Mathematical Structures in Computer Science (2022), **32**, pp. 829–869
doi:10.1017/S0960129522000123

CAMBRIDGE UNIVERSITY PRESS

PAPER

String diagram rewrite theory III: Confluence with and without Frobenius

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

[1]University of Pisa, Pisa, Italy, [2]University of Oxford, Oxford, UK, [3]Tallinn University of Technology, Tallinn, Estonia, [4]University College London, London, UK

*Corresponding author. Email: fabio.gadducci@unipi.it

(Received 3 May 2021; revised 8 April 2022; accepted 15 April 2022; first published online 13 June 2022)

Abstract

In this paper, we address the problem of proving confluence for string diagram rewriting, which was previously shown to be characterised combinatorially as double-pushout rewriting with interfaces (DPOI) on (labelled) hypergraphs. For standard DPO rewriting without interfaces, confluence for terminating rewriting systems is, in general, undecidable. Nevertheless, we show here that confluence for DPOI, and hence string diagram rewriting, *is* decidable. We apply this result to give effective procedures for deciding local confluence of symmetric monoidal theories with and without Frobenius structure by critical pair analysis. For the latter, we introduce the new notion of path joinability for critical pairs, which enables finitely many joins of a critical pair to be lifted to an arbitrary context in spite of the strong non-local constraints placed on rewriting in a generic symmetric monoidal theory.

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

1. Introduction

This is the final instalment in a series of three papers developing the rewriting theory of string diagrams. String diagrams are a practical and visually intuitive language for presenting compositions of morphisms in a symmetric monoidal category. These are particularly useful for expressing *symmetric monoidal theories* (SMTs), which enable one to present symmetric monoidal categories by generators and relations, strictly generalising algebraic theories. In Part I (Bonchi et al. 2022), we showed that when an SMT contains a Frobenius algebra, string diagrams are in one-to-one correspondence with (labelled) hypergraphs, and equational reasoning corresponds precisely to hypergraph rewriting. In Part II (Bonchi et al. 2020), we extended this representation to all SMTs, at the expense of requiring certain restrictions on which hypergraphs and hypergraph rewrites are allowed.

In this paper, we address one of the pillars of rewriting theory: The question of *confluence* for string diagram rewriting systems. For *term* rewriting, both confluence (Bauer and Otto 1984) and termination (Huet and Lankford 1978) are, in general, undecidable. However, for term rewriting systems known to be terminating, confluence is decidable. The key, celebrated property observed by Knuth and Bendix (1970) is that a terminating system is confluent exactly when *all its critical pairs are joinable*.

Since diagrams can be represented combinatorially as hypergraphs, it stands to reason that we should turn to the literature on *graph rewriting theory* to find answers about confluence. Here,

unfortunately, the status of confluence is less certain because established properties of critical pair analysis fail: Plump (1993), working in the framework of the double-pushout (DPO) graph rewriting (Ehrig and Kreowski 1976), showed that joinability of critical pairs *does not* entail confluence, and even worse: that confluence of terminating DPO rewriting systems is, in general, undecidable.

Thankfully, in the case of string diagrams, compositionality comes to the rescue. Unlike the graphs considered by Plump, string diagrams have a natural notion of an interface, namely the inputs and outputs of the diagram that represent the domain and codomain of a morphism in a symmetric monoidal category. Consequently, the appropriate notion of rewriting for string diagrams should preserve that interface. This is why in the first two parts of this series we have formalised string diagram rewriting using *DPO with interfaces* (DPOI).

The idea of performing rewrites that respect an interface is not new and has emerged in several research threads, including rewriting with borrowed contexts (Ehrig and König 2004), encodings of process calculi (Bonchi et al. 2009; Gadducci 2007) and connecting DPO rewriting systems with computads in cospans categories (Gadducci and Heckel 1998; Sassone and Sobociński 2005). Our key observation is that for DPOI rewriting, the Knuth-Bendix property is saved: Confluence of a terminating DPOI system can be decided by checking whether its critical pairs are joinable.

