and second, matches have to be restricted to convex matches, which have a topologically intuitive explanation. We call the resulting variant convex DPO rewriting and show that it is a sound and complete mechanism for rewriting modulo symmetric monoidal structure in Theorem 35.

To illustrate the framework, we study two examples of symmetric monoidal theories (SMTs): Frobenius semi-algebras and bialgebras. For both theories, we demonstrate straightforward proofs of termination making explicit use of the graph-theoretic structure. We furthermore show that the theory of Frobenius semi-algebras is not confluent using a surprising property of convex rewriting: namely that non-overlapping rewriting rule applications can interfere with each other. Our counter-example to confluence is adapted from an example due to Power (1991), which was originally given in the context of pasting diagrams for 3-categories. Incidentally, Power's example also occurred in a festschrift, in honour of Max Kelly's 60th birthday in 1991.

This paper is based on previous conference articles (Bonchi et al. 2016, 2017a, 2018a), and it is the second in the series, following Bonchi et al. (2022), that collects these results in a coherent and comprehensive narrative. The formulation of our characterisation for coloured props, as well as the case studies of Frobenius semi-algebras (Section 5.1) and bialgebras (Section 5.2), is novel with respect to the conference papers.

Structure of the paper. Although the material in this paper is a prosecution of Bonchi et al. (2022), we have tried to make the presentation self-contained. We give the background material in Section 2. In Section 3, we give the characterisation of string diagrams for PROPS as discrete cospans of hypergraphs that are acyclic and monogamous. In Section 4, we develop convex DPO rewriting, the mechanism that correctly implements rewriting modulo symmetric monoidal structure. Finally, in Section 5, we consider two case studies: Frobenius semi-algebras and bialgebras.

2. Background

2.1 Symmetric monoidal theories and PROPs

In this section, we recall some basic notions and fix notation. We confine ourselves to the definitions that are strictly necessary for our developments and refer the reader to the first part of this exposition (Bonchi et al. 2022) for a gentler introduction to the same notions.

A SMT is a pair $(\Sigma, \mathscr{E})$. Here, Σ is a *monoidal signature*, consisting of operations $o\colon n \to m$ of a fixed *arity* n and *coarity* m, for $n, m \in \mathbb{N}$. The second element $\mathscr{E}$ is a set of *equations*, namely pairs of Σ-*terms* $l, r\colon v \to w$ with the same arity and coarity. Recall that Σ-terms are constructed by combining the operations in Σ, *identities* $\mathrm{id}_n\colon n \to n$ and *symmetries* $\sigma_{m,n}\colon m + n \to n + m$ for each $m, n \in \mathbb{N}$, by sequential (;) and parallel ($\oplus$) composition.

SMTs have a categorical rendition as PROPs (Mac Lane 1965) (monoidal **pro**duct and **p**ermutation categories).

Definition 1. (PROP). *A PROP is a symmetric strict monoidal category with objects the natural numbers, where the product on objects, denoted $\oplus$, is addition. Morphisms are identity-on-objects symmetric strict monoidal functors. PROPs and their morphisms form the category* PROP.

An SMT $(\Sigma, \mathscr{E})$ gives raise to a PROP $\mathbf{S}_{\Sigma,\mathscr{E}}$ by letting the arrows $u \to v$ of $\mathbf{S}_{\Sigma,\mathscr{E}}$ be the Σ-terms $u \to v$ modulo the laws of symmetric monoidal categories (Figure 1) and the smallest congruence containing the equations $t = t'$ for any $(t, t') \in \mathscr{E}$. We are going to represent these arrows by using

$$(s\,;t)\,;u = s\,;(t\,;u) \qquad \mathrm{id}_n\,;s = s = s\,;\mathrm{id}_m$$
$$(s \oplus t) \oplus u = s \oplus (t \oplus u) \qquad \mathrm{id}_0 \oplus s = s = s \oplus \mathrm{id}_0$$
$$(s\,;u) \oplus (t\,;v) = (s \oplus t)\,;(u \oplus v) \qquad \mathrm{id}_m \oplus \mathrm{id}_n = \mathrm{id}_{m+n}$$
$$(\sigma_{m,n} \oplus \mathrm{id}_p)\,;(\mathrm{id}_n \oplus \sigma_{m,p}) = \sigma_{m,n+p} \qquad \sigma_{m,n}\,;\sigma_{n,m} = \mathrm{id}_{m+n}$$
$$(s \oplus \mathrm{id}_m)\,;\sigma_{m,n} = \sigma_{m,p}\,;(\mathrm{id}_m \oplus s)$$

Figure 1. Laws of symmetric monoidal categories instantiated to a PROP $(\mathbb{C}, \oplus, 0)$, with $m, n, p \in \mathbb{N}$ objects of $\mathbb{C}$ and s, t, u, v morphisms of $\mathbb{C}$ of the appropriate (co)arity. The laws express associativity of $;$ and $\oplus$, and how they interact with each other and with the identities. Also, they express that symmetries are natural and involutive.

the graphical language of *string diagrams* (Selinger 2011). When $\mathscr{E}$ is empty, we shall use notation $\mathbf{S}_\Sigma$ for the PROP presented by $(\Sigma, \mathscr{E})$.

The SMT of *special commutative Frobenius algebras* (which we shall usually refer to simply as Frobenius algebras, for brevity) plays a special role in our developments.

Example 2. (Frobenius Algebras). Consider the SMT $(\Sigma_{\mathbf{Frob}}, \mathscr{E}_{\mathbf{frob}})$, where

$$\Sigma_{\mathbf{Frob}} := \left\{ \;\; \text{⊸}\; : 2 \to 1, \;\; \bullet\!- : 0 \to 1, \;\; \text{⊸}\; : 1 \to 2, \;\; \multimap\!\bullet : 1 \to 0 \;\; \right\}$$

and $\mathscr{E}_{\mathbf{frob}}$ is the set consisting of the following three equations
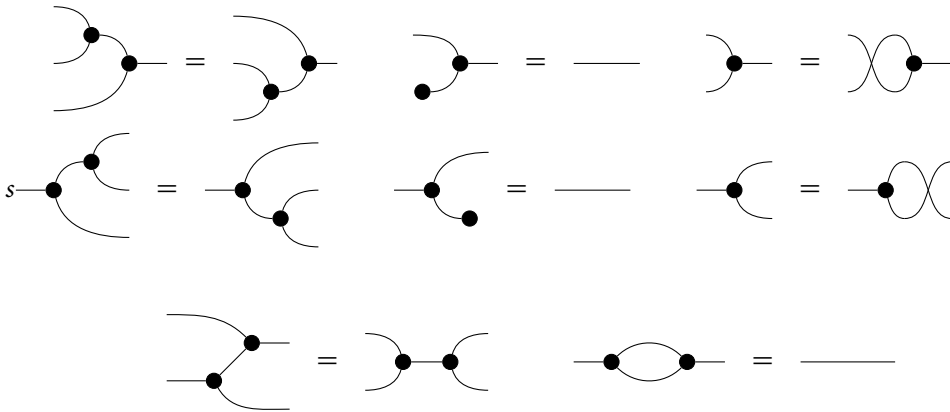
We use **Frob** to abbreviate the PROP $\mathbf{S}_{(\Sigma_{\mathbf{Frob}}, \mathscr{E}_{\mathbf{frob}})}$ presented by $(\Sigma_{\mathbf{Frob}}, \mathscr{E}_{\mathbf{Frob}})$.

As for regular algebraic theories, one may consider multi-coloured versions of SMTs and PROPs. What changes is the notion of signature, which is now a pair (C, Σ) consisting of a set C of *colours* and a set Σ of operations $o\colon w \to v$, with $w, v \in C^\star$ words over C.

Definition 3. (Coloured Prop). *Given a finite set C of colours, a C-coloured PROP $\mathbb{A}$ is a symmetric strict monoidal category where the set of objects $C^\star$ is the set of the finite words over C and the monoidal product on objects is word concatenation. A morphism from a C-coloured PROP $\mathbb{A}$ to a C'-coloured PROP $\mathbb{A}'$ is a symmetric strict monoidal functor $H\colon \mathbb{A} \to \mathbb{A}'$ acting on objects as a monoid homomorphism generated by a function $C \to C'$. Coloured PROPs and their morphisms form the category* CPROP.

As expected, PROP is the full sub-category of CPROP given by restricting to $\{c\}$-coloured PROPs, for a fixed colour c.

Example 4. For later use, we recall the multi-coloured analogous of Example 2, which is the theory of Frobenius algebras over a set C of colours. Its monoidal signature includes operations

$$\mathord{\succ}\!\!- : cc \to c \qquad \bullet\!\!- : \varepsilon \to c \qquad -\!\!\mathord{\prec} : c \to cc \qquad -\!\!\bullet : c \to \varepsilon$$

(where $\varepsilon \in C^{\star}$ denotes the empty word) and equations as in Example 2 for each colour $c \in C$. We write $\mathbf{Frob}_C$ for the C-coloured PROP presented by this SMT.

Remark 5. As observed in Bonchi et al. (2022), coproducts in CPROP work a bit differently than in PROP. Intuitively, a coproduct $\mathbb{C} + \mathbb{C}'$ in PROP will identify the common core of the two PROPs, i.e. the set of objects $\mathbb{N}$ and the symmetrical monoidal structure on these objects. Instead, a coproduct in CPROP will not make such identification, as the involved PROPs, say $\mathbb{D}$ and $\mathbb{D}'$, may be based on different sets of colours. However, if $\mathbb{D}$ and $\mathbb{D}'$ happen to be coloured from the same set C, we may still identify their common structure. Formally, this takes the form of a pushout, which we write $\mathbb{D} +_C \mathbb{D}'$. Such pushout is obtained from the span of the inclusion morphisms $\mathbb{D} \leftarrow \mathbf{P}_C \to \mathbb{D}'$, where $\mathbf{P}_C$ is the C-coloured PROP presented by the theory with an empty signature $(C, \varnothing)$ and no equations. One may think of $\mathbf{P}_C$ as having arrows $w \to v$ the permutations of w into v (thus arrows exist only when the word v is an anagram of the word w).

2.2 Syntactic rewriting for PROPs

Definition 6. *A rewriting rule in a PROP $\mathbb{A}$ consists of a pair of arrows $l, r \colon i \to j$ in $\mathbb{A}$ with the same arities and coarities, which we write as $\langle l, r \rangle$. Given $a, b \colon m \to n$ in $\mathbb{A}$, a rewrites into b via $\mathscr{R}$, written $a \Rightarrow_{\langle l,r \rangle} b$, if they are decomposable as follows*

(1)

In this situation, we say that a contains a redex *for $\langle l, r \rangle$. A rewriting system $\mathscr{R}$ is a set of rewriting rules, where we write $a \Rightarrow_{\mathscr{R}} b$ to mean there exists $\langle l, r \rangle \in \mathscr{R}$ such that $a \Rightarrow_{\langle l,r \rangle} b$.*

The equations $\mathscr{E}$ associated with an SMT $(\Sigma, \mathscr{E})$ can be oriented as rewriting rules. They give rise to a rewriting system, in the PROP $\mathbf{S}_\Sigma$ presented by $(\Sigma, \varnothing)$. Note that the decompositions (1) are equalities modulo the laws of Symmetric Monoidal Category (SMCs) (Figure 1). Thus, rewriting in a PROP always happens modulo these laws.

Notation 7. *Note that we write generic pairs and tuples using parentheses and reserve the notation $\langle l, r \rangle$ specifically for the case when the pair (l, r) forms a rewriting rule.*

2.3 Hypergraphs with interfaces

String diagrams in PROPs are interpreted as hypergraphs with interfaces, which we recall below.

Definition 8. (Hypergraphs). *A hypergraph G consists of a set $G_{\star}$ of nodes and, for each $k, l \in \mathbb{N}$, a (possibly empty) set of hyperedges $G_{k,l}$ with k (ordered) sources and l (ordered) targets of elements in $G_{\star}$, while a hypergraph morphisms $f \colon G \to H$ is a family of functions $\{f_{\star}, f_n \mid n \in \mathbb{N}\}$ satisfying the expected constraints.*

We denote by **Hyp** *the category of (finite) hypergraphs and hypergraph homomorphisms.*

Alternatively, and the characterisation will become useful later on, **Hyp** is the functor category $\mathbb{F}^{\mathbf{I}}$, where $\mathbf{I}$ has as objects pairs of natural numbers $(k, l) \in \mathbb{N} \times \mathbb{N}$ together with one extra object $\star$, and, for each $k, l \in \mathbb{N}$, there are $k + l$ arrows from (k, l) to $\star$.

Nodes will be drawn as dots and a (k, l) hyperedge h will be drawn as a rounded box, whose connections on the left represent the list $[s_1(h), \ldots, s_k(h)]$, ordered from top to bottom, and whose connections on the right represent the list $[t_1(h), \ldots, t_l(h)]$.

We now introduce hypergraphs with hyperedges typed in a monoidal signature Σ. First, define G_Σ as the hypergraph with just one node and for each $k, l \in \mathbb{N}$ the set of Σ-operations of arity k and coarity l as set of hyperedges with k sources and l targets. The category $\mathbf{Hyp}_\Sigma$ of Σ-*labelled hypergraphs* is the category whose objects consists of an hypergraph G together with a graph homomorphism $\lambda \colon G \to G_\Sigma$, which intuitively labels each hyperedge with an operation in Σ, while labelled graph homomorphisms are defined accordingly. We call such objects Σ-hypergraphs and we visualise them as hypergraphs whose hyperedges h are labelled by $\lambda(h)$. Observe that this definition ensures that a Σ-operation $o \colon n \to m$ labels a hyperedge only when it has n (resp. m) ordered input (resp. output) nodes.

Example 9. We show our notational conventions for labelled hypergraphs with the aid of an example. The hypergraph G has nodes $\{n_1, \ldots, n_8\}$, a $(3, 3)$-hyperedge h_1, a $(2, 1)$-hyperedge h_2 and a $(1, 0)$-hyperedge h_3, and the following source and target maps

$$
\begin{array}{llll}
s_1(h_1) := v_1 & t_1(h_1) := v_5 & s_1(h_2) := v_3 & \\
s_2(h_1) := v_2 & t_2(h_1) := v_6 \quad , & s_2(h_2) := v_4 \quad , & s_1(h_3) := v_6 \\
s_3(h_1) := v_3 & t_3(h_1) := v_6 & t_1(h_2) := v_8 &
\end{array}
$$

Also, suppose $\Sigma = \{o_1 \colon 3 \to 3, o_2 \colon 1 \to 0, o_3 \colon 2 \to 1\}$ is a monoidal signature and o_1, o_2, o_3 label the hyperedges of G of the matching type. Then G is drawn as follows
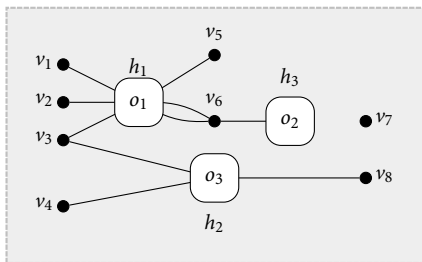
Arrows of a PROP will receive an interpretation as labelled hyergraphs *with interfaces*. The notion of interface is modelled using certain *cospans* in $\mathbf{Hyp}_\Sigma$.

