

Rewriting with Frobenius

Filippo Bonchi
University of Pisa

Fabio Gadducci
University of Pisa

Aleks Kissinger
Radboud University

Pawel Sobocinski
University of Southampton

Fabio Zanasi
University College London

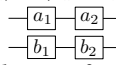
Abstract

Symmetric monoidal categories have become ubiquitous as a formal environment for the analysis of compound systems in a compositional, resource-sensitive manner using the graphical syntax of string diagrams. Recently, reasoning with string diagrams has been implemented concretely via double-pushout (DPO) hypergraph rewriting. The hypergraph representation has the twin advantages of being convenient for mechanisation and of completely absorbing the structural laws of symmetric monoidal categories, leaving just the domain-specific equations explicit in the rewriting system.

In many applications across different disciplines (linguistics, concurrency, quantum computation, control theory, ...) the structural component appears to be richer than just the symmetric monoidal structure, as it includes one or more Frobenius algebras. In this work we develop a DPO rewriting formalism which is able to absorb multiple Frobenius structures, thus sensibly simplifying diagrammatic reasoning in the aforementioned applications. As a proof of concept, we use our formalism to describe an algorithm which computes the reduced form of a diagram of the theory of interacting bialgebras using a simple rewrite strategy.

Keywords PROP, symmetric monoidal category, Frobenius algebra, DPO graph rewriting

1 Introduction

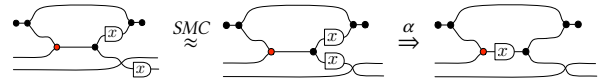
String diagrams Symmetric monoidal categories (SMCs) are an increasingly popular mathematical framework for reasoning about generic compositional systems. An SMC is a category that has sequential composition operation of morphisms ($c ; d$), a parallel composition ($c \oplus d$), and a collection of symmetry morphisms, giving the ability to ‘swap’ the components of a parallel composition [28]. It is convenient to use the two-dimensional notation of *string diagrams* to express morphisms in SMCs, where they are depicted as boxes, sequential composition as plugging boxes together, parallel composition as juxtaposition, and symmetries as wire-crossings: . This has the advantage of absorbing structural equalities prescribed by the definition of SMC. For instance, the two sides of the exchange law $(a_1 ; a_2) \oplus (b_1 ; b_2) = (a_1 \oplus b_1) ; (a_2 \oplus b_2)$ are encoded by the same string diagram . The graphical syntax emphasises *connectivity* and *sharing of resources* between

components, which makes it particularly applicable in the analysis of computational models with inter-connections between processes such as dynamical systems [16, 29], signal-flow diagrams [1, 7], digital circuits [19], and models of quantum computation [13].

Diagrammatic reasoning As in the case of universal algebra, it is often very useful to consider SMCs that are freely generated from a signature Σ —which provides the “building blocks” for the string diagrammatic syntax—and a set of equations E . Proving two morphisms in such a category are equal is done by *diagrammatic reasoning*, that is, by performing string diagram transformations using the rules of E and the SMC laws. Notably, there is a key conceptual distinction between the SMC laws (e.g. the exchange law above) and the equations of E : while the former are a *structural* part of any theory, the latter are *domain-specific*, depending on the class of systems under consideration. This distinction is reflected in the way that diagrammatic reasoning is both practised and mechanised (for instance in the graphical proof assistant Quantomatic [22]). While the laws of SMCs are treated as structural congruences on terms, allowing their representation as string diagrams, the domain-specific equations in E are, instead, usually understood as *rewriting rules*. For a concrete example, borrowed from the calculus of signal flow diagrams [7], consider the following diagram rewriting rule, which reduces the number of registers in a signal flow circuit

$$\alpha : \text{---} \begin{array}{c} \boxed{x} \\ \text{---} \bullet \\ \boxed{x} \end{array} \text{---} \Rightarrow \text{---} \boxed{x} \text{---} \bullet \text{---}$$

The diagram below on the left has a redex for α : notice that in order for the redex to appear as a sub-diagram, *à la* term rewriting, one must first deform it using the laws of SMCs



This example highlights the main challenge for implementing diagrammatic reasoning: matching is not “on-the-nose”, but modulo the laws of SMCs. In recent work [2, 3, 32] the authors developed a framework for implementing this style of rewriting. The key step is to interpret diagrams combinatorially as cospans of hypergraphs. This provides an off-the-shelf representation of string diagram rewriting in terms of the well-established theory of *double-pushout* (DPO) hypergraph rewriting, which is both machine-implementable and completely absorbs the structural component.

Frobenius algebras The theme of this paper is to provide an analogous implementation for a more sophisticated class of categories. Our starting observation is that a variety of applications demand a richer structure than a SMC. The structure we shall focus on is the one of a (commutative, separable) *Frobenius algebra*: it consists of a monoid and a comonoid that interact according to the Frobenius (bottom-left below) and separability (bottom-right) laws.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

LICS '18, July 9–12, 2018, Oxford, United Kingdom

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-5583-4/18/07...\$15.00

<https://doi.org/10.1145/3209108.3209137>

Frobenius algebras are an increasingly common feature in diagrammatic calculi across diverse research threads.

- The ZX-calculus [12] is a diagrammatic theory for quantum computation featuring two Frobenius algebras, each having a specific physical meaning in terms of quantum observables.
- In compositional approaches to natural language processing (e.g. [27]) Frobenius algebras model the information passing from and to the relative clauses in a sentence.
- Frobenius algebras also form the backbone of the calculus of stateless connectors [10], of signal flow diagrams [4, 7], of Baez's network theory [1], and of Pavlovic's monoidal computer [26]. In these theories they allow for elementary signal operations such as copying, discarding and merging.
- In well-supported compact closed categories (now also called hypergraph categories), each object carries a Frobenius structure. Their use in algebraic approaches to computation was pioneered by Walters [11, 18, 21] and reprised in recent years, when constructions for these categories have been studied using (co)spans and (co)relations [9, 15, 17, 24, 31].

These applications share a core intuition: the Frobenius component allows dangling wires in a string diagram to fork, merge, be initialised, be discarded, and be converted from inputs to outputs (or vice-versa), resulting in a flexible manipulation of the interfaces (variables, memory cells, ...) of the represented system. The perspective of this work is to acknowledge the Frobenius equations as part of the structural rules of diagrammatic theories, distinguished from the domain-specific equations in a rewriting system. We contend that a satisfactory implementation of rewriting in the aforementioned theories ought to take the Frobenius component as structural

Roadmap to the implementation The goal of the paper is to identify a suitable DPO rewriting interpretation which is sound and complete for graphical reasoning modulo Frobenius structure.

The first challenge is dealing with *multiple* Frobenius algebras in *single-sorted* diagrammatic theories. As shown in [2, 32], the hypergraph representation can absorb the structure of a Frobenius algebra, but only one per sort. Thus in order to reach the correct combinatorial domain we need an intermediate step, where we move from a single-sorted to a *multi-sorted* setting. Concretely, our starting domain is a *prop* (i.e. a single-sorted SMC) $\mathbb{C}$, with n Frobenius algebras. From this, we build an n -coloured *prop* $\mathbb{D}_Y$, where each colour/sort carries one of the Frobenius structures. In order for this representation to be functorial, “colour-switch” connectors are added –e.g. $\text{---} \blacktriangleleft \text{---}$ switches from red to black– together with a set Y of equations of the form $\text{---} \blacktriangleleft \text{---} = \text{---} \blacktriangleright \text{---}$, making all sorts in $\mathbb{D}_Y$ isomorphic.

For example, the left-most diagram in (2), from $\mathbb{C}$, is interpreted in $\mathbb{D}_Y$ as follows

We shall prove that $\mathbb{C} \rightarrow \mathbb{D}_Y$ is actually an *equivalence* of coloured props, meaning that $\mathbb{D}_Y$ is a faithful representation of the information carried by $\mathbb{C}$. By working in the multi-sorted setting provided by $\mathbb{D}_Y$, we can now exploit the correspondence established in [32]

Rewriting C -coloured props with a Frobenius algebra on each colour $c \in C$.	$\Leftrightarrow$	DPO rewriting of hypergraphs with C -sorted nodes.
---	-------------------	--

To continue (3), where the diagram of $\mathbb{C}$ is interpreted in $\mathbb{D}_Y$, it is then interpreted as the hypergraph on the top-left below. Observe that there are two sorts of nodes, black and red, and hyperedges are labeled with the operations in the signature, namely $\text{---} \text{---} \text{---}$, and the switches $\text{---} \blacktriangleleft \text{---}$ and $\text{---} \blacktriangleright \text{---}$ of $\mathbb{D}_Y$. We also draw the (black) nodes representing the input and the output dangling wire.

The above derivation is the sound DPO rewriting implementation of (2). The graphs correctly “absorb” the Frobenius structure. However, in order to create the redex for α , a preliminary step is needed: redundant switches are removed using Y as a (strongly normalising) rewrite system. Thus our implementation uses the following, new correspondence

Rewriting props with C Frobenius algebras.	$\Leftrightarrow$	DPO rewriting of hyper- graphs with C -sorted nodes, in Y -normal form.
--	-------------------	---

Part of proving this correspondence is showing that Y -rewriting does not interfere with other rewriting rules, for instance by removing redexes. We are going to show how this is achieved by imposing a simple syntactic transformation on the rewriting rules.