Definition 10. (Hypergraphs with Interfaces). *A cospan* from G to G' in $\mathbf{Hyp}_\Sigma$ *is a pair of arrows* $G \xrightarrow{f} G'' \xleftarrow{g} G'$ *in* $\mathbf{Hyp}_\Sigma$, *where* G'' *is called the* carrier *of the cospan and* G, G' *are the* interfaces *of* G''. *Two cospans* $G \xrightarrow{f1} G_1 \xleftarrow{g1} G'$ *and* $G \xrightarrow{f2} G_2 \xleftarrow{g2} G'$ *are* isomorphic *when there is an isomorphism* $\alpha \colon G_1 \to G_2$ *in* $\mathbf{Hyp}_\Sigma$ *making the following diagram commute*

We define $\mathsf{Csp}(\mathbf{Hyp}_\Sigma)$ *as the category with the same objects as* $\mathbf{Hyp}_\Sigma$ *and arrows* $G \to G'$ *the cospans from* G *to* G', *up-to cospan isomorphism. Composition of* $G \to H \xleftarrow{f} G'$ *and* $G' \xrightarrow{g} H' \leftarrow G''$ *is* $G \to H +_{f,g} H' \leftarrow G''$, *obtained by taking the pushout of* f *and* g.[1]

We define $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ *as the full subcategory of the category of cospans in* $\mathsf{Csp}(\mathbf{Hyp}_\Sigma)$ *with objects the discrete hypergraphs (i.e. hypergraphs with empty set of hyperedges).*

Notation in Definition 10 follows the one introduced in Bonchi et al. (2022), where $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is presented as an instance of a more general construction $\mathsf{Csp}_H(\mathbb{C})$, for a given functor H and a category $\mathbb{C}$. Without going in full details, in the case of $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$, the subscript D is a functor with the role of selecting those cospans whose source and target (the interfaces) are *discrete* hypergraphs. This means that the objects of $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ are natural numbers, and it is in fact a PROP.

As with PROPs, we shall also consider the multi-coloured case of hypergraphs with interfaces. Given a set C of colours, a (multi-coloured) signature (C, Σ) can be encoded as an hypergraph $G_{C,\Sigma}$: the set of nodes is C and each operation $o: u \to v$ yields an hyperedge, with ordered input nodes forming the word $u \in C^\star$ and ordered output nodes forming the word $v \in C^\star$. We then define the category $\mathbf{Hyp}_{C,\Sigma}$ of (C, Σ)-*labelled hypergraphs* as the slice category $\mathbf{Hyp} \downarrow G_{C,\Sigma}$. Objects of $\mathbf{Hyp}_{C,\Sigma}$ can be visualised as hypergraphs with nodes labelled in C and hyperedges labelled in Σ, in a way that is compatible with the arity and coarity of operations in Σ.

Analogously to the one-coloured case, we can form the category $\mathsf{Csp}(\mathbf{Hyp}_{C,\Sigma})$ of cospans in $\mathbf{Hyp}_{C,\Sigma}$. We will work in $\mathsf{Csp}_{D_C}(\mathbf{Hyp}_{C,\Sigma})$, the full subcategory of $\mathsf{Csp}(\mathbf{Hyp}_{C,\Sigma})$ with objects the discrete hypergraphs. Note that $\mathsf{Csp}_{D_C}(\mathbf{Hyp}_{C,\Sigma})$ is a C-coloured PROP.

2.4 Double-pushout rewriting of hypergraphs

Double-pushout (DPO) rewriting (Corradini et al. 1997) is an algebraic approach to rewriting that, originally given for the category of graphs, can be defined in categories whose pushouts obey certain well-behavedness conditions, called *adhesive* categories (Lack and Sobociński 2005). Now, note that $\mathbf{Hyp}_\Sigma$ of Definition 8 can be abstractly characterised as $\mathbf{Hyp} \downarrow G_\Sigma$, i.e. the coslice of a presheaf category: this guarantees that it is adhesive (Bonchi et al. 2022), and thus we may apply DPO rewriting therein. In fact, in order to properly interpret string diagram rewriting, we will need a variation of DPO rewriting that takes into account interfaces. This variation, which we call DPO rewriting with interfaces (DPOI), has appeared in different guises in the literature, see e.g. Ehrig and König (2004), Gadducci (2007), Bonchi et al. (2009), Gadducci and Heckel (1998), Sassone and Sobociński (2005). DPOI provides a notion of rewriting for arrows $G \leftarrow J$ in $\mathbf{Hyp}_\Sigma$, which we write this way to emphasise that J acts as the interface of the hypergraph G, allowing G to be "glued" to a context. We now recall the definition of DPOI rewriting step.

Definition 11. (DPOI Rewriting). *Given* $G \leftarrow J$ *and* $H \leftarrow J$ *in* $\mathbf{Hyp}_\Sigma$, G *rewrites into* H *with interface* J — *notation* $(G \leftarrow J) \rightsquigarrow_{\mathscr{R}} (H \leftarrow J)$ — *if there exist rule* $L \leftarrow K \to R$ *in* $\mathscr{R}$ *and object* C *and cospan of arrows* $K \to C \leftarrow J$ *in* $\mathbf{Hyp}_\Sigma$ *such that the diagram below commutes and its marked squares are pushouts*

$$
\begin{array}{ccccc}
L & \longleftarrow & K & \longrightarrow & R \\
{\scriptstyle m}\downarrow & \ulcorner & \downarrow & \urcorner & \downarrow \\
G & \longleftarrow & C & \longrightarrow & H \\
& \nwarrow & \uparrow & \nearrow & \\
& & J & &
\end{array}
\tag{2}
$$

Typically, DPOI rewriting takes two distinct steps: first one computes from $K \to L \xrightarrow{m} G$ the object C and the arrows $K \to C \to G$ (called a *pushout complement*), then one pushes out the span $C \leftarrow K \to R$ to produce the rewritten object H, preserving the interface J.

Pushout complements always exist in $\mathbf{Hyp}_\Sigma$, but they are not necessarily unique. They are so if the rule is *left-linear*, that is, if $K \to L$ is monic. We will come back to this point in Section 4, as it plays an important role in giving a sound interpretation of string diagram rewriting as DPOI rewriting. For more details on the properties of pushout complements in $\mathbf{Hyp}_\Sigma$, we refer to Part I of this work (Bonchi et al. 2022, Section 4).

3. Combinatorial Characterisation of String Diagrams

Let us fix a monoidal signature Σ. In Bonchi et al. (2022), we gave an interpretation of the arrows of the PROP $\mathbf{S}_\Sigma$ in terms of cospans in $\mathbf{Hyp}_\Sigma$. We also saw that to make this interpretation an isomorphism, one needs to augment $\mathbf{S}_\Sigma$ with the structure of a Frobenius algebra. Formally, there are PROP morphisms

$$\mathbf{S}_\Sigma \xrightarrow{\ [\![\cdot]\!]\ } \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma) \xleftarrow{\ [\cdot]\ } \mathbf{Frob}$$

such that their copairing $\langle\!\langle\cdot\rangle\!\rangle \colon \mathbf{S}_\Sigma + \mathbf{Frob} \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is an isomorphism of PROPs (Bonchi et al. 2022, Theorem 3.9). For the purpose of this paper, it is convenient to recall the definition of both morphisms: it suffices to define how they act on the generators, as for arbitrary arrows their action is given by induction on the structure of PROPs. The morphism $[\![\cdot]\!] \colon \mathbf{S}_\Sigma \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ maps a generator $o \in \Sigma$ into the following cospan

(3)
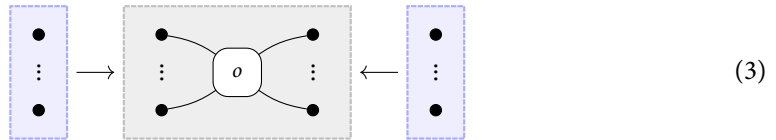
where the inputs (outputs) of the edge labelled o are in bijective correspondence with the nodes of the discrete graph on the left (on the right, respectively).

For the generators of $\mathbf{Frob}$, the morphism $[\cdot] \colon \mathbf{Frob} \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is defined as follows
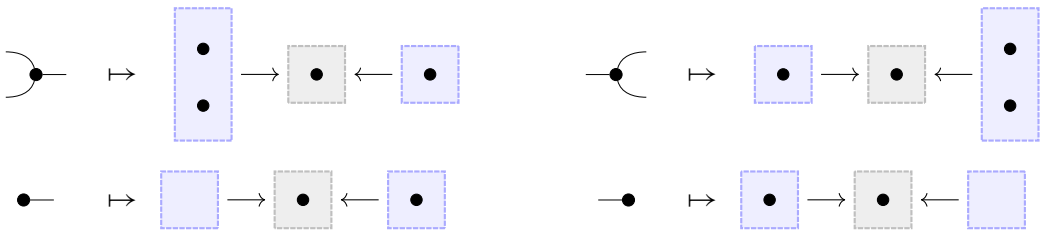
Note that $\langle\!\langle\cdot\rangle\!\rangle \colon \mathbf{S}_\Sigma + \mathbf{Frob} \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is defined on the generators of $\mathbf{S}_\Sigma + \mathbf{Frob}$, but it respects the laws of symmetric monoidal categories (Figure 1) and of Frobenius algebras (Example 2). Indeed, a major payoff of the combinatorial interpretation is that equivalent string diagrams are all interpreted as the same hypergraph with interfaces. We refer to Bonchi et al. (2022) for more discussion on this aspect.

In this paper, we plan to exploit $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ as a combinatorial domain where to interpret rewriting in $\mathbf{S}_\Sigma$. In the remainder of this section, we thus focus on $[\![\cdot]\!] \colon \mathbf{S}_\Sigma \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ and provide a combinatorial characterization of its image. A preliminary series of definitions introduces the relevant hypergraph notions: *monogamy* and *acyclicity*.

Definition 12. (Degree of a node). *The in-degree of a node v in hypergraph G is the number of pairs (h, i) where h is an hyperedge with v as its ith target. Similarly, the out-degree of v is the number of pairs (h, j) where h is an hyperedge with v as its jth source.*

Definition 13. (Monogamy). *Given $m \xrightarrow{f} G \xleftarrow{g} n$ in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$, let $\mathsf{in}(G)$ be the image of f and $\mathsf{out}(G)$ the image of g. We say that the cospan is* monogamous *iff f and g are mono and for all nodes v of G*

$$\text{the in-degree of } v \text{ is } \begin{cases} 0 & \text{if } v \in \mathsf{in}(G) \\ 1 & \text{otherwise.} \end{cases}$$

$$\text{the out-degree of } v \text{ is } \begin{cases} 0 & \text{if } v \in \mathsf{out}(G) \\ 1 & \text{otherwise} \end{cases}$$

We refer to the nodes in $\mathsf{in}(G)$ and $\mathsf{out}(G)$ as the inputs and the outputs of G and abusing notation we may say that G is monogamous. The cospan in (3) is clearly monogamous: all the nodes on the left are inputs and they have in-degree 0 and out-degree 1, while all the nodes on the right are outputs and they have in-degree 1 and out degree 0.

Example 14. Four examples of cospans that are *not* monogamous are displayed below. In here and the reminder of the paper, we use numeric labels when we wish to specify how the cospan legs are defined on the nodes.
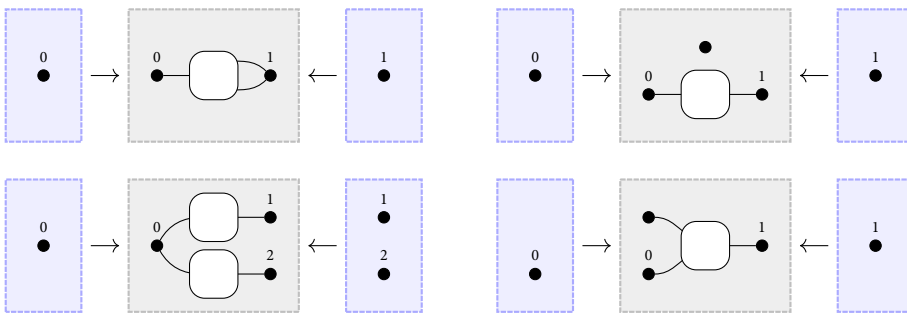
Lemma 15. *Identities and symmetries in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ are monogamous.*

Proof. The cospans identifying identities and symmetries involve discrete graphs, so the in-degree and the out-degree of all nodes are 0. Moreover, all these nodes are both inputs and outputs. □

Lemma 16. *Let $m \to G \leftarrow n$ and $n \to H \leftarrow o$ be arrows in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$. If both are monogamous cospans, then $(m \to G \leftarrow n)\,;(n \to H \leftarrow o)$ is monogamous.*

Proof. Since pushouts along monos in $\mathbf{Hyp}_\Sigma$ are mono, the morphisms of the cospans resulting from the composition $(m \to G \leftarrow n)\,;(n \to H \leftarrow o)$ are also mono. The condition on degrees is trivially preserved since $(m \to G \leftarrow n)\,;(n \to H \leftarrow o)$ is obtained by gluing together G with H along the nodes in n. This means that each of the nodes in $\mathsf{out}(G)$ is identified with exactly one of the node in $\mathsf{in}(H)$. □

Lemma 17. *Let $m_1 \to G_1 \leftarrow n_1$ and $m_2 \to G_2 \leftarrow n_2$ be arrows in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$. If both are monogamous cospans, then $(m_1 \to G_1 \leftarrow n_1) \oplus (m_2 \to G_2 \leftarrow n_2)$ is monogamous.*

Proof. By definition $(m_1 \to G_1 \leftarrow n_1) \oplus (m_2 \to G_2 \leftarrow n_2)$ is obtained by coproduct and therefore the degree of each node is the same as in the original graphs G_1 and G_2. Moreover, each node is an input iff it is an input in G_1 or in G_2 and it is an output iff it is an output in G_1 or in G_2. □

The notions of *acyclicity* and *(directed) path* between two nodes in a (directed) graph generalises to (directed) hypergraphs in the obvious way.

**Definition 18.** *For a pair of hyperedges h, h', we call h a predecessor of h' and h' a successor of h if there exists a node v in the target of h and in the source of h'.*

Definition 19. (Path). *For a hypergraph G and hyperedges h, h', a path p from h to h' is a sequence of hyperedges $[h_1, \ldots, h_n]$ such that $h_1 = h$, $h_n = h'$, and for $i < n$, h_{i+1} is a successor of h_i. We say p starts at a subgraph H if H contains a node in the source of h, and terminates at a subgraph H' if H' contains a node in the target of h'.*

By regarding nodes as single-node subgraphs, it clearly makes sense to talk about paths from/to nodes as well.