Application: interacting bialgebras Our contribution ensures that diagrammatic theories with multiple Frobenius algebras become easier to study as rewriting systems, since the Frobenius equations become structural. We show this via an example involving the system $\mathbb{IB}$, where two Frobenius algebras interact as a bialgebra. This system is of particular interest as it forms the core of the ZX-calculus [12]. We give a simple rewrite strategy for computing a constant-depth reduced form for any $\mathbb{IB}$ -diagram and transforming such a reduced form into its colour-dual, which is the graphical analogue of the passage between a system of homogeneous linear equations and a spanning set of solution vectors.

Outline §2 provides a background on props, rewriting in props, and DPO rewriting. In §3 we construct the multi-sorted representation for single-sorted theories with Frobenius algebras. In §4 we discuss how this interpretation affects rewrite rules. Our main result, the implementation of rewriting modulo Frobenius, is Theorem 4.7: to the proof of its key step is devoted §5. Finally, §6 gives an application of the framework to the simplification of $\mathbb{B}$ diagrams.

2 Background

2.1 Props, Frobenius algebras, rewriting in a prop

Definition 2.1 (Prop). Given a finite set C of sorts (or colours), a C -coloured prop $\mathbb{A}$ is a symmetric strict monoidal category where the set of objects is C^* —the set of 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 $H: \mathbb{A} \rightarrow \mathbb{A}'$ that maps elements of C (seen as one-letter words) to elements of C' . Coloured props and their morphisms form a category CPROP.

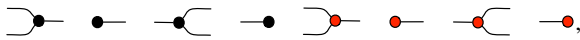
Fix a colour $\bullet$. Restricting to $\{\bullet\}$ -coloured props and morphisms between them yields a sub-category PROP of CPROP. $\{\bullet\}$ -coloured props are simply called *props* in the literature, and are equivalently described as symmetric monoidal categories where the objects are natural numbers and the monoidal product on objects is addition.

Definition 2.2 (Free prop on a theory). A *monoidal theory* is a pair (Σ, C) , where the *signature* Σ is a set of *operations* $o: v \rightarrow w$ with an *arity* v and a *coarity* w , with $v, w \in C^*$. We let $S_{\Sigma, C}$ denote the (free) C -coloured prop on (Σ, C) : its arrows are the Σ -terms quotiented by the laws of symmetric monoidal categories. Σ -terms are freely obtained by combining operations in Σ , *identities* $id: c \rightarrow c$ for each $c \in C$ and *symmetries* $\sigma_{c, d}: cd \rightarrow dc$ for each $c, d \in C$, by sequential $(:)$ and parallel $(\oplus)$ composition. That means, given terms $a: w_1 \rightarrow w_2$, $b: w_2 \rightarrow w_3$, and $a': v_1 \rightarrow v_2$, one constructs new terms such as $a \circ b: w_1 \rightarrow w_3$ and $a \oplus a': w_1 v_1 \rightarrow w_2 v_2$.

We will also refer to free props on a theory, in which case it is implicitly understood that the theory is single sorted, i.e. $C = \{\bullet\}$.

We shall adopt the graphical notation of string diagrams [28] for the arrows of $S_{\Sigma, C}$, drawing a box $\boxed{a}$ for an arrow $a: w_1 \rightarrow w_2$. In our examples we will mostly deal with coloured props on two colours, $\bullet$ and $\bullet$: in that case, the wire --- will indicate the sort $\bullet$ and wire --- the sort $\bullet$. E.g., $\boxed{a}$ has type $\bullet \bullet \rightarrow \bullet$.

Example 2.3. The prop *Frob* of *Frobenius algebras* is the free prop on signature $\{\bigcirc, \bigcirc, \bigcirc, \bigcirc\}$, quotiented by equations (2). In our work we will deal with the coproduct *Frob* + *Frob* in PROP, which can be also described freely as the prop on the signature



with equations two copies of (2), one for black and one for red.

It is helpful to see how the coproducts in PROP and CPROP differ. Rather than morphisms in the coproduct consisting of composition of morphisms from the component props living on the *same* sort, the coproduct in CPROP yield morphisms on two *different* sorts



again modulo two copies of the Frobenius equations. In particular, the generators of the $\bullet$ Frobenius algebra cannot be composed with the generators of the $\bullet$ Frobenius algebra. To emphasise this difference, we write the coproduct in CPROP as *Frob*($\bullet$) + *Frob*($\bullet$).

Even though coproducts in CPROP yield disjoint colours, the colours in two C -coloured props can be identified by using a pushout, as we see in the following example.

Example 2.4. The free C -coloured prop on the theory $(\emptyset, C)$ with an empty signature is written P_C and has arrows $w \rightarrow v$ the permutations of w into v (thus arrows exist only when the word v is an anagram of the word w). Given C -coloured props $\mathbb{A}$ and $\mathbb{A}'$, we use notation $\mathbb{A} +_C \mathbb{A}'$ for the pushout in CPROP of the span of the inclusions $\mathbb{A} \leftarrow P_C \rightarrow \mathbb{A}'$. Intuitively, $\mathbb{A} +_C \mathbb{A}'$ is the coproduct $\mathbb{A} + \mathbb{A}'$ where we have identified the copy from $\mathbb{A}$ and from $\mathbb{A}'$ of each $c \in C$. Thus $\mathbb{A} +_C \mathbb{A}$ is also a C -coloured prop.

Graphical reasoning with string diagrams is formally understood as a form of rewriting. The notion is given here for coloured props, but it clearly restricts to define rewriting in a (ordinary) prop.

Definition 2.5 (Syntactic rewriting). A *rewriting system* $\mathcal{R}$ in a coloured prop $\mathbb{A}$ is a set of *rewriting rules*, i.e. pairs $\langle l, r \rangle: v_1 \rightarrow v_2$ of morphisms $l, r: v_1 \rightarrow v_2$ in $\mathbb{A}$. Given $a, b: w_1 \rightarrow w_2$ in $\mathbb{A}$, a rewrites into b via $\mathcal{R}$, notation $a \Rightarrow_{\mathcal{R}} b$, if they are decomposable as follows, for some a_1 and a_2 , where $\langle l, r \rangle: v_1 \rightarrow v_2$ is a rule in $\mathcal{R}$.

$$\begin{aligned} \boxed{a} &= \boxed{a_1} \boxed{l} \boxed{a_2} \\ \boxed{b} &= \boxed{a_1} \boxed{r} \boxed{a_2} \end{aligned} \quad (4)$$

In this case, we say that a contains a *redex* for $\langle l, r \rangle$.

2.2 Rewriting with cospans of hypergraphs

Hypergraphs generalise directed graphs where edges connecting pairs of nodes are replaced by hyperedges connecting a list of source nodes to a list of target nodes. With the obvious notion of hypergraph morphisms, they form a category **Hyp**. A monoidal theory (Σ, C) can be seen as an hypergraph, where colours are nodes and operators are hyperedges (cf. [32]). We write $\mathbf{Hyp}_{\Sigma, C}$ for the category of hypergraphs with hyperedges labelled in Σ and nodes labelled in C , defined as the slice category $\mathbf{Hyp} \downarrow (\Sigma, C)$. We are now ready to introduce our combinatorial domain of interest.

Definition 2.6 (Hypergraphs with interfaces). Given a monoidal theory (Σ, C) , the category of (Σ, C) -labelled *hypergraphs with interfaces*, denoted $\mathbf{FTerm}_{\Sigma, C}$, is the C -coloured prop whose arrows $w \rightarrow v$ are cospans $w \rightarrow G \leftarrow v$ in $\mathbf{Hyp}_{\Sigma, C}$, quotiented by cospan isomorphism, and whose composition is given by pushout in $\mathbf{Hyp}_{\Sigma, C}$.

An equivalent definition of $\mathbf{FTerm}_{\Sigma, C}$ is as a certain sub-category of the category of cospans in $\mathbf{Hyp}_{\Sigma, C}$ [32]. The idea is that a morphism of $\mathbf{FTerm}_{\Sigma, C}$ is an hypergraph G with morphisms $w \rightarrow G$, $v \rightarrow G$ indicating along which nodes of G other hypergraphs can be “glued” on the left and on the right. Notice that here words $w, v \in C^*$ are seen as discrete hypergraphs (collections of C -labelled nodes) and act as the *interfaces* of G . The notation $\mathbf{FTerm}_{\Sigma, C}$ stands for “Frobenius termgraphs”, following [2, 32]— the terminology is justified by Proposition 2.8 below.

Double-pushout rewriting We now introduce a variant of double-pushout rewriting [14] with interfaces [18], abbreviated DPOI, that is suitable for rewriting in $\mathbf{FTerm}_{\Sigma, C}$. The notion can be abstractly formulated for (cospans over) arbitrary *adhesive categories* [23], and we use the fact that $\mathbf{Hyp}_{\Sigma, C}$ is indeed adhesive [32].