Definition 20. (Acyclicity). *A hypergraph G is acyclic if there exists no path from a node to itself. We also call a cospan $m \to G \leftarrow n$ acyclic if the property holds for G.*

Like for monogamy, it is easy to see that identities and symmetries are acyclic and that the monoidal product of acyclic cospans is acyclic. Unfortunately, the composition of two acyclic cospans might be cyclic: for instance by composing the following two acyclic cospans
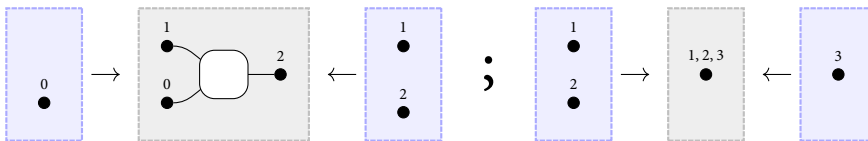
one obtains the cyclic cospan
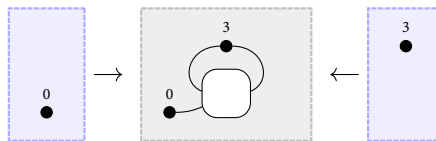
This issue can be avoided by additionally requiring the cospans to be monogamous.

Proposition 21. *Let $m \to G \leftarrow n$, $n \to H \leftarrow o$, $m_1 \to G_1 \leftarrow n_1$ and $m_2 \to G_2 \leftarrow n_2$ be monogamous acyclic cospans.*
  (1) Identities and symmetries in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ are monogamous acyclic.
  (2) $(m \to G \leftarrow n) \,;\, (n \to H \leftarrow o)$ is monogamous acyclic.
 *(3) $(m_1 \to G_1 \leftarrow n_1) \oplus (m_2 \to G_2 \leftarrow n_2)$ is monogamous acyclic.*

Proof. The first and the third points follow from what we discussed so far. The second point is the most interesting one. From Lemma 16, it follows immediately that $(m \to G \leftarrow n) \,;\, (n \to H \leftarrow o)$ is monogamous, so we only need to show that this is acyclic. Since both $m \to G \leftarrow n$ and $n \to H \leftarrow o$ are monogamous, their composition just identifies each of the nodes in $\mathsf{out}(G)$ with exactly one node in $\mathsf{in}(H)$. The identification of these nodes cannot create any cycle since there is no path in

G starting with one of these nodes (since their out-degree in G is 0) and there is no path in H arriving in one of these nodes (since their in-degree in H is 0). □

The above proposition informs us that monogamous acyclic cospans form a sub-PROP of $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$, which we call hereafter $\mathsf{MACsp}_D(\mathbf{Hyp}_\Sigma)$. The main result of this section (Theorem 25) shows that $\mathsf{MACsp}_D(\mathbf{Hyp}_\Sigma)$ is exactly the image of $\mathbf{S}_\Sigma$ through $[\![\cdot]\!]$. Its proof relies on an additional definition and a decomposition lemma.

Definition 22. (Convex sub-hypergraph). *A sub-hypergraph $H \subseteq G$ is convex if, for any nodes v, v' in H and any path p from v to v' in G, every hyperedge in p is also in H.*

Example 23. Consider the following hypergraph
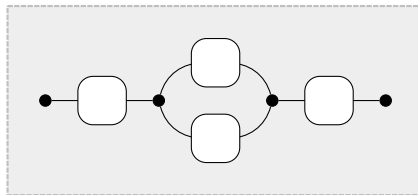
Below on the left and on the right are illustrated a convex and a non-convex sub-hypergraph
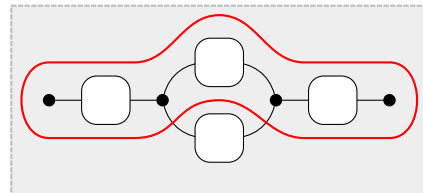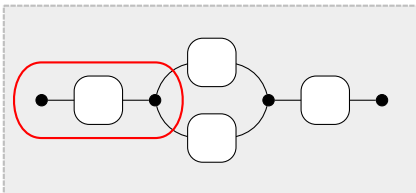
Lemma 24. (Decomposition). *Let $m \to G \leftarrow n$ be a monogamous acyclic cospan and L a convex sub-hypergraph of G. Then there exist $k \in \mathbb{N}$ and a unique cospan $i \to L \leftarrow j$ such that G factors as*

$$(4)$$

where all cospans in (4) are monogamous acyclic.

Proof. Let C_1 be the smallest sub-hypergraph containing the inputs of G and every hyperedge h that is not in L, but has a path to it. Let C_2 then be the smallest sub-hypergraph containing the outputs of G such that $C_1 \cup L \cup C_2 = G$. By construction, C_1 and L share no hyperedges. Should C_2 share a hyperedge with C_1 or L, then a smaller C_2' would exist such that $C_1 \cup L \cup C_2' = G$, so C_2 shares no hyperedges with either C_1 or L. Hence, the three sub-hypergraphs only overlap on nodes. Now let

$$i := C_{1*} \cap L_*$$
$$j := C_{2*} \cap L_*$$
$$k := (C_{1*} \cap C_{2*}) \backslash L_*$$

where L_* are the nodes of hypergraph L and the same for C_1 and C_2. Pictorially, these sub-hypergraphs are defined as follows
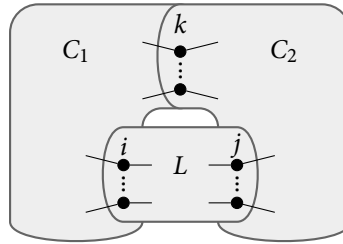
Now, define the following cospans, where arrows are all inclusions

$$m \to C_1 \leftarrow k + i$$
$$i \to L \leftarrow j$$
$$k + j \to C_2 \leftarrow n$$

Then (4) is computed as the colimit of the following diagram

$$m \to C_1 \leftarrow k + i \to k + L \leftarrow k + j \to C_2 \leftarrow n$$

The two spans identify precisely those nodes from G that occur in more than one sub-hypergraph, so this amounts to simply taking the union

$$m \to C_1 \cup L \cup C_2 \leftarrow n \;=\; m \to G \leftarrow n$$

Now C_1, C_2 and L are acyclic because G is, so it only remains to show that each of these cospans is monogamous. For C_1 and C_2 it follows straightforwardly from the observation that, by construction, C_1 is closed under predecessors and C_2 under successors. The interesting case is L, which relies on convexity. Suppose v has no in-hyperedge in L. Then either v is an input or there exists a hyperedge with a path to v. One of these two is true precisely when $v \in i$. Suppose v has no out-hyperedge in L. Then, either v is an ouput or it has an out-hyperedge in C_1 or C_2. But if it has an out-hyperedge h in C_1, then there is a path from v to another node v', going through h. By convexity, h must then be in L, which is a contradiction. Hence, $v \in C_2$, which is true if and only if $v \in j$. □

Theorem 25. *A cospan $n \to G \leftarrow m$ is in the image of $[\![\cdot]\!] \colon \mathbf{S}_\Sigma \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ if and only if $n \to G \leftarrow m$ is monogamous acyclic.*

Proof. The only if direction follows by induction on the arrows of $\mathbf{S}_\Sigma$: for the base case, it is immediate to check that (3) is monogamous acyclic, while the inductive cases follow from Proposition 21.

For the converse direction, we can reason by induction on the number of hyperedges in G. If G does not contain any, then monogamy and acyclicity imply that $n \to G$ and $m \to G$ are bijections, so that $n \to G \leftarrow m$ is in the image of an arrow only consisting of identities and symmetries. For the inductive step, pick any hyperedge e of G. Recall that e has a label $l(e) \in \Sigma$ and that $l(e)$ is an arrow of $\mathbf{S}_\Sigma$. By monogamy and acyclicity, $[\![l(e)]\!]$ is a convex sub-hypergraph of G. Hence, by Lemma 24, $n \to G \leftarrow m$ factors as (4), with L being $[\![l(e)]\!]$. The lemma guarantees that all the above cospans are monogamous acyclic. Therefore, by the inductive hypothesis they are in the image of $[\![\cdot]\!]$, and so the same holds for $n \to G \leftarrow m$. □

The following result (Corollary 3.11 in Bonchi et al. 2022) proves that $[\![\cdot]\!] \colon \mathbf{S}_\Sigma \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is faithful, so an immediate consequence of the above theorem is that $\mathbf{S}_\Sigma$ is isomorphic to the sub-PROP of $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ of monogamous acyclic cospans.

Corollary 26. $\mathbf{S}_\Sigma \cong \mathsf{MACsp}_D(\mathbf{Hyp}_\Sigma)$.

3.1 Characterisation for coloured PROPs

At the beginning of this section, we recalled from Bonchi et al. (2022) that $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is isomorphic to $\mathbf{S}_\Sigma + \mathbf{Frob}$. The isomorphism extends in the obvious way to the coloured case: Proposition 3.12 in Bonchi et al. (2022) states that $\mathsf{Csp}_{D_C}(\mathbf{Hyp}_{C,\Sigma})$ is isomorphic to $\mathbf{S}_{C,\Sigma} +_C \mathbf{Frob}_C$ (where $+_C$ and $\mathbf{Frob}_C$ are defined as in Example 4 and Remark 5). The same holds for Theorem 25. The definition of $[\![\cdot]\!]_C$ given in Bonchi et al. (2022) is the same as the one in (3), but with the proper interpretation of colours as labels of nodes. The definition of monogamous acyclic cospans (Definitions 13 and 20) does not change: the notions of degree and path are exactly the same in coloured and non coloured hypergraphs. All the results proved above hold straightforwardly by following the same proofs. In particular, we have the following.

Theorem 27. *A cospan $w \to G \leftarrow v$ is in the image of $[\![\,\cdot\,]\!]_C \colon \mathbf{S}_{C,\Sigma} \to \mathsf{Csp}_{D_C}(\mathbf{Hyp}_{C,\Sigma})$ if and only if $w \to G \leftarrow v$ is monogamous acyclic.*

4. A Sound and Complete Interpretation for String Diagram Rewriting

In this section, we develop a version of DPOI rewriting that is sound and complete for symmetric monoidal categories that *do not* come with a chosen Frobenius algebra on each object. Recall that, as shown in Bonchi et al. (2022), DPOI rewriting for hypergraphs (Definition 11) corresponds exactly to the rewriting for a SMT Σ, modulo Frobenius structure.

Before formally stating this correspondence (Theorem 28 below), we recall from Bonchi et al. (2022) the notation $\ulcorner d \urcorner$, which refers to the 'rewiring' of a syntactic term d, turning all of the inputs into outputs

Working with 'rewired' graphs is equivalent to working with the original ones, in the sense that $d \Rightarrow_{\langle l,r \rangle} e$ if and only if $\ulcorner d \urcorner \Rightarrow_{\langle \ulcorner l \urcorner, \ulcorner r \urcorner \rangle} \ulcorner e \urcorner$. However, since the rewired rules have only one boundary, they are readily interpreted as hypergraphs with interfaces. That is, if d corresponds to a cospan $i \to G \leftarrow j$, then $\ulcorner d \urcorner$ corresponds to $0 \to G \leftarrow i + j$, or simply $G \leftarrow i + j$.

Similarly, a syntactic rewriting rule $\langle \ulcorner l \urcorner, \ulcorner r \urcorner \rangle$ corresponds to a pair of hypergraphs with the same interface, $L \leftarrow i + j$ and $R \leftarrow i + j$, i.e. a span $L \leftarrow i + j \to R$. Hence, we can extend the definition of $\langle\!\langle \cdot \rangle\!\rangle \colon \mathbf{S}_\Sigma + \mathbf{Frob} \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ (*cf.* Section 3) to rewriting rules by letting $\langle\!\langle \langle \ulcorner l \urcorner, \ulcorner r \urcorner \rangle \rangle\!\rangle := L \leftarrow i + j \to R$. We now have all the ingredients to recall the correspondence theorem between (syntactic) rewriting modulo Frobenius relation $\Rightarrow$ and the DPOI rewriting relation $\rightsquigarrow$.

Theorem 28. (Bonchi et al. 2022). *Let $\langle l, r \rangle$ be a rewriting rule on $\mathbf{S}_\Sigma + \mathbf{Frob}$. Then*

$$d \Rightarrow_{\langle l,r \rangle} e \quad \textit{iff} \quad \langle\!\langle \ulcorner d \urcorner \rangle\!\rangle \rightsquigarrow_{\langle\!\langle \langle \ulcorner l \urcorner, \ulcorner r \urcorner \rangle \rangle\!\rangle} \langle\!\langle \ulcorner e \urcorner \rangle\!\rangle \, .$$

In this section, we will see that the full DPOI relation $\rightsquigarrow$ is not sound for rewriting in the absence of Frobenius structure. To fix this problem, we will put a restriction on which pushout complements (i.e. contexts) are allowed in a rewriting step. If a rule is not left-linear, there could be many different pushout complements for a given match, each one yielding a different result. It was shown in Bonchi et al. (2022) that this makes perfect sense when rewriting modulo Frobenius structure. However, without this structure not all results can be interpreted in a symmetric monoidal category.