Definition 2.7 (DPOI rewriting). A DPOI rule is a pair of cospans $\langle J_1 \xrightarrow{r_1} L \xleftarrow{r_2} J_2, J_1 \xrightarrow{p_1} R \xleftarrow{p_2} J_2 \rangle$ in $\mathbf{Hyp}_{\Sigma, C}$. A DPOI rewriting system $\mathcal{R}$ is a set of DPOI rules. For $I_1 \xrightarrow{r_1} G \xleftarrow{r_2} I_2, I_1 \xrightarrow{p_1} H \xleftarrow{p_2} I_2$ cospans in $\mathbf{Hyp}_{\Sigma, C}$, we say that G rewrites into H via $\mathcal{R}$ with interfaces I_1 and I_2 , written $(I_1 \xrightarrow{r_1} G \xleftarrow{r_2} I_2) \rightsquigarrow_{\mathcal{R}} (I_1 \xrightarrow{s_1} H \xleftarrow{s_2} I_2)$, if there is a DPOI rule as above and cospan $J_1 + J_2 \rightarrow C \leftarrow I_1 + I_2$ making the diagram below commute and its two squares pushouts

$$\begin{array}{ccccc}
 L & \xleftarrow{[r_1, p_1]} & J_1 + J_2 & \xrightarrow{[r_2, p_2]} & R \\
 \downarrow & & \downarrow & & \downarrow \\
 G & \xleftarrow{q_1} & C & \xrightarrow{q_2} & H \\
 \uparrow & & \uparrow & & \uparrow \\
 I_1 + I_2 & \xleftarrow{[q_1, s_1]} & & \xrightarrow{[q_2, s_2]} &
 \end{array} \quad (5)$$

We now recall from [2, 32] the correspondence of DPOI with syntactic rewriting.¹ For the use of $+_C$ below, cf. Example 2.4.

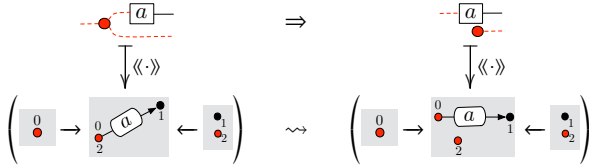
Proposition 2.8. Let C be a collection of sorts and Σ a signature over C . Then there is an isomorphism of C -coloured props

$$\langle \cdot \rangle : S_{\Sigma, C} +_C \sum_{c \in C} \text{Frob}(c) \cong \mathbf{FTerm}_{\Sigma, C}.$$

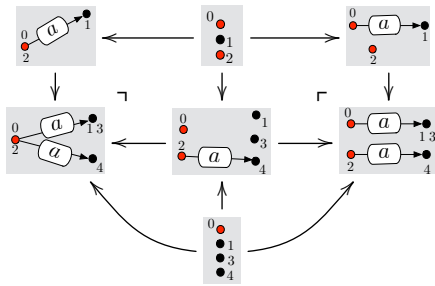
Let α be any rewriting rule on $S_{\Sigma, C} +_C \sum_{c \in C} \text{Frob}(c)$. Then

$$a \Rightarrow_{\alpha} b \quad \text{iff} \quad \langle \langle a \rangle \rangle \rightsquigarrow_{\langle \alpha \rangle} \langle \langle b \rangle \rangle.$$

Example 2.9. We give an illustration of the correspondence of Proposition 2.8. Fix $C = \{\bullet, \bullet\}$ and $\Sigma = \{\boxed{a}\}$, and take the C -coloured prop $S_{\Sigma, C} +_C (\text{Frob}(\bullet) + \text{Frob}(\bullet))$. We consider a rule α on such a prop, together with its interpretation in $\mathbf{FTerm}_{\Sigma, C}$



We conventionally use a grey background for hypergraphs to distinguish them from string diagrams, and numbers to indicate how the morphisms in the cospan are defined. Notice that these legs may be non-injective. Also, notice how the interpretation “absorbs” the Frobenius component. With the above rule, one can perform the syntactic rewriting step $\dots \Rightarrow_{\alpha} \dots$. It is implemented in $\mathbf{FTerm}_{\Sigma, C}$ via the DPOI rewriting step below



Outcomes of syntactic and DPOI rewriting coincide, modulo $\langle \cdot \rangle$.

¹The result in [2, 32] is actually stated for rewriting of single-interface hypergraphs. Our two-interfaces version is just a corollary, because a rewriting as in (5) exists for any pair of interfaces of G and H that are a coproduct decomposition of $I_1 + I_2$.

Props are also coloured props, thus Proposition 2.8 offers an implementation for rewriting in ordinary props. However, because props are single-sorted, it allows to “absorb” a single Frobenius structure in the rewriting procedure. Instead, we would like to implement rewriting for single-sorted theories with *multiple* Frobenius structures on that sort. This motivates next section.

3 The Polychromatic Interpretation

Throughout this and the next sections we fix a prop

$$\mathbb{C} := S_{\Sigma} + \text{Frob} + \text{Frob}$$

freely generated by a signature Σ and two Frobenius algebras (cf. Example 2.3), together with a rewriting system $\mathcal{R}$ on $\mathbb{C}$. Our goal is to provide a DPOI rewriting implementation for $\mathcal{R}$ -rewriting in $\mathbb{C}$.

Remark 1. Even though our exposition deals with rewriting modulo two Frobenius algebras, this is just for simplicity. The theory works for an arbitrary number of Frobenius algebras, via a straightforward generalisation of the developments presented in this paper.

Towards this goal, this section provides the intermediate step of representing $\mathbb{C}$ in terms of a coloured prop $\mathbb{D}_Y$; this setup will make our diagrammatic theory adapted to DPOI rewriting, via Proposition 2.8. $\mathbb{D}_Y$ is defined as follows. Consider a signature of “colour conversion” operations Γ , which we denote graphically as

$$\Gamma = \{ \text{---} \blacktriangleleft \text{---} : \bullet \rightarrow \bullet, \text{---} \blacktriangleright \text{---} : \bullet \rightarrow \bullet, \dots \},$$

together with equations

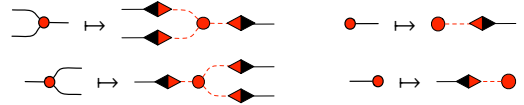
$$Y = \{ \text{---} \blacktriangleleft \blacktriangleright \text{---} = \text{---}, \text{---} \blacktriangleright \blacktriangleleft \text{---} = \text{---} \}.$$

Then $\mathbb{D}$ is defined as the $\{\bullet, \bullet\}$ -coloured prop

$$\mathbb{D} := S_{\Sigma \cup \Gamma} +_{\{\bullet, \bullet\}} (\text{Frob}(\bullet) + \text{Frob}(\bullet)) \quad (6)$$

and $\mathbb{D}_Y$ as $\mathbb{D}$ quotiented by Y . Notice that $\mathbb{D}_Y$ is generated by the same signature Σ as $\mathbb{C}$, including operations on sort $\bullet$, but also from the colour conversion operations in Γ . Whereas the two Frobenius structures in $\mathbb{C}$ were on the same sort, the two in $\mathbb{D}_Y$ are on two different sorts: $\bullet$ and $\bullet$. We use $+_{\{\bullet, \bullet\}}$ (cf. Example 2.4) to identify the sorts of $\text{Frob}(\bullet) + \text{Frob}(\bullet)$ with those of $S_{\Sigma \cup \Gamma}$.

We now define the “polychromatic interpretation” $(\cdot)_{\bullet} : \mathbb{C} \rightarrow \mathbb{D}_Y$. Intuitively, $(\cdot)_{\bullet}$ will “shift” one of the two Frobenius structures of $\mathbb{C}$ from sort $\bullet$ to sort $\bullet$, so that each sort hosts a single Frobenius algebra. Formally, $(\cdot)_{\bullet} : \mathbb{C} \rightarrow \mathbb{D}_Y$ is a morphism of coloured props, where $\mathbb{C}$ is here seen as a $\{\bullet\}$ -coloured prop. It suffices to define $(\cdot)_{\bullet}$ on the generating objects and arrows of $\mathbb{C}$. For objects, the single sort $\bullet$ of $\mathbb{C}$ is mapped to $\bullet$. For arrows, $(\cdot)_{\bullet}$ acts as the identity with the exception of the generators of the second Frobenius algebra



Notice that equations Y are needed in order for this functor to be well-defined. For instance, they ensure preservation of the separability law for the “red” Frobenius algebra

$$\begin{aligned}
 (\text{---} \bullet \text{---})_{\bullet} &= \text{---} \blacktriangleleft \bullet \blacktriangleright \text{---} \\
 &= \text{---} \blacktriangleleft \bullet \blacktriangleright \text{---} \\
 &= \text{---} \blacktriangleleft \bullet \blacktriangleright \text{---} = \text{---} = (\text{---})_{\bullet}.
 \end{aligned}$$

Remark 2. As $(\cdot)_\bullet$ has been defined in terms of the generators of $\mathbb{C}$ and $\mathbb{D}$, it will sometimes be useful to regard, by abuse of notation, $(\cdot)_\bullet$ as a mapping from formal string diagrams of the generators of $\mathbb{C}$ to $\mathbb{D}$ -morphisms. It is worth noting that this mapping would not extend to a well-defined functor from $\mathbb{C}$ to $\mathbb{D}$, since the latter is missing the equations of Υ .