Example 29. Consider $\Sigma = \{\alpha_1 : 0 \to 1, \alpha_2 : 1 \to 0, \alpha_3 : 1 \to 1\}$ and the PROP rewriting system

$\mathcal{R} = \left\{ \begin{array}{ccc} \text{———} & \Rightarrow & \boxed{\alpha_3} \end{array} \right\}$ on $\mathbf{S}_\Sigma$. Its interpretation in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is given by the rule
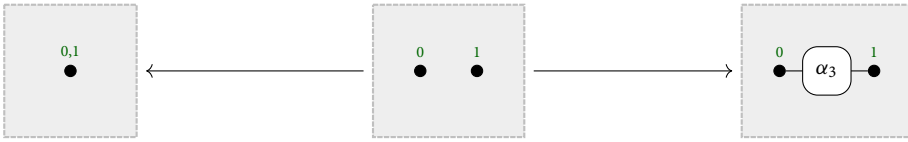
The rule is not left-linear and therefore pushout complements are not necessarily unique for the application of this rule. For example, the following pushout complement yields a rewritten graph that can be interpreted as an arrow in an SMC
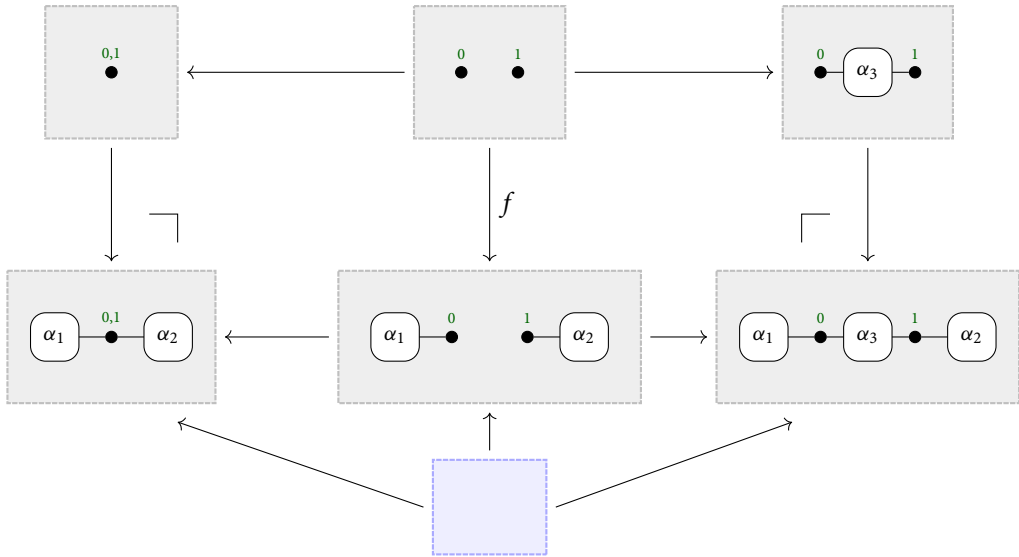
On the other hand, if we choose a different pushout complement, we obtain a rewritten graph that does not look like an SMC morphism
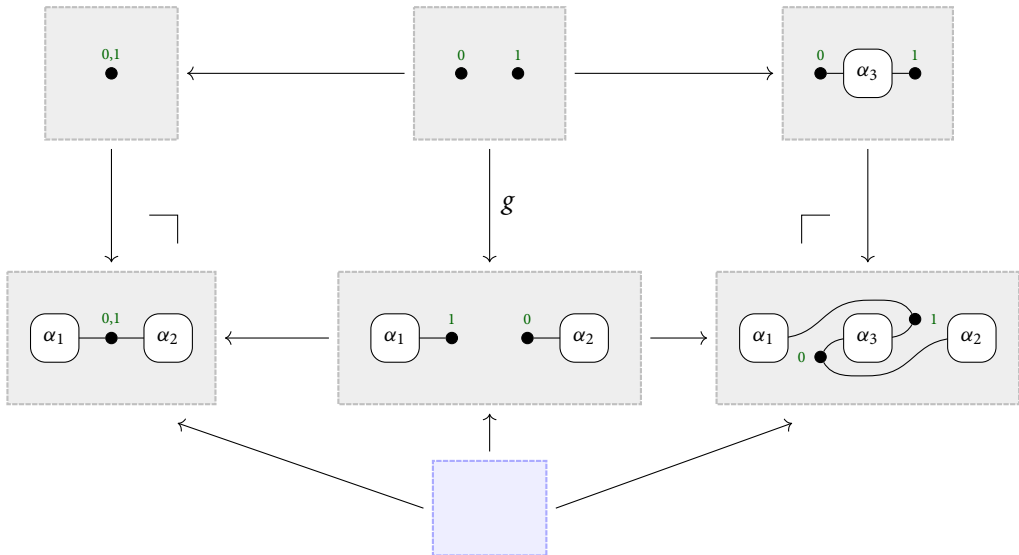
The different outcome is due to the fact that f maps 0 to the leftmost and 1 to the rightmost node, whereas g swaps the assignments. Even though both rewriting steps could be mimicked at the syntactic level in $\mathbf{S}_\Sigma + \mathbf{Frob}$, the second hypergraph rewrite yields a hypergraph that is not in the image of any morphism of $\mathbf{S}_\Sigma$. In particular, the rewritten graph in the second derivation is not monogamous: the outputs of α_1 and α_3 and the inputs of α_2 and α_3 have been glued together by the right pushout.

To rule out 'bad' pushout complements, i.e. those not yielding monogamous acyclic hypergraphs after rewriting, we introduce the notion of *boundary complement*, which requires that inputs only ever get glued to outputs (and vice-versa) in the two pushout squares of a DPO diagram.

Definition 30. (Boundary complement). *For monogamous cospans $i \xrightarrow{a_1} L \xleftarrow{a_2} j$ and $n \xrightarrow{b_1} G \xleftarrow{b_2} m$ and mono $f : L \to G$, a pushout complement as depicted in (†) below*
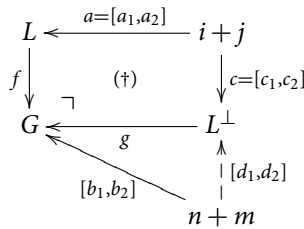
is called a boundary complement if $[c_1, c_2]$ is mono and there exist $d_1 : n \to L^\perp$ and $d_2 : m \to L^\perp$ making the above triangle commute and such that

$$n + j \xrightarrow{[d_1, c_2]} L^\perp \xleftarrow{[d_2, c_1]} m + i \tag{5}$$

is a monogamous cospan.

Intuitively, being a pushout complement, $L^\perp$ can be figured as G with a 'hole', filled by L. As G and L are both monogamous. This means that in $\mathbf{S}_\Sigma$ we have
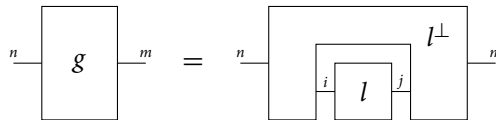
where $g : n \to m$, $l : i \to j$, and $l^\perp : n + j \to m + i$ such that $[\![g]\!] = n \leftarrow G \to m$, $[\![l]\!] = i \leftarrow L \to j$ and $[\![l^\perp]\!] = n + j \leftarrow L^\perp \to m + i$ are guaranteed to exist by monogamicity of the three cospans involved and Theorem 25. Note that, in particular, the requirement that (5) is monogamous enforces that the string diagram
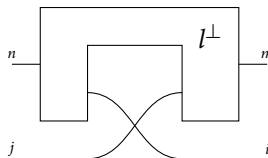
properly lives in $\mathbf{S}_\Sigma$ instead of $\mathbf{S}_\Sigma + \mathbf{Frob}$.

The notion of boundary complement has the pleasant property of restoring uniqueness of pushout complements, even though we consider some rules that are not left-linear (namely, those with an identity morphism on the left-hand side of the syntactic rule).

Proposition 31. *When boundary complements in* $\mathbf{Hyp}_\Sigma$ *exist, they are unique.*

Proof. Since $\mathbf{Hyp}_\Sigma$ is a presheaf category, a pushout of hypergraphs consists of pushouts on the underlying sets of nodes and hyperedges and an appropriate choice of source and target maps. Hence, the underlying sets of $L^\perp$ must give pushout complements in the category $\mathbf{F}$ of finite sets

$$
\begin{array}{ccc}
L_\star & \xleftarrow{\ a_\star\ } & i+j \\
{\scriptstyle f_\star}\downarrow & \ulcorner & \downarrow{\scriptstyle c_\star} \\
G_\star & \xleftarrow[g_\star]{} & L_\star^\perp
\end{array}
\qquad
\begin{array}{ccc}
L_{k,l} & \xleftarrow{\quad} & 0 \\
{\scriptstyle f_{k,l}}\downarrow & \ulcorner & \downarrow \\
G_{k,l} & \xleftarrow[g_{k,l}]{} & L_{k,l}^\perp
\end{array}
$$

where $G_\star$ and $G_{k,l}$ are the nodes and the (k, l)-hyperedges of G, and similarly for L and $L^\perp$. Since $i+j$ is discrete, $G_{k,l}$ is a disjoint union, so $L_{k,l}^\perp$ must be $G_{k,l}\backslash L_{k,l}$. Since these are diagrams in $\mathbf{F}$, and c, f are mono, we can rewrite the left pushout square as

$$
\begin{array}{ccc}
l+x & \xleftarrow{\ a_\star\ } & i+j \\
{\scriptstyle f_\star}\downarrow & \ulcorner & \downarrow{\scriptstyle c_\star} \\
l+x+y & \xleftarrow[g_\star]{} & i+j+z
\end{array}
$$

where the two downward arrows are coproduct injections and l is the image of $a_\star$. One can easily verify that $l+x+z$ also gives a pushout for the given span, so we obtain a commuting isomorphism $l+x+y \cong l+x+z$, from which we conclude $y \cong z$ and that, up to isomorphism, the pushout complement on nodes must be

$$
\begin{array}{ccc}
l+x & \xleftarrow{\ a_\star\ } & i+j \\
{\scriptstyle f_\star}\downarrow & \ulcorner & \downarrow{\scriptstyle c_\star} \\
l+x+y & \xleftarrow[g_\star]{} & i+j+y
\end{array}
$$

from whence it follows that $g_\star = a_\star + id_n$.

So far, we have proved that the sets $L_\star^\perp$ and $L_{k,l}^\perp$ are defined uniquely by the property of being a pushout complement. The only thing that remains is the definition of the source and target maps $s_{k,l}\colon L_{k,l}^\perp \to L_\star^\perp$ and target $t_{k,l}\colon L_{k,l}^\perp \to L_\star^\perp$ maps. Since g is a homomorphism, by abusing notation and denoting as $g_\star$ also its extension to sequences, we have that for all hyperedges h

$$g_\star(s_{k,l}(h)) = s_{k,l}(g_{k,l}(h)) \implies s_{k,l}(h) \in g_\star^{-1}(s_{k,l}(g_{k,l}(h)))$$

Since $g_\star$ is of the form $[(a_1)_\star, (a_2)_\star] + 1_n$, where a_1 and a_2 are mono, the inverse image $g_\star^{-1}(s_{k,l}(g_{k,l}(h)))$ contains at most two elements. In the case where it has one element, $s_{k,l}$ is uniquely fixed, so consider when it has two. It must then be the case that

$$g_\star^{-1}(s_{k,l}(g_{k,l}(h))) = \{v_1 \in i, v_2 \in j\}$$

But monogamy of (5) says that the image of i in $L^\perp$ cannot be the source of any hyperedge. Therefore, it must be $s_{k,l}(h) = v_2$. Similarly, if

$$g_\star^{-1}(t_{k,l}(g_{k,l}(h))) = \{v_1 \in i, v_2 \in j\}$$

then $t_{k,l}(h)$ must be v_1. Since there is at most one choice of source and target maps for $L^\perp$ making g a homomorphism, $L^\perp$ must be unique. $\qquad\square$

Boundary complements solve the problem highlighted in Example 29, in that restricting to boundary complements guarantees that the result of a rewrite will be a monogamous acyclic hypergraph, which can be interpreted as morphisms in an SMC. However, a slightly more subtle problem remains: some graph rewrites can be performed in such a way that the rule, target graph, and rewritten hypergraph are all monogamous acyclic, but they correspond to an equation between morphisms that is not derivable using the SMC laws and the rules of a SMT.

Example 32. Consider a $\Sigma = \{e_1 \colon 1 \to 2, e_2 \colon 2 \to 1, e_3 \colon 1 \to 1, e_4 \colon 1 \to 1\}$ and the following rewriting rule in $\mathbf{S}_\Sigma$

$$\left\langle \quad \boxed{e_1} \!\!\!\times\!\!\! \boxed{e_2} \;\; : 2 \to 2\,, \quad \times \begin{matrix} \boxed{e_4} \\ \boxed{e_4} \end{matrix} \;\; : 2 \to 2 \right\rangle \tag{6}$$

Left and right side are interpreted in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ as cospans
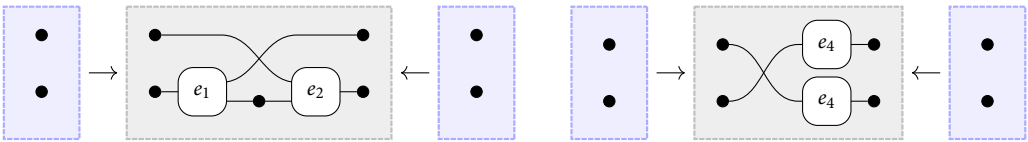
We introduce another diagram $c \colon 1 \to 1$ in $\mathbf{S}_\Sigma$ and its interpretation in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$
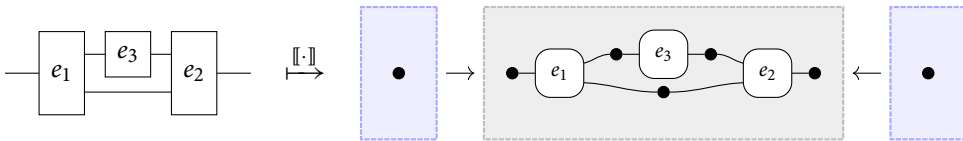
Now, rule (6) cannot be applied to c, even modulo the SMC equations. However, their interpretation yields a DPO rewriting step in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ as below
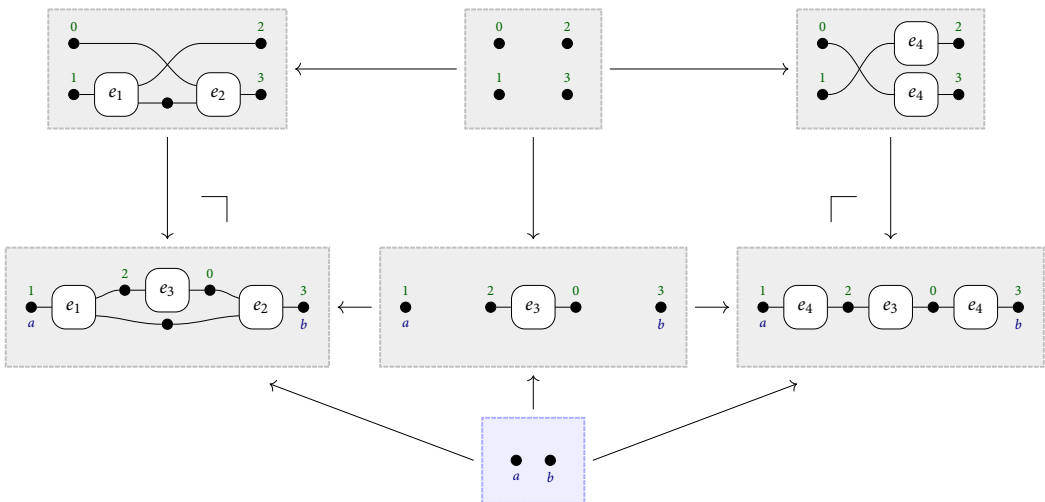
Observe that the leftmost pushout above *is* a boundary complement: the input–output partition is correct. Still, the rewriting step cannot be mimicked at the syntactic level using rewriting modulo the SMC laws. That is because, in order to apply our rule, we need to deform the diagram such that e_3 occurs outside of the left-hand side. This requires moving e_3 either before or after the

occurrence of the left-hand side in the larger expression, but both of these possibilities require a feedback loop
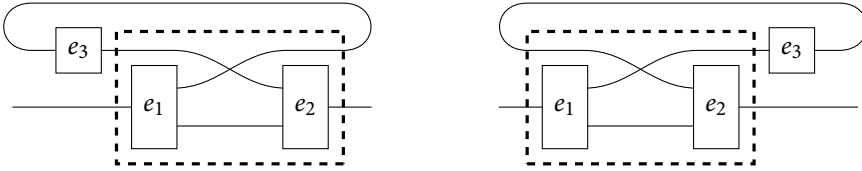
Hence, if the category does not have at least a traced symmetric monoidal structure (Joyal et al. 1996), there is no way to apply the rule.