It is essential for our developments that the polychromatic interpretation is without loss (or gain) of information. This is guaranteed by the following result.

Proposition 3.1. $(\cdot)_\bullet$ induces an equivalence $\mathbb{C} \simeq \mathbb{D}_\Upsilon$ in CPROP.

Proof. We have already shown that $(\cdot)_\bullet$ gives a strict monoidal functor from $\mathbb{C}$ to $\mathbb{D}_\Upsilon$. We define another functor $K : \mathbb{D}_\Upsilon \rightarrow \mathbb{C}$ on objects by letting $K(\bullet) = K(\bullet) = \bullet$. Since $\mathbb{D}_\Upsilon$ is presented by generators and equations, it suffices to say what it does on generators of $\mathbb{D}_\Upsilon$. It sends the generators Σ and the two Frobenius algebras to their monochromatic versions, whereas it sends each of the two colour-changers in Γ to $id_\bullet$. One can straightforwardly check this gives a well-defined, strict monoidal functor and that $K((\cdot)_\bullet) = Id_{\mathbb{C}}$. So, it remains only to give a natural isomorphism $\kappa : Id_{\mathbb{D}_\Upsilon} \cong (K(\cdot))_\bullet$.

For a word w in $\{\bullet, \bullet\}$, $(K(\cdot))_\bullet = \bullet^{|w|}$. So, let $\kappa_w : w \rightarrow \bullet^{|w|}$ be the unique monoidal product of $id_\bullet$ and $\dashv\dashv$ morphisms of the correct type. This is an isomorphism by construction. Naturality then follows from the definition of $(\cdot)_\bullet$ and the equations Υ . $\square$

Remark 3. The construction in this section ‘splits the difference’ between the two coproducts discussed in Example 2.3. As noted there, the embedding $U : \text{PROP} \rightarrow \text{CPROP}$ does not preserve coproducts. However, we can consider the introduction of the colour changers and equations Υ as a weak truncation operation on coloured props $(\cdot)_\bullet$, which forces all of the colours to be isomorphic to $\bullet$. Then, we do indeed have an equivalence of coloured props $U(\mathbb{A} + \mathbb{A}') \simeq (U(\mathbb{A}) + U(\mathbb{A}'))_\bullet$. Proposition 3.1 is then the instantiation of this fact for $\mathbb{A} := S_\Sigma + \text{Frob}$ and $\mathbb{A}' := \text{Frob}$.

4 Interpreting the Rewriting

Now that we represented of $\mathbb{C}$ as a coloured prop $\mathbb{D}_\Upsilon$, we can pass to hypergraphs by instantiating Proposition 2.8.

Corollary 4.1. There is an isomorphism of $\{\bullet, \bullet\}$ -coloured props between $\mathbb{D}_\Upsilon$ and $\text{FTerm}_{\Sigma \cup \Gamma, \{\bullet, \bullet\}}$ quotiented by $\langle\langle \Upsilon \rangle\rangle$.

With respect to rewriting, Corollary 4.1 is still unsatisfactory: rewriting in $\mathbb{D}_\Upsilon$ (and thus in $\mathbb{C}$) corresponds to DPOI rewriting in $\text{FTerm}_{\Sigma \cup \Gamma, \{\bullet, \bullet\}}$ only modulo the equations Υ . Clearly, it would be computationally obnoxious – and definitively not an implementation – to reason about rewriting of $\langle\langle \Upsilon \rangle\rangle$ -equivalence classes of graphs. We now proceed in steps towards a solution to the problem. First, henceforth we shall treat the two equations in Υ as rewriting rules on $\mathbb{D}$, with a left-to-right orientation. For simplicity we write Υ also the resulting rewriting system. We observe the following.

Lemma 4.2. Υ is terminating and confluent on $\mathbb{D}$.

Our next goal is then to show that rewriting modulo Υ can be simulated without loss of generality by putting a graph in Υ -normal form and then apply the rewriting rule. However, a naive application of this approach immediately poses problems, as it is shown by the following example.

Example 4.3. Suppose Σ contains an operation $\boxed{O}$ and consider the rewrite rule α defined as $\boxed{O} \dashv \dashv \Rightarrow \boxed{O}$. Under the interpretation $(\cdot)_\bullet$, it yields a rule $\alpha_\bullet$ in $\mathbb{D}$ defined as $\boxed{O} \dashv \dashv \Rightarrow \boxed{O}$. The translated rule conflicts with Υ , meaning that Υ can erase $\alpha_\bullet$ -redexes. For instance

This kind of problematic example motivates the following transformation on the rewrite rules of $\mathbb{D}$. As a preparatory step, we record the following lemma, where $\downarrow c$ is the unique Υ -normal form of a morphism c of $\mathbb{D}$, guaranteed by Lemma 4.2.

Lemma 4.4. Let l be a morphism of $\mathbb{D}$ in the image of $(\cdot)_\bullet$. Then, there is a morphism l' not containing any Υ -redex such that (in $\mathbb{D}$)

where $p_1 + j_1 + \dots + p_s + j_s = n$ and $q_1 + k_1 + \dots + q_r + k_r = m$, with $p_i, j_i, q_i, k_i \in \mathbb{N}$. Moreover, there is a unique such l' in $\mathbb{D}$ for each l .

Proof. Given l , by Lemma 4.2 there is a unique $\downarrow l$ in Υ -normal form. Because l is in the image of $(\cdot)_\bullet$, besides $\dashv\dashv$ and $\dashv\dashv$ it can only contain $\bullet$ -sorted operators, and external dangling wires are also of sort $\bullet$: thus every wire of sort $\bullet$ inside $\downarrow l$ can only be connected to the left boundary via a $\dashv\dashv$ and to the right boundary via a $\dashv\dashv$. We can use the laws of SMCs to “pull out” all such connecting operators towards the corresponding boundary: what we obtain is the left-hand side of (8). $\square$

We are now ready to introduce the transformation that will remove the conflicts between a rewrite rule and Υ .

Definition 4.5. Let α be a rewriting rule of type $\bullet^n \rightarrow \bullet^m$ on $\mathbb{D}$

$$n \boxed{l} m \Rightarrow_\alpha n \boxed{r} m$$

We obtain l' through Lemma 4.4.

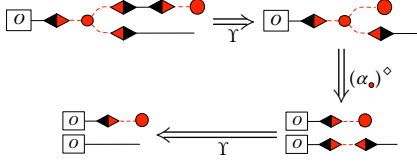
The rule $\alpha^\diamond$ is defined as

Given a rewriting system $\mathcal{R}$, we write $\mathcal{R}^\diamond = \{\alpha^\diamond \mid \alpha \in \mathcal{R}\}$.

It is instructive to show how this transformation neutralises the problem of (7).

Example 4.6. The rule $\alpha_\bullet$ from Example 4.3 is transformed into $(\alpha_\bullet)^\diamond$, defined as $\boxed{O} \dashv \dashv \Rightarrow \boxed{O}$. Observe that coexistence with Υ is not problematic anymore, as Υ cannot erase $(\alpha_\bullet)^\diamond$ -redexes.

For instance, in computation (7) we are not stuck anymore



We now have all the ingredients to state the main theorem of the paper: the DPOI rewriting implementation of rewriting in $\mathbb{C}$.

Theorem 4.7. *Let $\mathcal{R}$ be a rewriting system on $\mathbb{C}$. Then*

$$a \Rightarrow_{\mathcal{R}} b \quad \text{iff} \quad \Downarrow \langle \langle a \rangle \rangle \rightsquigarrow_{\langle \langle \mathcal{R} \rangle \rangle}^{\star} \Downarrow \langle \langle b \rangle \rangle$$

Contrary to the situation depicted at the beginning of the section, in Theorem 4.7 DPOI rewriting in the combinatorial domain is “on-the-nose”: instead of dealing with $\langle \langle Y \rangle \rangle$ -equivalence classes of hypergraphs, we can now deal exclusively with $\langle \langle Y \rangle \rangle$ -normal forms, which thanks to Lemma 4.2 are straightforward to compute.

The proof of Theorem 4.7 will go in steps. The theory developed so far ensures a correspondence between

- rewriting in $\mathbb{C}$ and rewriting in $\mathbb{D}_Y$, thanks to Proposition 3.1;
- rewriting in $\mathbb{D}$ and rewriting in $\mathbf{FTerm}_{\Sigma \cup \Gamma, \{\bullet, \circ\}}$, thanks to Proposition 2.8.

Thus the only missing link to complete the correspondence in Theorem 4.7 is to adequately represent rewriting in $\mathbb{D}_Y$ as rewriting in $\mathbb{D}$. This is the remit of the next section.

5 Adequacy of the Implementation

For the purposes of this section, let $\mathcal{R}$ be a rewriting system on $\mathbb{D}$. We focus on the only missing piece of the proof of Theorem 4.7: showing that the rule transformation of Definition 4.5 provides an adequate implementation for $\mathcal{R}$ -rewriting modulo Y in $\mathbb{D}$.

Proposition 5.1. *Let c and d be arrows in $\mathbb{D}$. Then*

$$c \stackrel{\star}{\Rightarrow}_Y \Rightarrow_{\mathcal{R}} \stackrel{\star}{\Leftarrow}_Y d \quad \text{iff} \quad \downarrow c \Rightarrow_{\mathcal{R}^\diamond} \stackrel{\star}{\Rightarrow}_Y \downarrow d.$$

The proof will follow from Propositions 5.2 and 5.5. For the right-to-left direction (completeness), we can prove a stronger statement.

Proposition 5.2. *$c \Rightarrow_{\mathcal{R}^\diamond} d$ implies $c \stackrel{\star}{\Rightarrow}_Y \Rightarrow_{\mathcal{R}} \stackrel{\star}{\Leftarrow}_Y d$.*

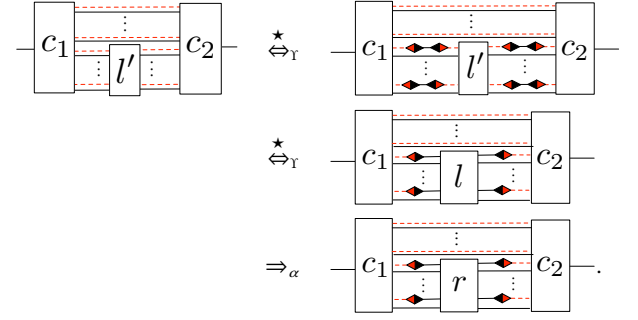
Proof. For the sake of readability, all the diagrams in the proofs of this section are depicted with unlabelled wires— it is intended that each wire stands for a number of parallel wires of the same type, arbitrary but compatible with its position in the diagram.

By assumption c has a redex for $\alpha^\diamond$ for some rule $\alpha \in \mathcal{R}$. If α is given by $l \Rightarrow r$, then $\alpha^\diamond$ rewrites c as follows

$$\begin{array}{c} \text{---} \boxed{C} \text{---} \\ = \\ \text{---} \boxed{C_1} \text{---} \boxed{l'} \text{---} \boxed{C_2} \text{---} \end{array} \quad (9)$$

$$\Rightarrow_{\alpha^\diamond} \text{---} \boxed{C_1} \text{---} \boxed{r} \text{---} \boxed{C_2} \text{---} \quad (10)$$

In light of (9), c modulo Y contains a redex for α as well



where the second step is justified by the definition of l' as in (8). Thus rewriting with α modulo Y gives the same outcome as applying $\alpha^\diamond$. This proves the statement. $\square$

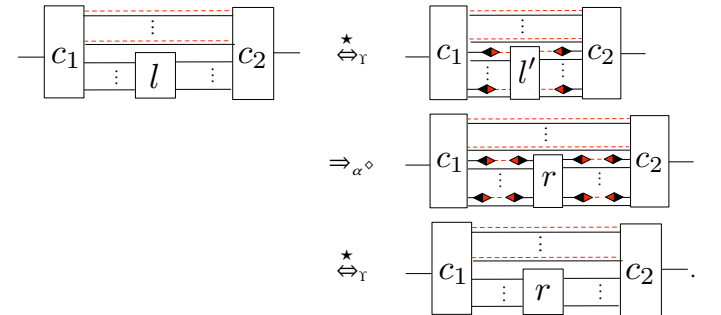
The left-to-right direction (soundness) of Proposition 5.1 requires more work. First, we have that $\mathcal{R}^\diamond$ is as powerful as $\mathcal{R}$, modulo Y .

Lemma 5.3. *$c \stackrel{\star}{\Rightarrow}_Y \Rightarrow_{\mathcal{R}} \stackrel{\star}{\Leftarrow}_Y d$ iff $c \stackrel{\star}{\Rightarrow}_Y \Rightarrow_{\mathcal{R}^\diamond} \stackrel{\star}{\Leftarrow}_Y d$.*

Proof. It suffices to show that $c \Rightarrow_{\mathcal{R}} d$ implies $c \stackrel{\star}{\Rightarrow}_Y \Rightarrow_{\mathcal{R}^\diamond} \stackrel{\star}{\Leftarrow}_Y d$ and $c \Rightarrow_{\mathcal{R}^\diamond} d$ implies $c \stackrel{\star}{\Rightarrow}_Y \Rightarrow_{\mathcal{R}} \stackrel{\star}{\Leftarrow}_Y d$. For the left-to-right implication, the assumption is that c contains a redex for a rule $\alpha \in \mathcal{R}$, say of the form $l \Rightarrow r$.

$$\begin{array}{c} \text{---} \boxed{C} \text{---} \\ = \\ \text{---} \boxed{C_1} \text{---} \boxed{l} \text{---} \boxed{C_2} \text{---} \\ \Rightarrow_{\alpha} \\ \text{---} \boxed{C_1} \text{---} \boxed{r} \text{---} \boxed{C_2} \text{---} \end{array} \quad (11)$$

Then, modulo Y , c also contains a redex for $(l \Rightarrow r)^\diamond$. Applying this rule yields the same outcome, modulo- Y , as (11).



This proves the left-to-right implication of the statement. The right-to-left implication is given by Proposition 5.2. $\square$

The next step is to show that $\Rightarrow_{\mathcal{R}^\diamond}$ satisfies a “diamond property” with respect to Y . This property implies that Y -rewriting does not interfere with $\mathcal{R}^\diamond$ -rewriting— whence the latter can be assumed without loss of generality to work on arrows in Y -normal form, as in the desired implementation (Proposition 5.1). As shown in Example 4.3, the diamond property fails for arbitrary rewriting systems and justifies the introduction of the transformation $(\cdot)^\diamond$.

Lemma 5.4 (Diamond Property). *If $c \Rightarrow_{\mathcal{R}^\diamond} d$ and $c \Rightarrow_Y e$ then there exists an f such that $d \Rightarrow_Y f$ and $e \Rightarrow_{\mathcal{R}^\diamond} f$.*

Proof. This is immediate from the fact that, by Definition 4.5, $\mathcal{R}^\diamond$ contains no Υ -redex. Therefore, $\mathcal{R}^\diamond$ and Υ are orthogonal rewriting systems (i.e. they have no critical pairs between each other). $\square$

We are now ready to show soundness.

Proposition 5.5. $c \xrightarrow{\star} \Upsilon \Rightarrow_{\mathcal{R}} \Upsilon \xrightarrow{\star} d$ implies $\downarrow c \Rightarrow_{\mathcal{R}^\diamond} \Upsilon \xrightarrow{\star} \downarrow d$.

Proof. Since Υ is confluent and terminating (Lemma 4.2), the conclusion is equivalent to $\downarrow c \Rightarrow_{\mathcal{R}^\diamond} \Upsilon \xrightarrow{\star} d$, so we focus on this statement.

Assume $c \xrightarrow{\star} \Upsilon \Rightarrow_{\mathcal{R}} \Upsilon \xrightarrow{\star} d$. By Lemma 5.3, this implies $c \xrightarrow{\star} \Upsilon \Rightarrow_{\mathcal{R}^\diamond} \Upsilon \xrightarrow{\star} d$.

Since Υ is confluent and terminating, this implies $c \xrightarrow{\star} \Upsilon \downarrow c \xrightarrow{\star} \Upsilon \Rightarrow_{\mathcal{R}^\diamond} \Upsilon \xrightarrow{\star} d$.

We can now drop the first part of the rewrite sequence and focus on $\downarrow c \xrightarrow{\star} \Upsilon \Rightarrow_{\mathcal{R}^\diamond} \Upsilon \xrightarrow{\star} d$. By repeatedly applying Lemma 5.4, we can commute $\Rightarrow_{\mathcal{R}^\diamond}$ through $\Upsilon \xrightarrow{\star}$ to obtain $\downarrow c \Rightarrow_{\mathcal{R}^\diamond} \Upsilon \xrightarrow{\star} \Upsilon \xrightarrow{\star} d$. Finally, merging $\Upsilon \xrightarrow{\star}$ and $\Upsilon \xrightarrow{\star}$ yields $\downarrow c \Rightarrow_{\mathcal{R}^\diamond} \Upsilon \xrightarrow{\star} d$ as required. $\square$

5.1 Proof of Theorem 4.7

We now have all the pieces to conclude the proof of our main result.