The source of the problem in Example 32 is the fact that the image of the match f forms a non-convex, 'U-shaped' sub-graph of the target graph. In other words, we identify a forward-directed path of hyperedges going out of the image of f and back inside again. Hyperedges in such a path (namely e_3 in the example) cause obstructions to rewriting in an SMC. Hence, we introduce the notion of *convex* matches, which forbid forward-directed paths from outputs to inputs.

Definition 33. (Convex match). *We call* $m : L \to G$ *in* $\mathbf{Hyp}_\Sigma$ *a convex match if it is mono and its image is convex.*

We saw from Lemma 24 that, for any convex sub-graph L of a monogamous acyclic hypergraph G, G can be decomposed into parts using '$\oplus$' and '$;$', where one of those parts is L. This will play a crucial role in our soundness theorem.

We now combine the notions of boundary complement and of convex match to tailor a family of DPOI rewriting steps which only yield legal $\mathbf{S}_\Sigma$-rewriting.

Definition 34. *Given* $D \leftarrow n + m$ *and* $E \leftarrow n + m$ *in* $\mathbf{Hyp}_\Sigma$, *D rewrites convexely into E with interface* $n + m$ — *notation* $(D \leftarrow n + m) \Rrightarrow_{\mathscr{R}} (E \leftarrow n + m)$ — *if there exist rule* $L \leftarrow i + j \to R$ *in* $\mathscr{R}$ *and object C and cospan arrows* $i + j \to C \leftarrow n + m$ *in* $\mathbf{Hyp}_\Sigma$ *such that the diagram below commutes and its marked squares are pushouts*

$$
\begin{array}{ccccc}
L & \xleftarrow{[a_1,a_2]} & i+j & \xrightarrow{[b_1,b_2]} & R \\
{\scriptstyle f}\downarrow & \ulcorner & \downarrow & \urcorner & \downarrow \\
D & \longleftarrow & C & \longrightarrow & E \\
& {\scriptstyle [q_1,q_2]}\nwarrow & \uparrow & \nearrow {\scriptstyle [p_1,p_2]} & \\
& & n+m & &
\end{array}
\tag{7}
$$

and the following conditions hold

- *$f : L \to D$ is a convex match;*
- *$i + j \to C \to D$ is a boundary complement in the leftmost pushout.*

The relation $\Rrightarrow_{\mathscr{R}}$ is contained in the DPOI rewriting relation $\leadsto_{\mathscr{R}}$ (Definition 11), the difference being that the leftmost pushout must consist of a convex match and a boundary complement.

We have now all the ingredients to prove the adequacy of convex DPO rewriting with respect to rewriting in $\mathbf{S}_\Sigma$.

Theorem 35. *Let $\mathscr{R}$ by any rewriting system on $\mathbf{S}_\Sigma$. Then,*

$$
d \Rrightarrow_{\mathscr{R}} e \ \ \textit{iff} \ \ \langle\!\langle \ulcorner d \urcorner \rangle\!\rangle \Rrightarrow_{\langle\!\langle \ulcorner \mathscr{R} \urcorner \rangle\!\rangle} \langle\!\langle \ulcorner e \urcorner \rangle\!\rangle.
$$

Proof. For the only if direction, by Theorem 28 $d \Rightarrow_{\mathscr{R}} e$ implies $\langle\!\langle \ulcorner d \urcorner \rangle\!\rangle \rightsquigarrow_{\langle\!\langle \ulcorner \mathscr{R} \urcorner \rangle\!\rangle} \langle\!\langle \ulcorner e \urcorner \rangle\!\rangle$. One may check that the argument constructs a *convex* DPO rewriting step, thus yielding the desired statement.

More in detail, our assumption gives that

$$n - \boxed{d} - m \;=\; n - \boxed{c_1} \; \substack{k \\ i \; \boxed{l} \; j} \; \boxed{c_2} - m$$

$$n - \boxed{e} - m \;=\; n - \boxed{c_1} \; \substack{k \\ i \; \boxed{r} \; j} \; \boxed{c_2} - m$$
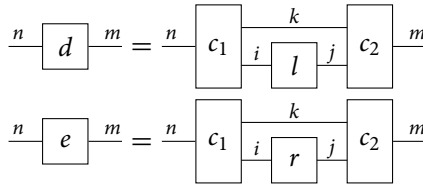
and hence the following equalities hold in $\mathbf{S}_\Sigma + \mathbf{Frob}$, where $c_i^\star$, $i \in \{1, 2\}$, is notation for $\boxed{c_i}$ (with a cap/cup)

$$\overline{n \; \boxed{d} \; m} \;=\; \left(\substack{i \\ k \\ i \; \boxed{l} \; j} \; \substack{c_1^* - n \\ \\ c_2 - m} \right) \tag{8}$$

$$\overline{n \; \boxed{e} \; m} \;=\; \left(\substack{i \\ k \\ i \; \boxed{r} \; j} \; \substack{c_1^* - n \\ \\ c_2 - m} \right) \tag{9}$$

We now define

$$\left(0 \to D \xleftarrow{[q_1, q_2]} n + m \right) := \langle\!\langle \ulcorner d \urcorner \rangle\!\rangle = \left\langle\!\!\left\langle \; \overline{n \; \boxed{d} \; m} \; \right\rangle\!\!\right\rangle$$

$$\left(0 \to E \xleftarrow{[p_1, p_2]} n + m \right) := \langle\!\langle \ulcorner e \urcorner \rangle\!\rangle = \left\langle\!\!\left\langle \; \overline{n \; \boxed{e} \; m} \; \right\rangle\!\!\right\rangle$$

$$\left(0 \to L \xleftarrow{[a_1, a_2]} i + j \right) := \langle\!\langle \ulcorner l \urcorner \rangle\!\rangle = \left\langle\!\!\left\langle \; \overline{i \; \boxed{l} \; j} \; \right\rangle\!\!\right\rangle$$

$$\left(0 \to R \xleftarrow{[b_1, b_2]} i + j \right) := \langle\!\langle \ulcorner r \urcorner \rangle\!\rangle = \left\langle\!\!\left\langle \; \overline{i \; \boxed{r} \; j} \; \right\rangle\!\!\right\rangle$$

$$\left(i + j \to C \leftarrow n + m \right) := \left\langle\!\!\left\langle \; \substack{i \\ k \\ j} \; \substack{c_1^* - n \\ \\ c_2 - m} \; \right\rangle\!\!\right\rangle$$

By these definitions and (8)–(9) it follows that

$$(0 \to D \leftarrow n + m) = (0 \to L \leftarrow i + j) \; ; \; (i + j \to C \leftarrow n + m)$$
$$(0 \to E \leftarrow n + m) = (0 \to R \leftarrow i + j) \; ; \; (i + j \to C \leftarrow n + m).$$

Since composition of cospans is defined by pushout, we have a commutative diagram with two pushouts as in (7). It remains to check that the match $L \to D$ is convex and that C is a boundary

complement: these conditions can be verified by definition of the involved components. Therefore, $\langle\!\langle\ulcorner d\urcorner\rangle\!\rangle \Rightarrow_{\langle\!\langle\ulcorner\mathscr{R}\urcorner\rangle\!\rangle} \langle\!\langle\ulcorner e\urcorner\rangle\!\rangle$ by application of the rule $\langle\langle\!\langle\ulcorner l\urcorner\rangle\!\rangle, \langle\!\langle\ulcorner r\urcorner\rangle\!\rangle\rangle$.

We now turn to the converse direction. Let

$$\langle\!\langle\ulcorner d\urcorner\rangle\!\rangle =: \quad {}_0\nearrow {}^{D}\searrow^{[q_1,q_2]}_{n+m} \qquad \langle\!\langle\ulcorner e\urcorner\rangle\!\rangle =: \quad {}_0\nearrow {}^{E}\searrow^{[p_1,p_2]}_{n+m} \ .$$

Our assumption gives us a diagram as in (7), with application of a rule $\langle\langle\!\langle\ulcorner l\urcorner\rangle\!\rangle, \langle\!\langle\ulcorner r\urcorner\rangle\!\rangle\rangle$ in $\langle\!\langle\ulcorner\mathscr{R}\urcorner\rangle\!\rangle$. We now want to show that $d \Rightarrow_{\mathscr{R}} e$ with rule $\langle l, r\rangle$, say of type (i, j). Now, because $n \xrightarrow{q_1} D \xleftarrow{q_2} m = [\![d]\!]$, it is monogamous acyclic by Theorem 25. Since the match $f : L \to D$ in (7) is convex, Lemma 24 yields a decomposition of $n \xrightarrow{q_1} D \xleftarrow{q_2} m$ in terms of monogamous acyclic cospans

$$\left(n \to C_1 \leftarrow i{+}k\right) ; \quad \begin{array}{c} \left(k \xrightarrow{id} k \xleftarrow{id} k\right) \\ \oplus \\ \left(i \to L \leftarrow j\right) \end{array} ; \left(j{+}k \to C_2 \leftarrow m\right).$$

Applying again Theorem 25 we obtain c_1, c_2 in $\mathbf{S}_\Sigma$ such that

$$[\![c_1]\!] = n \to C_1 \leftarrow i + k \qquad [\![c_2]\!] = j + k \to C_2 \leftarrow m.$$

By functoriality of $[\![\cdot]\!]$, $[\![d]\!] = [\![c_1 ; (id \oplus l) ; c_2]\!]$ and, since $[\![\cdot]\!]$ is a faithful PROP morphism, $d = c_1 ; (id \oplus l) ; c_2$. Thus, we can apply the rule $\langle l, r\rangle$ on e, which yields $e = c_1 ; (id \oplus r) ; c_2$ such that $d \Rightarrow_{\mathscr{R}} e$. We can conclude that $[\![e]\!] = n \xrightarrow{p_1} E \xleftarrow{p_2} m$ because boundary complements are unique (Proposition 31). $\qquad\square$

Hence, we have shown soundness and adequacy of convex DPOI rewriting for SMTs. In other words, whenever we want to perform rewriting in a free symmetric monoidal category, we could just as well do convex DPOI.

Remark 36. A natural question is ask is whether we can do convex DPOI rewriting efficiently. This is not obvious since computing the match in a DPOI step involves solving a subgraph isomorphism problem and, as we saw in Section 4.5 of Part 1 (Bonchi et al. 2022), enumerating pushout complements can require a substantial amount of computation for general. The issue with matches is not really a problem since we consider rewriting with rules whose left-hand side is of fixed constant size, which is typically much smaller than the target graph. In this regime, efficient subgraph isomorphism algorithms exist going back (at least) to Ullmann (1976) and can be easily adapted to our setting.

In fact, we can do even better in the case of monogamous hypergraphs. One can construct a homomorphism $m : L \to G$ by traversing the nodes and hyperedges of L and mapping them one-by-one. At each step in the traversal, the image of the next node (resp. hyperedge) is uniquely fixed by the image of an adjacent hyperedge (resp. node), so the match will be uniquely fixed by the image of a single node in each connected component of L. Hence, if L is connected, we can fix a starting node v in L and check if for each node v' in G the setting $m(v) := v'$ yields a valid match in time linear in L. Once a match is found, we can check convexity by computing the successors of the outputs of L in G and checking whether any input of L is contained in that set, which has worse-case complexity $O(|G_\star|)$, since we need to visit each node in G at most once. That is, we can enumerate matches of L and G in $O(|L_\star||G_\star|^2)$ time.

The second issue is solved for convex DPOI by requiring pushout complements to be boundary complements, which are unique, as we saw in Proposition 31. Whereas in the general case, we may have to search an exponential space of potential pushout complements, boundary complements

Mathematical Structures in Computer Science   531

force the fact that there is at most one solution, which can be constructed efficiently from the maps $i + j \to L \to G$. Hence, convex DPOI rewriting is amenable to efficient implementations.

We conclude this section by showing that, for certain well-behaved rewriting systems, convexity of matches follows automatically.

Definition 37. *A monogamous acyclic cospan* $n \xrightarrow{f} G \xleftarrow{g} m$ *is strongly connected if for every input* $x \in f(n)$ *and output* $y \in g(n)$ *there exists a path from* x *to* y. *A DPO rewriting system is left-connected if it is left-linear and, for every rule* $L \leftarrow i + j \to R$, *the induced cospans* $i \to L \leftarrow j$ *and* $i \to R \leftarrow j$ *are monogamous acyclic and* $i \to L \leftarrow j$ *is strongly connected. We call a PROP rewriting system* $\mathscr{R}$ *on* $\mathbf{S}_\Sigma$ *left-connected if for every* $\langle l, r \rangle \in \mathscr{R}$ *the associated DPO rule* $\langle\!\langle \ulcorner l \urcorner, \ulcorner r \urcorner \rangle\!\rangle$ *is left-connected.*

In Definition 37, strong connectedness prevents non-convex matches as in Example 32, whereas left-linearity guarantees uniqueness of the pushout complements, and prevents the problem in Example 29. We are then able to prove the following theorem, for the not necessarily convex DPOI rewriting relation $\rightsquigarrow$.

Theorem 38. *Let* $\mathscr{R}$ *be a left-connected rewriting system on* $\mathbf{S}_\Sigma$. *Then*
  (1) if $d \Rightarrow_{\mathscr{R}} e$ *then* $\langle\!\langle \ulcorner d \urcorner \rangle\!\rangle \rightsquigarrow_{\langle\!\langle \ulcorner \mathscr{R} \urcorner \rangle\!\rangle} \langle\!\langle \ulcorner e \urcorner \rangle\!\rangle$;
  (2) if $\langle\!\langle \ulcorner d \urcorner \rangle\!\rangle \rightsquigarrow_{\langle\!\langle \ulcorner \mathscr{R} \urcorner \rangle\!\rangle} \langle\!\langle \ulcorner e \urcorner \rangle\!\rangle$ *then* $d \Rightarrow_{\mathscr{R}} e$.

Proof. (1) follows from Theorem 28. For (2), suppose $\langle\!\langle \ulcorner d \urcorner \rangle\!\rangle \cong G \leftarrow n + m$ and the rewriting relation arose from applying a left-connected rule $L \leftarrow i + j \to R$ at match $p : L \to G$. By left-connectedness, there exists a path from every input of L to every output. Hence, this will also be the case for the sub-graph $p(L)$. If there was a directed path from an output of $p(L)$ to an input, this would induce a directed cycle. But since G is a monogamous acyclic hypergraph, this cannot be the case. Hence, $m(L)$ is a convex sub-graph of G.

Furthermore, since the rewriting rule is left-linear, $L \leftarrow i + j$ is mono. Hence, we can compute the (unique) pushout complement by removing the hyperedges and non-interface nodes of $p(L)$ from G. Since L and G are both monogamous acyclic hypergraphs, the resulting pushout complement will always be a boundary complement. Hence, the DPOI rewriting step must in fact be a convex DPOI step, so we can apply Theorem 35 to complete the proof. $\qquad\square$

As a consequence of this theorem, if a rewriting system is left-connected, we can forego the convexity check mentioned in Remark 36, so we can enumerate matches of a single rule with left-hand side L in G in time $O(|L_\star||G_\star|)$.

4.1 Characterisation for coloured PROPs

It is a routine exercise to generalise the results in this section to coloured props. First, fixed a set C of colours and a monoidal signature Σ on C, Bonchi et al. (2022) also states a multi-coloured version of Theorem 28, proving a correspondence between rewriting in $\mathbf{S}_{C,\Sigma}$ modulo the equations of $\mathbf{Frob}_C$, and DPOI rewriting in $\mathbf{Hyp}_{C,\Sigma}$. One may then define convex DPOI rewriting in $\mathbf{Hyp}_{C,\Sigma}$, in the same way as we did for $\mathbf{Hyp}_\Sigma$, and show correspondence results between this and rewriting in $\mathbf{S}_{C,\Sigma}$, analogous to Theorems 35 and 38: the colouring on nodes does not affect how these characterisations are formulated and proven.

Theorem 39. *Let* $\mathscr{R}$ *by any rewriting system on* $\mathbf{S}_{C,\Sigma,}$. *Then*

$$d \Rightarrow_{\mathscr{R}} e \text{ iff } \langle\!\langle \ulcorner d \urcorner \rangle\!\rangle \Rightarrow_{\langle\!\langle \ulcorner \mathscr{R} \urcorner \rangle\!\rangle} \langle\!\langle \ulcorner e \urcorner \rangle\!\rangle.$$

https://doi.org/10.1017/S0960129522000317 Published online by Cambridge University Press

Furthermore, let $\mathscr{R}$ be left-connected. Then
(1) *if $d \Rightarrow_{\mathscr{R}} e$ then $\langle\!\langle \ulcorner d \urcorner \rangle\!\rangle \rightsquigarrow_{\langle\!\langle \ulcorner \mathscr{R} \urcorner \rangle\!\rangle} \langle\!\langle \ulcorner e \urcorner \rangle\!\rangle$;*
(2) *if $\langle\!\langle \ulcorner d \urcorner \rangle\!\rangle \rightsquigarrow_{\langle\!\langle \ulcorner \mathscr{R} \urcorner \rangle\!\rangle} \langle\!\langle \ulcorner e \urcorner \rangle\!\rangle$ then $d \Rightarrow_{\mathscr{R}} e$.*

5. Case Studies

The two most fundamental properties of interest for a rewriting system are *termination* and *confluence*. A rewriting relation is terminating if it admits no infinite sequence of rewrites, and it is confluent if any pair of hypergraphs (or terms, etc.) arising from G by a sequence of rewriting steps can eventually be rewritten to the same hypergraph. Taken together, these properties imply the existence of unique normal forms.[2]

We will now apply the framework we have developed to two specific SMTs: Frobenius semi-algebras and bialgebras. For both of these structures, we construct the associated DPOI rewriting system and show that it is terminating. We will also show that the first theory is not confluent, by adapting a counter-example due to Power to the setting of convex rewriting. The second theory is confluent, but we leave the proof for the sequel paper, where we develop critical pair analysis for convex rewriting (Bonchi et al. 2022).

As with term rewriting theory, an important tool for termination proofs is that of *reduction orderings*. For a preorder $\preceq_a$ on hypergraphs, we can define the associated equivalence relation $\sim_a$ and the strict ordering $\prec_a$ as follows

$$G \sim_a H \iff (G \preceq_a H \wedge H \preceq_a G) \qquad G \prec_a H \iff (G \preceq_a H \wedge G \not\sim_a H)$$

Definition 40. *A preorder $\preceq_a$ is called a reduction ordering for a rewriting system $\mathscr{R}$ if it is well-founded (i.e. has no infinite decreasing chains with respect to $\prec_a$) and*

$$G \Rightarrow_{\mathscr{R}} H \implies H \prec_a G$$

Similarly, a preorder $\preceq_a$ is called a weak reduction ordering for $\mathscr{R}$ if it is well-founded and

$$G \Rightarrow_{\mathscr{R}} H \implies H \preceq_a G$$

Clearly the existence of a reduction ordering forbids infinite sequences of rewrites, hence any $\mathscr{R}$ that admits a reduction ordering is terminating.

A common strategy in termination proofs is to define reduction orderings in pieces which are then combined lexicographically. For pre-orders $\preceq_a$ and $\preceq_b$, we define the *lexicographic ordering* $\preceq_{a,b}$ as follows

$$H \preceq_{a,b} G \iff (H \prec_a G \vee (H \sim_a G \wedge H \preceq_b G))$$

The following lemma can be shown straightforwardly from the definitions above.

Lemma 41. *For a rewriting system $\mathscr{R} = \mathscr{R}_1 \cup \mathscr{R}_2$, if $\preceq_a$ is a reduction ordering for $\mathscr{R}_1$, $\preceq_a$ is a weak reduction ordering for $\mathscr{R}_2$, and $\preceq_b$ is a reduction ordering for $\mathscr{R}_2$, then $\preceq_{a,b}$ is a reduction ordering for $\mathscr{R}$.*

Proof. Using the definition of a lexicographic ordering, we can see that the associated strict ordering can be expressed as follows

$$H \prec_{a,b} G \iff (H \prec_a G \vee (H \sim_a G \wedge H \prec_b G))$$

From this, we see that $\preceq_{a,b}$ is well-founded whenever $\preceq_a$ and $\preceq_b$ are. Then, if $G \Rightarrow_{r_1} H$ for some $r_1 \in \mathscr{R}_1$, then $H \prec_a G$, so $H \prec_{a,b} G$. If $G \Rightarrow_{r_2} H$ for some $r_2 \in \mathscr{R}_2$, then $H \preceq_a G$ and $H \prec_b G$. From $H \preceq_a G$, it is either the case that $H \prec_a G$ or $H \sim_a G$, which in either case yields $H \prec_{a,b} G$. $\square$

With this bit of rewriting theory in hand, we are ready to look at our two case studies.

5.1 Frobenius semi-algebras

Frobenius semi-algebras are Frobenius algebras lacking the unit and counit equations. That is, they are the free PROP generated by the signature

$$\left\{ \mu := \;\raisebox{-3pt}{\includegraphics[height=12pt]{mu}}\; , \delta := \;\raisebox{-3pt}{\includegraphics[height=12pt]{delta}}\; \right\}$$

modulo the following equations

(10)

It is interesting to study such structures, because full (co)unital Frobenius algebras always induce a compact closed structure. Hence, categories that are not compact closed, such as infinite-dimensional vector spaces, do not in general have Frobenius algebras. They can nevertheless have Frobenius semi-algebras, which form the basis of interesting algebraic structures relevant to quantum theory, such as H⋆-algebras (Abramsky and Heunen 2012).

Since Frobenius semi-algebras lack many of the equations of a Frobenius algebra, we cannot use the technique for rewriting modulo Frobenius developed in Bonchi et al. (2022). Nevertheless, we can represent this theory using a hypergraph rewriting system **FS**, defined as follows
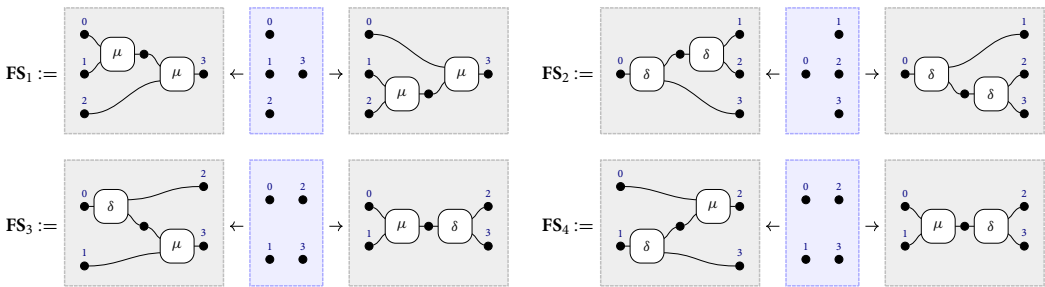
We first give a proof of termination for this rewriting system. This would be quite involved if we wished to prove it using syntactic rewriting, modulo the equations of an SMC, but here we show it is relatively straightforward, using some graph-theoretic reduction orderings.

We first deal with (co)associativity. It should be the case that naïvely applying rules **FS**$_1$ and **FS**$_2$ will eventually terminate with all trees of multiplications and comultiplications associated to the right (or bottom, as we are reading diagrams left-to-right). More formally, for any vertex x, let a μ-tree with root x be a maximal tree of μ-hyperedges with output x. Similarly, a δ-tree with root x is a maximal tree of δ-hyperedges with input x.

For a μ-hyperedge h, let the $\mathscr{L}$-weight $\ell(h)$ be the size of the μ-tree whose root is the first input of h. Similarly, for a δ-hyperedge, let $\ell(h)$ be the size of the δ-tree whose root is the first output of h. Let $\ell(h) = 0$ otherwise and

$$\mathscr{L}(G) := \sum_{h \in G_{2,1} \cup G_{1,2}} \ell(h)$$

Lemma 42. *The following is a reduction ordering for* {**FS**$_1$, **FS**$_2$}

$$H \preceq_{\mathscr{L}} G \iff \mathscr{L}(H) \leq \mathscr{L}(G)$$

Proof. $\mathscr{L}$ is $\mathbb{N}$-valued, so $\preceq_{\mathscr{L}}$ is well-founded.

Applying the rule $\mathbf{FS}_1$ has no effect on the $\mathscr{L}$-weight of any μ-hyperedges outside of the image of the left-hand side. Suppose that there are μ-trees of size a, b, c connected to inputs 0, 1, 2 of the left-hand side, respectively. The $\mathscr{L}$-weight of the two μ-hyperedges on the left-hand side are thus a and $a + b + 1$, whereas on the right-hand side they are a and b. Hence, $\preceq_{\mathscr{L}}$ is strictly decreased by $\mathbf{FS}_1$. The property for $\mathbf{FS}_2$ follows symmetrically. $\square$

The previous result accounts for associativity of μ and of δ, but we should do the same with the two Frobenius equations. We can use the fact that each of the Frobenius rewriting rules $\mathbf{FS}_3$ and $\mathbf{FS}_4$ strictly decreases $|\mathscr{D}(G)|$ where

$$\mathscr{D}(G) := \{(h \in G_{2,1}, h' \in G_{1,2}) \mid \text{there is no path from } h \text{ to } h'\}$$

Following Definition 19, we will use the term *path* in this and the next section to refer exclusively to directed paths, i.e. sequences of hyperedges $[h_1, \ldots, h_n]$ such that h_{i+1} is a successor of h_i. Note that, since $h \in G_{2,1}$, it must be a μ-hyperedge, and since $h' \in G_{1,2}$, it must be δ-hyperedge. Also note the negation in the definition of $\mathscr{D}$: as more paths are introduced, the set $\mathscr{D}(G)$ gets smaller.
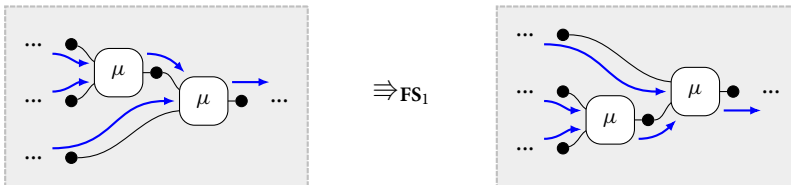
Lemma 43. *The following is a weak reduction ordering for* $\{\mathbf{FS}_1, \mathbf{FS}_2\}$ *and a reduction ordering for* $\{\mathbf{FS}_3, \mathbf{FS}_4\}$

$$H \preceq_{\mathscr{D}} G \iff |\mathscr{D}(H)| \leq |\mathscr{D}(G)|$$

Proof. $\mathscr{D}$ sends a hypergraph to a finite set of ordered pairs, so $\preceq_{\mathscr{D}}$ is well-founded.

Note that all four of the rules $\mathbf{FS}_i$ preserve the number of μ- and δ-hyperedges in G. Hence, if $G \Rightarrow_{\mathbf{FS}_i} H$, we can take H to have the same set of hyperedges as G, but with different connectivity. For the remainder of the proof, we examine how each rule application affects the paths in a hypergraph. For this, we rely on the fact that all of the hypergraphs involved in the rewriting are monogamous and acyclic. As a consequence of monogamy, any path entering the left-hand side of a rule must do so via an input, and any path exiting must do so via an output.

For $G \Rightarrow_{\mathbf{FS}_1} H$, a μ-hyperedge h has a path to a δ-hyperedge in G if and only if it does in H. This follows from the fact that there is a path from every input to the output and from both μ-hyperedges to the output on both sides of the rewriting rule

Hence $G \sim_{\mathscr{D}} H$. A symmetric argument holds for $\mathbf{FS}_2$, so we conclude that $\preceq_{\mathscr{D}}$ gives a weak reduction ordering for $\{\mathbf{FS}_1, \mathbf{FS}_2\}$.

For $G \Rightarrow_{\mathbf{FS}_3} H$, let L be the image of the left-hand side of $\mathbf{FS}_3$ in G and R the image of the right-hand side of $\mathbf{FS}_3$ in H. We will refer to the unique μ-hyperedge in L and R as h, and the unique δ-hyperedge in L and R as h'. First, note that there is a path from every input in R to every output. There is also a path from every input of R to h' and from h to every output
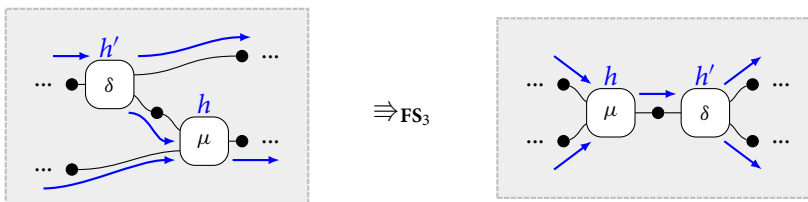
Hence, applying $\mathbf{FS}_3$ can only create more paths from μ-hyperedges to δ-hyperedges and never breaks them, so $\mathscr{D}(H) \subseteq \mathscr{D}(G)$. Furthermore, by acyclicity, there must not be a path from h to h' in G, but there is one in H. So the containment $\mathscr{D}(H) \subseteq \mathscr{D}(G)$ is strict and thus $H \prec_{\mathscr{D}} G$. The argument for $\mathbf{FS}_4$ is identical. $\qquad\square$

Theorem 44. *FS is terminating.*

Proof. We form the lexicographic ordering $\preceq_{\mathbf{FS}} := \preceq_{\mathscr{D}, \mathscr{L}}$. It then follows from Lemmas 41, 42, and 43 that $\preceq_{\mathbf{FS}}$ gives a reduction ordering for $\mathbf{FS}$. $\qquad\square$
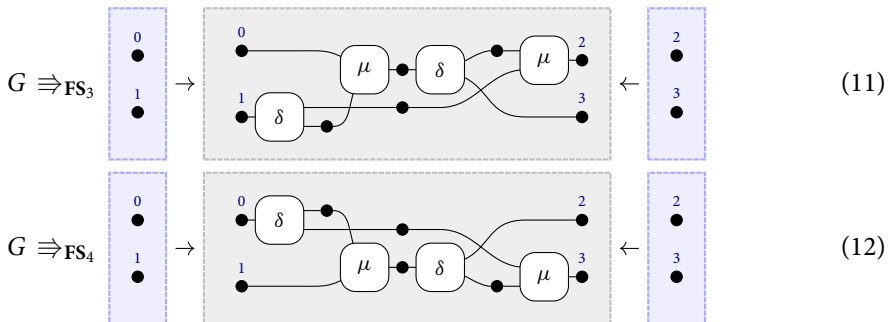
It is worth noting that acylicity plays a crucial role in the above proof. If the two hyperedges in the left-hand side of $\mathbf{FS}_3$ or $\mathbf{FS}_4$ were part of a directed cycle, one could potentially find an infinite sequence of rule applications.