Proof of Theorem 4.7. First, we have a correspondence at the level of syntactic rewriting (Definition 2.5) in the props $\mathbb{C}$ and $\mathbb{D}_\Upsilon$

$$\boxed{a \Rightarrow_{\mathcal{R}} b \text{ in } \mathbb{C}} \quad \text{iff} \quad \boxed{a_\bullet \Rightarrow_{\mathcal{R}_\bullet} b_\bullet \text{ in } \mathbb{D}_\Upsilon} \quad (12)$$

This is ensured by the fact that $(\cdot)_\bullet$ is a functorial and fully-faithful mapping. Second, we interpret Υ as a set of rewrite rules instead of a set of equations. Then, rewriting in $\mathbb{D}_\Upsilon$ is just the same as rewriting in $\mathbb{D}$ modulo Υ -rewriting. Starting from the right-hand side of (12)

$$\boxed{a_\bullet \Rightarrow_{\mathcal{R}_\bullet} b_\bullet \text{ in } \mathbb{D}_\Upsilon} \quad \text{iff} \quad \boxed{a_\bullet \xrightarrow{\star} \Upsilon \Rightarrow_{\mathcal{R}_\bullet} \Upsilon \xrightarrow{\star} b_\bullet \text{ in } \mathbb{D}} \quad (13)$$

where $a_\bullet$ and $b_\bullet$ are understood on the right as arrows of $\mathbb{D}$, cf. Remark 2. Third, we use Proposition 5.1 to give an implementation for rewriting modulo- Υ . Starting from the right-hand side of (13)

$$\boxed{a_\bullet \xrightarrow{\star} \Upsilon \Rightarrow_{\mathcal{R}_\bullet} \Upsilon \xrightarrow{\star} b_\bullet \text{ in } \mathbb{D}} \quad \text{iff} \quad \boxed{\downarrow a_\bullet \Rightarrow_{\mathcal{R}^\diamond} \Upsilon \xrightarrow{\star} \downarrow b_\bullet \text{ in } \mathbb{D}} \quad (14)$$

Last, Corollary 4.1 and Proposition 2.8 yield the correspondence between rewriting in $\mathbb{D}$ and DPO-rewriting in $\mathbf{FTerm}_{\Sigma \cup \Gamma, \{\bullet, \bullet\}}$. Starting from the right-hand side of (14)

$$\boxed{\downarrow a_\bullet \Rightarrow_{\mathcal{R}^\diamond} \Upsilon \xrightarrow{\star} \downarrow b_\bullet \text{ in } \mathbb{D}} \quad \text{iff} \quad \boxed{\langle \downarrow a_\bullet \rangle \rightsquigarrow_{\langle \mathcal{R}^\diamond \rangle} \Upsilon \rightsquigarrow_{\langle \Upsilon \rangle} \langle \downarrow b_\bullet \rangle \text{ in } \mathbf{FTerm}_{\Sigma \cup \Gamma, \{\bullet, \bullet\}}} \quad (15)$$

Notice that $\langle \downarrow a_\bullet \rangle = \downarrow \langle a_\bullet \rangle$, where the normal form on the right is computed in $\mathbf{FTerm}_{\Sigma \cup \Gamma, \{\bullet, \bullet\}}$ with the rules $\langle \Upsilon \rangle$. To conclude, by chaining (12) to (15), we obtain the statement of the theorem. $\square$

6 Application: Interacting Bialgebras

The natural use-cases for rewriting modulo Frobenius equations come from the study of multiple, interacting Frobenius algebras.

Perhaps the best studied example is a pair of such algebras whose respective monoid and comonoid structures interact as bialgebras

$$\begin{array}{c} \text{Diagram (16):} \\ \text{Left: } \text{cup} = \text{cap} \text{ (with red and black dots)} \\ \text{Middle: } (c1) = (c2) \text{ (with red and black dots)} \\ \text{Right: } (u) = \text{empty box} \end{array} \quad (16)$$

$$\begin{array}{c} \text{Diagram (17):} \\ \text{Left: } (b') = \text{cap} \text{ (with red and black dots)} \\ \text{Middle: } (c1') = (c2') \text{ (with red and black dots)} \\ \text{Right: } (u') = \text{empty box} \end{array} \quad (17)$$

and whose induced ‘cup’ and ‘cap’ maps coincide

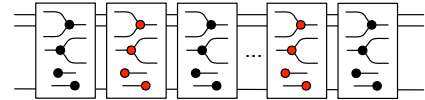
$$\text{Diagram (18):} \quad \text{cup} = \text{cap} \quad (18)$$

This system is referred to as $\mathbb{IB}$ [5] and forms the core of the *ZX-calculus* [12], which has recently been extended to give sound and complete equational theories for approximately [20] and fully [25] universal families of quantum circuits.

From the rules above, one can derive (see e.g. [12]) the following two rules, which will soon be useful

$$\begin{array}{c} \text{Diagram (19):} \\ \text{Left: } (d) = \text{cup} \text{ (with red and black dots)} \\ \text{Middle: } (h) = \text{cap} \text{ (with red and black dots)} \\ \text{Right: } (u1) = \text{empty box}, (u2) = \text{empty box} \end{array} \quad (19)$$

A generic diagram composed of generators from these two Frobenius algebras consists of arbitrarily many alternating layers of $\bullet$ and $\bullet$ generators



What we will show in this section is a rewriting strategy for turning any such diagram into a $\bullet$ -reduced form that consists of just four layers: an initial layer of $\bullet$ -comonoid structure, followed by $\bullet$ -monoid structure, followed by $\bullet$ -comonoid structure, followed by a final layer of $\bullet$ -monoid structure

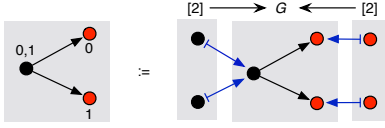
$$\text{Diagram (20):} \quad \text{Four layers of generators (cup/cap with dots)} \quad (20)$$

We call this the $\bullet$ -reduced form because there are no internal layers of $\bullet$ generators. We now give a characterisation of these forms in terms of their associated hypergraphs with interfaces. To express hypergraphs with interfaces compactly and unambiguously, we adopt the following notational conventions

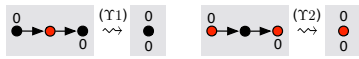
1. The hyperedges corresponding to cup and cap are depicted as unlabelled, directed edges between nodes of appropriate colour, hence

$$\begin{array}{c} \text{Diagram (21):} \\ \text{Left: } \text{cup} = \langle \text{cup} \rangle \\ \text{Right: } \text{cap} = \langle \text{cap} \rangle \end{array}$$

2. Like in §2.2, interface maps are specified by labels above and below nodes. For a hypergraph with interfaces $m \xrightarrow{f} G \xleftarrow{g} n$, a label $i_1, \dots, i_k$ above a node v indicates that $f^{-1}(v) = \{i_1, \dots, i_k\}$, whereas such a label below indicates that $g^{-1}(v) = \{i_1, \dots, i_k\}$. For example, the following hypergraph with interfaces is abbreviated as



Using these conventions, the rules in the system Υ can be written as the following two DPOI rules



Hence, normalising with respect to Υ contracts away any node with precisely one in-edge and one out-edge. We are now ready to characterise $\bullet$ -reduced forms.

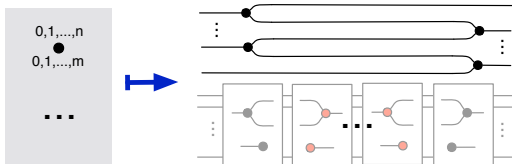
Proposition 6.1. *A string diagram generated by two Frobenius algebras $\bullet$ and $\bullet$ is in $\bullet$ -reduced form as in (20) (modulo Frobenius equations) if and only if its associated hypergraph with interfaces $I_1 \rightarrow G \leftarrow I_2$ satisfies the following conditions. G is directed acyclic and every $\bullet$ node in G is either*

- (I) in the image of a single node in I_1 and has no in-edges,
- (O) in the image of a single node in I_2 and has no out-edges, or
- (IO) in the image of one or more nodes in **both** I_1 and I_2 .

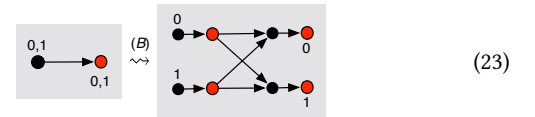
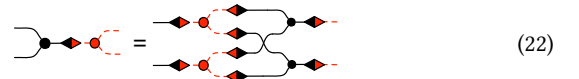
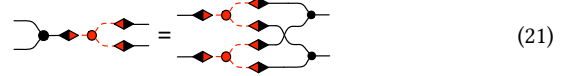
Proof. When hypergraph nodes representing Frobenius algebra generators are composed, they fuse together. Hence, the nodes in G correspond to maximal connected components of Frobenius algebra generators of the same colour.

First, suppose a string diagram is in the form of (20). Then, each $\bullet$ node in the hypergraph G corresponds to a maximal connected component of $\bullet$ Frobenius generators in (20). We first note that any (co)unit connected to a (co)multiplication can be reduced away. Hence, we need to consider only 5 cases for connected components of $\bullet$ generators: (1) a counit applied to an input wire, (2) a unit applied to an output wire, (3) a tree of comultiplications applied to an input wire, (4) a tree of multiplications connected to an output wire, or (5) a connected component of cases (3) and (4). Cases (1) and (3) yield a $\bullet$ node of type (I). Cases (2) and (4) yield a node of type (O), and case (5) yields a node of type (IO).