Next example shows that, while *convex* rewriting prevents us from introducing cycles (and hence non-terminating behaviour), it also breaks confluence from this system. Our counter-example is based on Example 3.11 from Power (1991), which was given in terms of string diagrams by Hadzihasanovic (2020*a*). While Power's original example concerned morphisms in a 3-category, the same phenomenon appears in symmetric monoidal categories and can be understood as a surprising consequence of the convexity condition: namely, even non-overlapping rule applications can block one another.

Example 45. Consider the following diagram, and its rendering as a cospan of hypergraphs
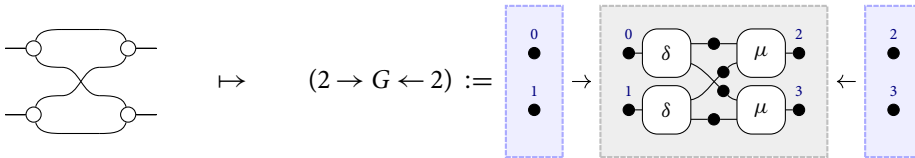
For a bialgebra (cf. the next section), this is a familiar diagram, as it is the right-hand side of one of the equations. In that context, it is also a normal form. That is, none of the bialgebra rules can be applied, so this diagram is considered fully simplified. However, for Frobenius semi-algebras there are two different rules that apply: $\mathbf{FS}_3$ and $\mathbf{FS}_4$. Let us have a look at the hypergraph we obtain when we apply each of these two rules

$$G \Rrightarrow_{\mathbf{FS}_3} \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (11)$$

$$G \Rrightarrow_{\mathbf{FS}_4} \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (12)$$

Note that these two rule applications act on disjoint sets of hyperedges, and yet they still interfere with each other. In particular, applying $\mathbf{FS}_3$ introduces a new path from the leftmost δ-hyperedge in hypergraph (11) to the rightmost μ-hyperedge. Whereas these two hyperedges previously defined a convex sub-hypergraph of G, they are no longer convex after $\mathbf{FS}_3$ has been applied. Consequently, these no longer define a valid match for $\mathbf{FS}_4$. Similarly, applying $\mathbf{FS}_4$ to G in (12) blocks the application of $\mathbf{FS}_3$. In fact, neither of the hypergraphs (11) nor (12) contain a match for
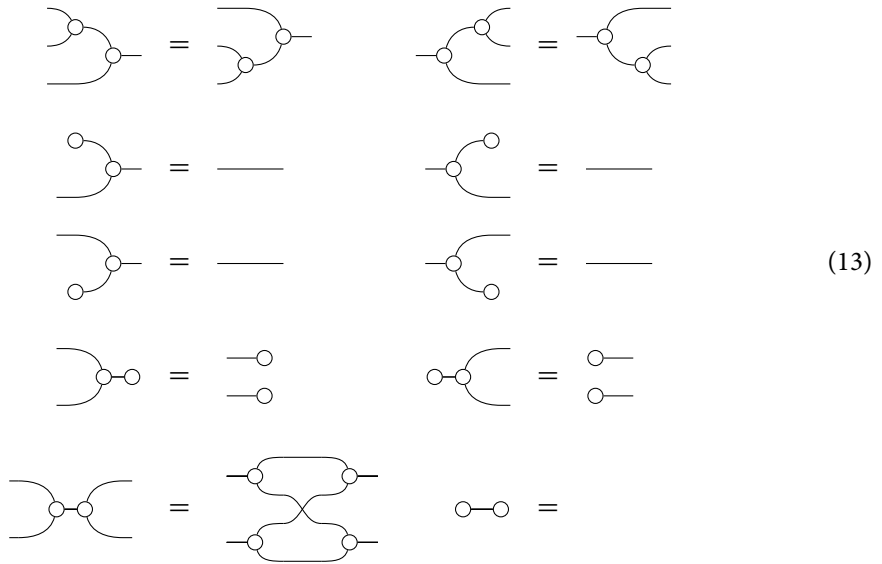
Figure 2. The equations of a bialgebra.

any of the rules of **FS**. Hence, we have distinct hypergraphs H_1 and H_2 arising from G that cannot be rewritten into the same hypergraph by the rules of **FS**, so **FS** is not confluent.

5.2 Bialgebras

We now consider *bialgebras*, i.e. a theory with the same generators as a Frobenius algebra

$$\left\{ \mu := \quad , \eta := \quad , \delta := \quad , \varepsilon := \quad \right\}$$

but with a different set of equations. It is the theory underlying the (bi)-category of spans of sets with disjoint union as monoidal product (Bruni and Gadducci 2001), and it has been used in the axiomatisation of flownomials, an algebraic presentation of flowcharts (Stefanescu 2000). The equations of non-commutative bialgebras are given in Figure 2, and their associated DPO rewriting rules, forming the rewriting system **BA**, are shown in Figure 3.

   We now focus on proving termination for this system. Its proof is slightly more elaborate, as there are more rules, and the rules do not always preserve the number of hyperedges in a hypergraph. Notably, repeated applications of $\mathbf{BA}_9$ can significantly increase the number of hyperedges.

   Nevertheless, we can find useful reduction orderings by counting paths rather than counting hyperedges. For this, we define two kinds of paths, one that tracks paths involving (co)Units and the other for (co)Multiplications

-   a *U-path* is a path p from an input or an η-hyperedge to an output or an ε-hyperedge;
-   an *M-path* is a path from a μ-hyperedge to a δ-hyperedge.

Next, define orders $\preceq_U$, $\preceq_M$, $\preceq_\mu$, $\preceq_\delta$ based on counting the number of U-paths, M-paths, μ-hyperedges, and δ-hyperedges, respectively. Using these four orderings, along with $\preceq_{\mathscr{L}}$ defined in the previous section, we define the following lexicographic ordering

$$\preceq_{\mathbf{BA}} := \preceq_{U,M,\mu,\delta,\mathscr{L}} \tag{14}$$

Each of the components of $\preceq_{\mathbf{BA}}$ is well-founded, so $\preceq_{\mathbf{BA}}$ itself is well-founded. Thus, we can conclude as follows.
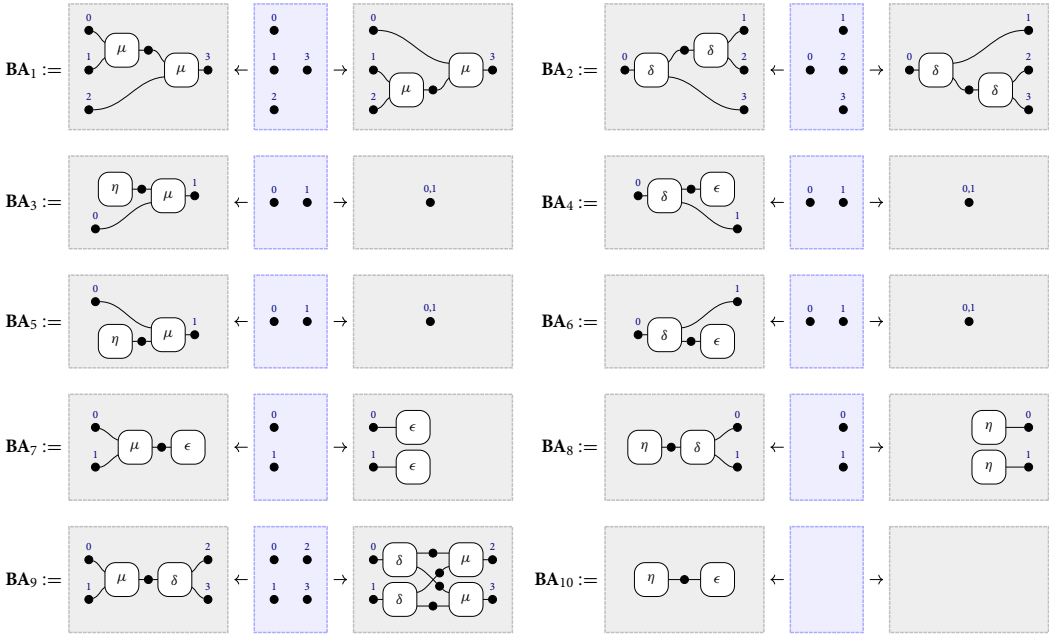
Figure 3. DPO rewriting system **BA** for bialgebras.

Theorem 46. $\preceq_{BA}$ *is a reduction ordering for* **BA**, *thus* **BA** *terminates.*

Proof. We argue rule-by-rule, showing that each one is strictly decreasing in one of the orders from (14), and non-increasing in every order that is prior in the lexicographic ordering.

Since every rule $\mathbf{BA}_j$ has a unique path from every input to every output for both left- and right-hand side, applications of these rules have no effect on paths which start and finish outside of their image. Hence, for each rule, we only need to consider paths which start or terminate in the image of the left-hand side.

$\mathbf{BA}_1$ has no effect on η or ε hyperedges, hence on $\preceq_U$. No M-path can terminate in $\mathbf{BA}_1$ and any M-path originating on $\mathbf{BA}_1$ must exit through the unique output. Since there are precisely two μ-hyperedges in both the left- and the right-hand side, there is a one-to-one correspondence between M-paths before and after applying the rule. $\mathbf{BA}_1$ leaves the number of μ and δ hyperedges fixed, so it suffices to show it strictly decreases $\preceq_{\mathscr{L}}$. Applying the rule has no effect on the $\mathscr{L}$-weight of any μ-hyperedges outside of the image of the left-hand side. Suppose there are μ-trees of size a, b, c connected to inputs $0, 1, 2$ of the left-hand side, respectively. The $\mathscr{L}$-weight of the two μ-hyperedges on the left-hand side is thus a and $a + b + 1$, whereas on the right-hand side they are a and b. Hence, $\preceq_{\mathscr{L}}$ is strictly decreased. $\mathbf{BA}_2$ follows via a symmetric argument.

Since $\mathbf{BA}_3$–$\mathbf{BA}_6$ and $\mathbf{BA}_{10}$ remove η- and ε-hyperedges from the hypergraph, they will strictly decrease the number of U-paths.

For $\mathbf{BA}_7$, no U-path can terminate in the left-hand side, and any U-path starting in the left-hand side must exit through one of the two outputs. Hence, it corresponds to a unique U-path exiting the right-hand side. M-paths are unaffected, as is the number of δ-hyperedges. However, the number of μ-hyperedges is strictly decreased, so $\mathbf{BA}_7$ strictly decreases $\preceq_{\mu}$. The argument for $\mathbf{BA}_8$ is symmetric, yet with respect to $\preceq_{\delta}$.

$\mathbf{BA}_9$ has no η or ε-hyperedges in either the left- or the right-hand side, so it leaves the number of U-paths fixed. Consider an M-path that enters the left-hand side from the left. It enters either from input 0 or input 1, hence it corresponds to a unique M-path entering the right-hand side.

We can argue similarly for M-paths exiting on the right. Hence, the only M-path left to consider is the one from the μ-hyperedge to the δ-hyperedge in the left-hand side, which is eliminated. Thus, **BA**$_9$ strictly reduces $\preceq_M$. □

It is also possible to show that, unlike **FS**, the rewriting system **BA** is confluent. An important factor in the confluence proof is the fact that the rewriting system **BA** is left-connected (cf. Definition 37), so we do not need to impose convexity as an additional requirement when we do rewriting, thanks to Theorem 38. This rules out situations like the one for **FS** in Example 45, where disjoint rule applications can block one another due to convexity considerations.

In order to show confluence, we can use a technique known as *critical pair analysis*. There are various subtleties arising in critical pair analysis for DPO rewriting (Plump 1993) and general rewriting for symmetric monoidal categories (Lafont 2003), which are beyond the scope of the this paper. Hence, we leave a formal proof of the confluence of **BA** for the sequel to this paper, in which we develop a comprehensive framework for critical pair analysis on convex DPOI rewriting (Bonchi et al. 2022).

6. Conclusions and Further Works

In this paper, we developed a practical approach to the rewriting of symmetric monoidal categories. Relying on a previously identified Bonchi et al. (2022) correspondence between string diagrams and cospans of hypergraphs, we classify those cospans that do not rely on the presence of an additional Frobenius structure, i.e. those that are relevant when considering only symmetric monoidal categories. Having thereby identified a combinatorial structure that serves as a sound encoding of string diagrams, we use the mechanism of DPO rewriting, which we modify in order to ensure soundness and completeness. This involves the identification of sufficiently well-behaved pushout complements and the restriction to similarly well-behaved matches: roughly speaking, these restrictions ensure that the rewrites themselves can only rely on the "vanilla" symmetric monoidal structure, without any use of the laws of Frobenius algebras. We arrive at a practical procedure for rewriting modulo symmetric monoidal laws: assuming an implementation of DPO rewriting of hypergraphs, each restriction can be easily checked algorithmically.

While originating in category theory (Joyal and Street 1991), string diagrams have been influential in computer science, especially after the paper on traced monoidal categories by Joyal, Street and Verity (Joyal et al. 1996). However, the correspondence between terms of "2-dimensional" algebraic structures – i.e. those with sequential and parallel composition, understood as arrows of a free symmetric monoidal category – and suitable hypergraphs (flow diagrams) were recognised earlier and studied at least since the work of Stefanescu (see the references in Selinger 2011).

Closely related to our work, Dixon and Kissinger (2013) use cospans of string graphs (called there open graphs) to encode morphisms in a symmetric monoidal category and reason equationally via DPO rewriting. There is an evident encoding of the hypergraphs we use into string graphs. However, the notion of rewriting considered there is only sound if there is a trace on the symmetric monoidal category, whereas our notion of convex DPO rewriting guarantees soundness for any symmetric monoidal category. Another difference is that we directly work in an adhesive category, while the category of open graphs inherits the relevant properties from an embedding into the adhesive category of typed graphs.