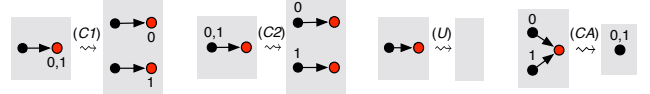
Conversely, we can interpret each of the $\bullet$ nodes of types (I) and (O) as cases (1)-(4) described above. The only slightly difficult case is nodes of type (IO). These can be interpreted as a ‘zig-zag’ of $\bullet$ comultiplications in the first layer of (20) and multiplications in the last layer, with no $\bullet$ generators in between



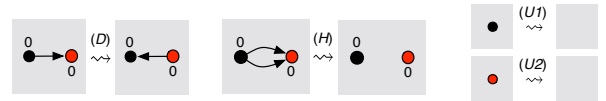
Crucially, an hypergraph with interfaces which satisfies the conditions above contains no *interior* $\bullet$ nodes, i.e. nodes not in the image of I_1 or I_2 . Eliminating these nodes will form the main component of the strategy below. In order to obtain a hypergraph with interfaces satisfying these conditions, we first perform the transformation of the interacting bialgebra rules into a DPOI rewrite system. This is a mechanical procedure, but for clarity, we will show it explicitly for the rule (b). Following the recipe of Theorem 4.7, we first use $(\cdot)^\circ$ to get the polychromatic interpretation —(21) below— then apply $(\cdot)^\diamond$ to shift the colour change maps on inputs/outputs to the right-hand side —(22)— and finally apply $\langle\langle \cdot \rangle\rangle$ to interpret



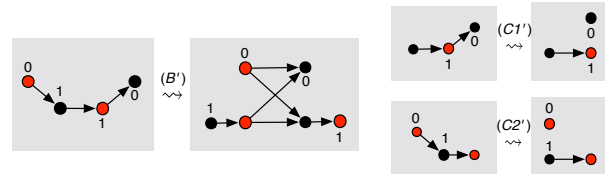
Similarly, the rules (cp1), (cp2), (u), and (ca) produce the following



We have mentioned the derived equations in box (19) because, once we translate them into DPOI rules, we see that rule (d) allows to reverse the direction of an arbitrary edge, rule (h) to delete parallel edges, and rules (u1) and (u2) to delete single, isolated nodes



Note the rule (D) (and its converse) render the additional equations in box (17) and the rule (cu) redundant, since they are the same as the rules above, but with some of the directions reversed. Hence, the strategy we will describe only relies on the DPOI rules above. First, we derive variations of the laws (B), (C1), and (C2)

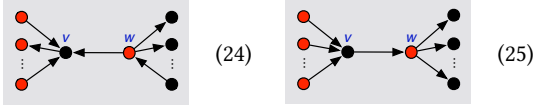


Unlike the original rules, these variations will terminate in the strategy below. They are obtained by applying the original rule to their LHS, then normalising with respect to the rules (Y1) and (Y2).

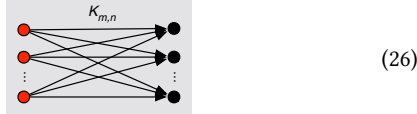
Reduction Strategy We begin with a hypergraph with interfaces $I_1 \rightarrow G \leftarrow I_2$, whose interfaces I_1, I_2 are all of the $\bullet$ -sort. It should be understood that after every rewrite step, the graph is normalised with respect to rules (Y1) and (Y2). The strategy proceeds as follows

□

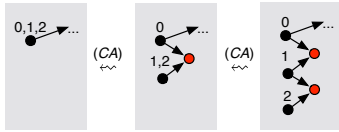
1. Reduce as much as possible using rules (U1), (U2), and (H), using the inverse of rule (D) to create additional redexes for (H) if necessary.
2. If there are no interior $\bullet$ nodes, go to step 5. Otherwise, fix a full sub-graph consisting of a single interior $\bullet$ node v , a neighbouring $\bullet$ node w , and all of the nodes adjacent to v and w . This looks like the graph (24) below, where the directions of the edges can be arbitrary. Apply the rule (D) or its converse to direct all of the edges in the sub-graph from left to right, thus obtaining (25)



3. Reduce sub-graph (25) as much as possible using rules (B'), (C1'), (C2'), and (U). This will eliminate v and w , yielding a totally connected bipartite graph between their neighbours



4. If there are remaining interior $\bullet$ nodes, go to step 1.
5. If a $\bullet$ node is in the image of multiple nodes in I_1 and no nodes in I_2 , apply the converse of rule (CA) to split it into multiple $\bullet$ nodes connected by $\bullet$ nodes. For example



We split nodes only in the image of I_2 similarly.

6. Apply (D) or its converse to direct the remaining edges from the image of I_1 , to the $\bullet$ nodes, then to the image of I_2 .

Theorem 6.2. *The Reduction Strategy above terminates and yields a graph in reduced form.*

Proof. First we show termination. Steps 1, 2, 3, and 6 clearly terminate after finitely many rule applications. We now show that step 4 terminates yielding a graph of the shape $K_{m,n}$.

If either v or w has no additional neighbours, then (B') has no redex, whilst exactly one of the rules (C1'), (C2'), or (U) will have a redex. Applying any of these rules will strictly decrease the edges in (25) without creating a redex for (B'), so step 4 will terminate.

Suppose that both v and w have at least one additional neighbour. Then only the rule (B') has a redex. Applying (B') will propagate one in-edge of a $\bullet$ node to the right, past an out-edge of a $\bullet$ node, possibly creating more $\bullet$ and $\bullet$ nodes in the process. However, each of the new $\bullet$ nodes will be 'closer' to the output of the graph, in the sense that the total number of out-edges for successor $\bullet$ nodes will have decreased. Repeatedly applying (B') will thus terminate once all of the in-edges on $\bullet$ nodes have been propagated to the right as much as possible. Moreover, application of the rule (B') preserves the number of paths from the left nodes in (25) to the right nodes. Since precisely one path exists from every node on the left to every node on the right, this procedure yields $K_{m,n}$ at termination.

Each iteration of steps 1-4 reduces the number of interior $\bullet$ nodes by 1. Hence, it terminates after n iterations for n interior $\bullet$ nodes with no interior $\bullet$ nodes. Step 5 guarantees all remaining, non-interior $\bullet$ nodes are of the form (I), (O), or (IO) as in Proposition 6.1, and step 6 guarantees the directed acyclicity conditions. $\square$

We close this section with a brief discussion about the $\bullet$ -reduced form, and its relationship to the semantics of $\mathbb{IB}$. It was shown in [5] that the prop for $\mathbb{IB}$ is isomorphic to the prop $\mathbf{LinRel}(\mathbb{Z}_2)$ of $\mathbb{Z}_2$ -linear relations. That is, morphisms $S : m \rightarrow n$ are linear sub-spaces $S \subseteq \mathbb{Z}_2^m \times \mathbb{Z}_2^n \cong \mathbb{Z}_2^{m+n}$, $\oplus$ is given by direct product, and composition is done relation-style

$$(v, w) \in (S; T) \iff \exists u. (v, u) \in S, (u, w) \in T \quad (27)$$

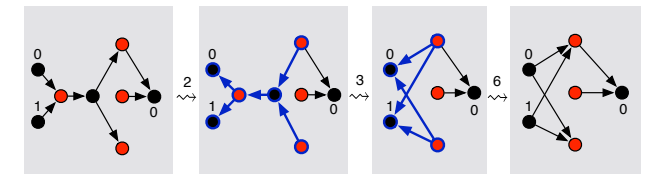
As explained in [30], the $\bullet$ -reduced form (called the *cospan form* therein) enables us to 'read off' S as a homogeneous system of equations (or equivalently, as a basis for $S^\perp$). In this form, $\bullet$ nodes correspond to variables, and $\bullet$ nodes to equations, whose LHS and RHS consist of those variables connected by in-edges and out-edges, respectively. For example, the diagram below represents the space of solutions to the following system of equations

$$\begin{pmatrix} x_0 + x_1 & e_0 & y_0 \\ 0 & e_1 & y_0 \\ x_0 + x_1 & e_2 & 0 \end{pmatrix} \quad (28)$$

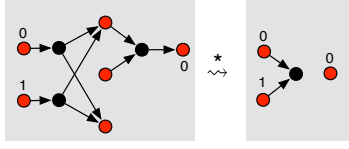
This interpretation gives a semantical view of the **Reduction Strategy** as a quantifier elimination procedure. The main purpose of the procedure is to eliminate interior $\bullet$ -nodes. Since these nodes arise from sequential compositions in $\mathbf{LinRel}(\mathbb{Z}_2)$, equation (27) tells that they correspond to existentially quantified variables

$$\begin{pmatrix} x_0 + x_1 & d_0 & z_0 \\ z_0 & e_0 & y_0 \\ 0 & e_1 & y_0 \\ z_0 & e_2 & 0 \end{pmatrix} \quad \exists z_0.$$

The core of the **Reduction Strategy** are steps 2 and 3. The former isolates an existentially quantified variable z on the LHS of an equation e , and step 3 substitutes any occurrence for that variable with its RHS, simultaneously eliminating z and e . Applying this procedure to the diagram above yields the $\bullet$ -reduced form in (28)



Since everything in $\mathbb{IB}$ and the **Reduction Strategy** is colour-symmetric, we can use the same strategy to compute the analogous $\bullet$ -reduced forms. To do so, we first pre- and post-compose with colour changers to obtain a graph with an interface consisting entirely of $\bullet$ -nodes, then apply the **Reduction Strategy** with the colours reversed. Applying this to example (28) yields



This again gives a canonical representation of a sub-space S (called the *span form*), but this time as a basis for S itself, rather than $S^\perp$ [30]. $\bullet$ -nodes correspond to basis vectors, where the presence of an edge indicates a 1 in the corresponding position. The final diagram in the rewrite sequence above represents S as $\text{span}\{((1, 1), (0))\} \subseteq \mathbb{Z}_2^2 \times \mathbb{Z}_2$, which is indeed the space of solutions to the system given in (28).