In logic and computer science, diagrammatic rewriting was motivated in part by computational patterns appearing in the proof theory of linear logic, leading to general diagrammatic rewriting frameworks such as interaction nets (Lafont 1990; Mazza 2006). A general rewriting theory of such structures has also been developed; notably, Burroni (1993) generalised term rewriting to higher dimensions, including the 3-dimensional case of string diagram rewriting; see Mimram's survey (Mimram 2014). Here, in order to capture symmetric monoidal structure, the laws of symmetric monoidal categories would usually be considered as explicit rewriting rules, resulting in

sophisticated rewriting systems whose analysis is often challenging (see e.g. Guiraud 2006; Lafont 2003). Abstract higher dimensional rewriting is far more general than our approach, which has been tailored over symmetric monoidal categories. The loss of generality brings the benefits of specialisation: our approach has the laws of symmetric monoidal categories built-in, reducing the complexity of the resulting rewriting systems. Thus, our work can be seen as part of a more general effort the search for characterisations that capture some parts of relevant algebraic structure in combinatorial models that bring the possibility of implementation; see e.g. Obradovic's work on capturing the algebra of cyclic operads (Curien and Obradovic 2020; Obradovic 2017) and Hadzihasanovic's recent work on diagrammatic sets (Hadzihasanovic 2020*b*, 2021).

This is the second of a three paper series, the first being Bonchi et al. (2022), and the third Bonchi et al. (2022) devoted to solving the problem of confluence for string diagram rewriting. With these papers, we hope to lay the foundations for the next generation of diagrammatic proof assistants for SMTs. Such tools would lie between Globular (Bar et al. 2018) and `homotopy.io` (https://homotopy.io/) on the one hand, the foundations of which are designed for reasoning about higher dimensional weak structures and thus have minimal algebraic structure built-in, and Quantomatic (Kissinger and Zamdzhiev 2015), PyZX and QuiZX on the other, in which the implementations of rewriting rely on the rich algebraic structure of the ZX-calculus.

Notes

1 Pushouts are unique only up-to iso, which explains why arrows of $\mathsf{Csp}(\mathbf{Hyp}_\Sigma)$ are defined up-to the same equivalence.
2 For background on termination and confluence in term rewriting systems, see for instance Terese (2003). Termination for string diagram rewriting has been studied as an instance of higher-dimensional term rewriting, see Guiraud (2006) and the discussion in Section 6.

References

Abramsky, S. and Heunen, C. (2012). H*-algebras and nonunital Frobenius algebras: first steps in infinite-dimensional categorical quantum mechanics. In: Abramsky, S. and Mislove, M. (eds.) *Mathematical Foundations of Information Flow*, Proceedings of Symposia in Applied Mathematics, vol. 71, American Mathematical Society, 1–14.

Baez, J. and Erbele, J. (2015). Categories in control. *Theory and Application of Categories* **30** 836–881.

Bar, K., Kissinger, A. and Vicary, J. (2018). Globular: an online proof assistant for higher-dimensional rewriting. *Logical Methods in Computer Science* **14** (1).

Behrisch, M., Kerkhoff, S. and Power, J. (2012). Category theoretic understandings of universal algebra and its dual: monads and Lawvere theories, comonads and what? In: Berger, U. and Mislove, M. W. (eds.) *MFPS 2012*, ENTCS, vol. 286, Elsevier, 5–16.

Bonchi, F., Gadducci, F., Kissinger, A., Sobociński, P. and Zanasi, F. (2016). Rewriting modulo symmetric monoidal structure. In: Grohe, M., Koskinen, E. and Shankar, N. (eds.) *LICS 2016*, ACM, 710–719.

Bonchi, F., Gadducci, F., Kissinger, A., Sobociński, P. and Zanasi, F. (2017a). Confluence of graph rewriting with interfaces. In: Yang, H. (ed.) *ESOP 2017*, LNCS, vol. 10201, Springer, 141–169.

Bonchi, F., Holland, J., Pavlovic, D. and Sobociński, P. (2017b). Refinement for signal flow graphs. In: Meyer, R. and Nestmann, U. (eds.) *CONCUR 2017*, LIPIcs, vol. 85, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 24:1–24:16.

Bonchi, F., Pavlovic, D. and Sobociński, P. (2017c). Functorial semantics for relational theories. Preprint available at arXiv:1711.08699.

Bonchi, F., Sobociński, P. and Zanasi, F. (2017d). The calculus of signal flow diagrams I: linear relations on streams. *Information and Computation* **252** 2–29.

Bonchi, F., Gadducci, F., Kissinger, A., Sobociński, P. and Zanasi, F. (2018a). Rewriting with Frobenius. In: Dawar, A. and Grädel, E. (eds.) *LICS 2018*, ACM, 165–174.

Bonchi, F., Gadducci, F., Kissinger, A., Sobociński, P. and Zanasi, F. (2022). String diagram rewrite theory I: rewriting with Frobenius structure. *Journal of the ACM* **69** (2) 14:1–14:58.

Bonchi, F., Gadducci, F., Kissinger, A., Sobociński, P. and Zanasi, F. (2022). String diagram rewrite theory III: confluence with and without Frobenius. *Mathematical Structures in Computer Science*.

Bonchi, F., Gadducci, F. and König, B. (2009). Synthesising CCS bisimulation using graph rewriting. *Information and Computation* **207** (1) 14–40.

Bonchi, F., Piedeleu, R., Sobociński, P. and Zanasi, F. (2019). Graphical affine algebra. In: *LICS 2019*, IEEE, 1–12.

Bonchi, F., Sobociński, P. and Zanasi, F. (2018b). Deconstructing Lawvere with distributive laws. *Logic and Algebraic Methods in Programming* **95** 128–146.

Bruni, R. and Gadducci, F. (2001). Some algebraic laws for spans (and their connections with multirelations). In: Kahl, W., Parnas, D. L. and Schmidt, G. (eds.) *RELMIS 2001*, ENTCS, vol. 44(3), Elsevier, 175–193.

Burroni, A. (1993). Higher dimensional word problems with applications to equational logic. *Theoretical Computer Science* **115** (1) 43–62.

Coecke, B. and Duncan, R. (2011). Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics* **13** (4) 1–85.

Coecke, B. and Kissinger, A. (2017). *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*, Cambridge University Press.

Corradini, A. and Gadducci, F. (2002). Functorial semantics for multi-algebras and partial algebras, with applications to syntax. *Theoretical Computer Science* **286** (2) 293–322.

Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R. and Löwe, M. (1997). Algebraic approaches to graph transformation - Part I: basic concepts and double pushout approach. In: Rozenberg, G. (ed.) *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*, World Scientific, 163–246.

Curien, P.-L. and Obradovic, J. (2020). Categorified cyclic operads. *Applied Categorical Structures* **28** (1) 59–112.

Dixon, L. and Kissinger, A. (2013). Open-graphs and monoidal theories. *Mathematical Structures in Computer Science* **23** (2) 308–359.

Ehrig, H. and König, B. (2004). Deriving bisimulation congruences in the DPO approach to graph rewriting. In: Walukiewicz, I. (ed.) *FOSSACS 2004*, LNCS, vol. 2987, Springer, 151–166.

Fox, T. (1976). Coalgebras and cartesian categories. *Communications in Algebra* **4** 665–667.

Gadducci, F. (2007). Graph rewriting for the π-calculus. *Mathematical Structures in Computer Science* **17** (3) 407–437.

Gadducci, F. and Heckel, R. (1998). An inductive view of graph transformation. In: Parisi-Presicce, F. (ed.) *WADT 1997*, vol. 1376, Springer, 223–237.

Garner, R. and Power, J. (2018). An enriched view on the extended finitary monad-Lawvere theory correspondence. *Logical Methods in Computer Science* **14** (1).

Ghica, D. R., Jung, A. and Lopez, A. (2017). Diagrammatic semantics for digital circuits. In: Goranko, V. and Dam, M. (eds.) *CSL 2017*, LIPIcs, vol. 82, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 24:1–24:16.

Guiraud, Y. (2006). Termination orders for three-dimensional rewriting. *Pure and Applied Algebra* **207** (2) 341–371.

Hadzihasanovic, A. (2020a). Via Twitter: twitter.com/amar_hh/status/1336274654923788288?s=20.

Hadzihasanovic, A. (2020b). A combinatorial-topological shape category for polygraphs. *Applied Categorical Structures* **28** (3) 419–476.

Hadzihasanovic, A. (2021). The smash product of monoidal theories. In: *LICS 2021*, IEEE, 1–13.

Hyland, M., Plotkin, G. D. and Power, J. (2002). Combining computational effects: commutativity & sum. In: Baeza-Yates, R. A., Montanari, U. and Santoro, N. (eds.) *TCS 2002*, IFIP Conference Proceedings, vol. 223, Kluwer, 474–484.

Hyland, M. and Power, J. (2007). The category theoretic understanding of universal algebra: Lawvere theories and monads. In: Cardelli, L., Fiore, M. P. and Winskel, G. (eds.) *Computation, Meaning, and Logic*, ENTCS, vol. 172, Elsevier, 437–458.

Jacobs, B., Kissinger, A. and Zanasi, F. (2019). Causal inference by string diagram surgery. In: Bojanczyk, M. and Simpson, A. (eds.) *FOSSACS 2019*, LNCS, vol. 11425, Springer, 313–329.

Joyal, A. and Street, R. (1991). The geometry of tensor calculus, I. *Advances in Mathematics* **88** (1) 55–112.

Joyal, A., Street, R. and Verity, D. (1996). Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society* **119** (3) 447–468.

Kissinger, A. and Zamdzhiev, V. (2015). Quantomatic: a proof assistant for diagrammatic reasoning. Preprint available at arXiv:1503.01034.

Lack, S. and Power, J. (2009). Gabriel-Ulmer duality and Lawvere theories enriched over a general base. *Functional Programming* **19** (3–4) 265–286.

Lack, S. (2004). Composing PROPs. *Theory and Application of Categories* **13** (9) 147–163.

Lack, S. and Sobociński, P. (2005). Adhesive and quasiadhesive categories. *Theoretical Informatics and Applications* **39** (3) 511–546.

Lafont, Y. (1990). Interaction nets. In: Allen, F. E. (ed.) *POPL 1990*, 95–108.

Lafont, Y. (2003). Towards an algebraic theory of Boolean circuits. *Pure and Applied Algebra* **184** (2–3) 257–310.

Liberti, I. D., Loregian, F., Nester, C. and Sobociński, P. (2021). Functorial semantics for partial theories. In: *POPL 2021*.

Mac Lane, S. (1965). Categorical algebra. *Bulletin of the American Mathematical Society* **71** (1) 40–106.

Mazza, D. (2006). *Interaction Nets: Semantics and Concurrent Extensions*. Phd thesis, Université Aix-Marseille II/Università degli Studi Roma Tre.

Mimram, S. (2014). Towards 3-dimensional rewriting theory. *Logical Methods in Computer Science* **10** (2).

Moggi, E. (1991). Notions of computation and monads. *Information and Computation* **93** (1) 55–92.

Obradovic, J. (2017). *Cyclic Operads: Syntactic, Algebraic and Categorified Aspects*. Phd thesis, Université Paris 7.

Plotkin, G. D. and Power, J. (2001a). Adequacy for algebraic effects. In: Honsell, F. and Miculan, M. (eds.) *FOSSACS 2001*, LNCS, vol. 2030, Springer, 1–24.

Plotkin, G. D. and Power, J. (2001b). Semantics for algebraic operations. In: Brookes, S. D. and Mislove, M. W. (eds.) *MFPS 2001*, ENTCS, vol. 45, Elsevier, 332–345.

Plotkin, G. D. and Power, J. (2002). Notions of computation determine monads. In: Nielsen, M. and Engberg, U. (eds.) *FOSSACS 2002*, LNCS, vol. 2303, Springer, 342–356.

Plotkin, G. D. and Power, J. (2003). Algebraic operations and generic effects. *Applied Categorical Structures* **11** (1) 69–94.

Plotkin, G. D. and Power, J. (2004). Computational effects and operations: an overview. In: Escardó, M. H. and Jung, A. (eds.) *Workshop on Domains VI*, ENTCS, vol. 73, Elsevier, 149–163.

Plump, D. (1993). Hypergraph rewriting: critical pairs and undecidability of confluence. In: Ronan Sleep, M., Plasmeijer, M. J. and van Eekele, M. C. J. D. (eds.) *Term Graph Rewriting: Theory and Practice*, Wiley, 201–213.

Power, J. (1991). An n-categorical pasting theorem. In: Carboni, A., Pedicchio, M. C. and Rosolini, G. (eds.) *Category theory*, Lecture Notes in Mathematics, Springer, 326–358.

Power, J. (1999). Enriched Lawvere theories. *Theory and Applications of Categories* **6** (7) 83–93.

Power, J. (2004). Canonical models for computational effects. In: Walukiewicz, I. (ed.) *FOSSACS 2004*, LNCS, vol. 2987, Springer, 438–452.

Power, J. (2005). Discrete Lawvere theories. In: Fiadeiro, J. L., Harman, N., Roggenbach, M. and Rutten, J. J. M. M. (eds.) *CALCO 2005*, LNCS, vol. 3629, Springer, 348–363.

Power, J. (2006a). Countable Lawvere theories and computational effects. In: Seda, A. K., Hurley, T., Schellekens, M. P., Airchinnigh, M. M. and Strong, G. (eds.) *MFCSIT 2004*, ENTCS, vol. 161, Elsevier, 59–71.

Power, J. (2006b). The universal algebra of computational effects: Lawvere theories and monads. In: McBride, C. and Uustalu, T. (eds.) *MSFP@MPC 2006*, Workshops in Computing. BCS.

Sadrzadeh, M., Clark, S. and Coecke, B. (2013). The Frobenius anatomy of word meanings I: subject and object relative pronouns. *Logic and Computation* **23** (6) 1293–1317.

Sassone, V. and Sobociński, P. (2005). Reactive systems over cospans. In: *LICS 2005*, IEEE Computer Society, 311–320.

Selinger, P. (2011). A survey of graphical languages for monoidal categories. *Springer Lecture Notes in Physics* **13** (813) 289–355.

Stefanescu, G. (2000). *Network Algebra*, Springer, London.

Terese. (2003). *Term Rewriting Systems*, Cambridge University Press.

Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *Journal of the ACM* **23** (1) 31–42.

Zanasi, F. (2015). *Interacting Hopf Algebras: The Theory of Linear Systems*. Phd thesis, École Normale Supérieure de Lyon.

Zanasi, F. (2016). The algebra of partial equivalence relations. In: Birkedal, L. (ed.) *MFPS 2016*, ENTCS, vol. 325, Elsevier, 313–333.