Note that we have focused on the case of $\mathbb{B}$ and $\mathbb{Z}_2$ -linear equations because it is the simplest. It was shown in [8] that this system generalises straightforwardly to a system $\mathbb{B}_F$ that has as its prop $\text{LinRel}(F)$ for an arbitrary field F . In that case, we introduce a family of generators for each $a \in F \setminus \{0, 1\}$ that give weights to edges. By modifying the **Reduction Strategy** to account for these weights, we can still obtain a (slightly more elaborate) procedure for removing internal nodes. This then gives the graphical analogue to quantifier elimination over an arbitrary field F .

Interestingly, this graphical version of quantifier elimination is *inherently compositional*. It is possible to introduce generators and relations, breaking the semantic connection with $\text{LinRel}(F)$, while using **Reduction Strategy** on sub-diagrams in the $\mathbb{B}_F$ fragment. This technique can exploit the fact that the ZX-calculus contains $\mathbb{B}$ to perform peephole optimisations on quantum circuits, even though the latter have a more complex semantics than $\text{LinRel}(\mathbb{Z}_2)$.

For yet another perspective, recall that $\mathbb{B}$ enjoys a modular characterisation in terms of distributive laws of props [5], which prescribes that each diagram can be turned into *cospan form* and *span form*. As observed, these correspond to $\bullet$ -reduced and $\circ$ -reduced forms respectively: thus our result provides algorithmic means to reach them, that was lacking in the abstract picture. It also fills the main gap in formulating the realisability procedure for signal flow graphs [6, 7] entirely as a diagram rewriting procedure.

7 Conclusions

The aim of this paper was to establish an implementation of rewriting modulo Frobenius algebras, in terms of DPOI hypergraph rewriting. This posed a series of technical challenges that we solved in a principled way: in particular, it turns out that our “polychromatic interpretation” is an equivalence, and thus adequately captures the information carried by the original single-sorted theory. We believe that this interpretation is part of a broader picture. Namely, the forgetful functor $U : \text{PROP} \rightarrow \text{CPROP}$ mentioned in Remark 3 has a left adjoint involving the ‘weak truncation’ operation described in §3. We plan to investigate this construction and elucidate its connection to the polychromatic interpretation in future work.

As illustrated in §1, diagrammatic theories featuring multiple Frobenius algebras are widespread across different disciplines. Our result drastically reduces the number of equations to consider when reasoning about these theories, thus simplifying the path to termination and normal form result, as well as easing their mechanisation in proof assistants. We concluded our paper with one such demonstration: a rewriting strategy for putting diagrams of the theory $\mathbb{B}$ in reduced forms. We mentioned how these forms have linear algebraic significance and are also directly relevant for the study

of interacting bialgebras in terms of distributive laws. Further applications to the theories based on $\mathbb{B}$, such as the ZX-calculus and the calculus of signal flow diagrams, as well as different calculi featuring multiple Frobenius algebras, remain to be explored.

Acknowledgements. Fabio Zanasi acknowledges support from EP-SRC grant n. EP/R020604/1.

References

- [1] J. Baez and J. Erbele. 2015. Categories in control. *Theory and Application of Categories* 30 (2015), 836–881.
- [2] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobociński, and F. Zanasi. 2016. Rewriting modulo symmetric monoidal structure. In *LICS 2016*. ACM, 710–719.
- [3] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobociński, and F. Zanasi. 2017. Confluence of graph rewriting with interfaces. In *ESOP 2017 (LNCS)*, Vol. 10201. Springer, 141–169.
- [4] F. Bonchi, P. Sobociński, and F. Zanasi. 2014. A categorical semantics of signal flow graphs. In *CONCUR 2014 (LNCS)*, Vol. 8704. Springer, 435–450.
- [5] F. Bonchi, P. Sobociński, and F. Zanasi. 2014. Interacting bialgebras are Frobenius. In *FoSSaCS 2014 (LNCS)*, Vol. 8412. Springer, 351–365.
- [6] F. Bonchi, P. Sobociński, and F. Zanasi. 2015. Full abstraction for signal flow graphs. In *POPL 2015*. ACM, 515–526.
- [7] F. Bonchi, P. Sobociński, and F. Zanasi. 2017. The calculus of signal flow diagrams I: Linear relations on streams. *Information and Computation* 252 (2017), 2–29.
- [8] F. Bonchi, P. Sobociński, and F. Zanasi. 2017. Interacting Hopf algebras. *Journal of Pure and Applied Algebra* 221, 1 (2017), 144 – 184.
- [9] R. Bruni and F. Gadducci. 2001. Some algebraic laws for spans (and their connections with multirelations). In *RelMIS 2001 (ENTCS)*, Vol. 44(3). Elsevier, 175–193.
- [10] R. Bruni, I. Lanese, and U. Montanari. 2006. A basic algebra of stateless connectors. *Theoretical Computer Science* 366, 1–2 (2006), 98–120.
- [11] A. Carboni and R. F. C. Walters. 1987. Cartesian bicategories I. *Journal of Pure and Applied Algebra* 49, 1–2 (1987), 11–32.
- [12] B. Coecke and R. Duncan. 2008. Interacting quantum observables. In *ICALP 2008 (LNCS)*, Vol. 5216. Springer, 298–310.
- [13] B. Coecke and A. Kissinger. 2016. *Picturing Quantum Processes. A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press.
- [14] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Loewe. 1997. Algebraic approaches to graph transformation, part I: Basic concepts and double pushout approach. In *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific, 163–246.
- [15] B. Fong. 2016. *The Algebra of Open and Interconnected Systems*. Ph.D. Dissertation. University of Oxford.
- [16] B. Fong, P. Rapisarda, and P. Sobociński. 2016. A categorical approach to open & interconnected dynamical systems. In *LICS 2016*. ACM, 495–504.
- [17] B. Fong and F. Zanasi. 2017. A universal construction for (co)relations. In *CALCO 2017 (LIPIcs)*, Vol. 72. Leibniz-Zentrum für Informatik, 12:1–12:6.
- [18] F. Gadducci and R. Heckel. 1998. An inductive view of graph transformation. In *WADT 1997 (LNCS)*, Vol. 1376. Springer, 223–237.
- [19] D. R. Ghica, A. Jung, and A. Lopez. 2017. Diagrammatic semantics for digital circuits. In *CSL 2017 (LIPIcs)*, Vol. 82. Leibniz-Zentrum für Informatik, 24:1–24:16.
- [20] E. Jeandel, S. Perdrix, and R. Vilmart. 2018. A complete axiomatisation of the ZX-calculus for Clifford+T quantum mechanics. In *LICS 2018*. ACM.
- [21] P. Katis, N. Sabadini, and R.F.C. Walters. 1997. *Span(Graph): An algebra of transition systems*. In *AMAST 1997 (LNCS)*, Vol. 1349. Springer, 322–336.
- [22] A. Kissinger and V. Zamdzhiev. 2015. Quantomatic: A proof assistant for diagrammatic reasoning. In *CADE 2015 (LNCS)*, Vol. 9195. Springer, 326–336.
- [23] S. Lack and P. Sobociński. 2005. Adhesive and quasiadhesive categories. *Theoretical Informatics and Applications* 39, 3 (2005), 511–546.
- [24] D. Marsden and F. Genovese. 2017. Custom hypergraph categories via generalized relations. In *CALCO 2017 (LIPIcs)*, Vol. 72. Leibniz-Zentrum für Informatik, 17:1–17:16.
- [25] K. F. Ng and Q. Wang. 2017. A universal completion of the ZX-calculus. *CoRR* abs/1706.09877 (2017).
- [26] D. Pavlovic. 2013. Monoidal computer I: Basic computability by string diagrams. *Information and Computation* 226 (2013), 94–116.
- [27] M. Sadrzadeh, S. Clark, and B. Coecke. 2013. The Frobenius anatomy of word meanings I: subject and object relative pronouns. *Journal of Logic and Computation* 23, 6 (2013), 1293–1317.
- [28] P. Selinger. 2011. A survey of graphical languages for monoidal categories. *Springer Lecture Notes in Physics* 13, 813 (2011), 289–355.
- [29] D. I. Spivak and J. Tan. 2017. Nesting of dynamical systems and mode-dependent networks. *Complex Networks* 5, 3 (2017), 389–408.
- [30] F. Zanasi. 2015. *Interacting Hopf Algebras: The Theory of Linear Systems*. Ph.D. Dissertation. Ecole Normale Supérieure de Lyon.
- [31] F. Zanasi. 2016. The algebra of partial equivalence relations. In *MFPS 2016 (ENTCS)*, Vol. 325. Elsevier, 313–333.
- [32] F. Zanasi. 2017. Rewriting in free hypergraph categories. In *GAM@ETAPS 2017 (EPTCS)*, Vol. 263. Open Publishing Association, 16–30.