In fact, this result is more general than our particular encoding of string diagrams as hypergraphs: Under some mild assumptions related to the computability of individual rewriting steps, our result holds for DPOI rewriting in generic adhesive categories (Lack and Sobociński 2005).

Our results do not falsify Plump's: In DPOI rewriting, one rewrites morphisms $G \leftarrow J$, thought of as a graph (or graph-like object) G with a fixed interface J. The latter allows to consider G in a larger context, where J acts as the "glue" between G and its context. This is analogous to how variables allow a single term to apply to a variety of contexts via substitution. In the light of our analysis, Plump's result states that it is undecidable to check whether rewriting is confluent for all morphisms $G \leftarrow 0$, that is, objects with an empty interface. Intuitively, the failure of Knuth-Bendix for such morphisms is due to the loss of expressive power of critical pairs, if deprived of an interface.

This reveals an attractive analogy with term rewriting: Morphisms $G \leftarrow 0$ – representing graphs that can be only trivially attached to other graphs, since they have an empty interface – are akin to *ground terms*, that cannot be extended since they have no variables. The property that Plump showed to be undecidable should be compared to *ground confluence* for term rewriting (Padawitz 1980), that is, confluence with respect to all ground terms. And in fact, this property is undecidable for terminating term rewriting systems (Kapur et al. 1990). Summarising, for both term and DPOI rewriting, confluence of terminating rewriting systems is decidable, while ground confluence is not.

	Terminating term rewriting system	Terminating DPO system
Ground confluence	undecidable (Kapur et al. 1990)	undecidable (Plump 1993)
Confluence	decidable (Knuth and Bendix 1970)	decidable (this paper)

We can apply this result about confluence for DPOI rewriting to string diagrams. This problem is known to be particularly challenging, for example a directed form of the Yang-Baxter equation generates infinitely many critical pairs (Lafont 2003; Mimram 2014).

We show that this issue can be avoided by using DPOI rewriting and that confluence is decidable in many cases by checking only finitely many critical pairs. The generic result for DPOI rewriting applies essentially verbatim for SMTs containing Frobenius structure that we considered in Part I.

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

For generic SMTs that do not necessarily have Frobenius structure, the story is a bit more nuanced. It was shown in Part II that in order to obtain a sound rewriting theory for generic SMTs and avoid introducing directed cycles in diagrams, one should consider *convex* DPOI rewriting. As we foreshadowed in Part II, this can cause problems for confluence, as convex rewrites can sometimes have unexpected non-local effects. Namely, the validity of a convex match depends on the non-existence of paths from outputs to inputs in the image of a pattern graph in the target. Hence, rule applications that create paths can break convexity of matches elsewhere.

In this paper, we provide two solutions to this problem. The first solution solves the problem by putting a strong condition on the rewriting systems called *left-connectedness*. This condition essentially requires the left-hand sides of all rules to take a form that guarantees that *any* rewrite is already a convex rewrite; hence, it gives a very simple way to sidestep the technical challenges of convex rewriting. Many interesting rewriting systems arising from SMTs (e.g., Fiore and Campos 2013; Ghica 2013; Lafont 2003), including the aforementioned Yang-Baxter rule, enjoy this property. Amongst these is the rewriting system for *non-commutative bimonoids* that was shown to be terminating in Part II. In this paper, we will apply our technique to show that it is also confluent.

The second technique we provide is the first, to our knowledge, necessary and sufficient condition for checking local confluence of generic SMTs with finitely many critical pairs (although necessity comes with a caveat discussed at the end of Section 5.5). The key point for such systems is that the presence of paths from outputs to inputs in the context of a critical pair can affect its joinability, so we formally adjoin certain additional generators to a theory, which enable us to check not only the critical pairs themselves, but *path extensions* of the critical pairs, which account for these troublesome paths. Perhaps surprisingly, it is sufficient to check only finitely many of these to guarantee that a critical pair is joinable in *any* context. We apply this technique to show confluence of a simple SMT that is not left-connected.

Synopsis. The paper has the following structure. Section 2 recalls the basic notions concerning DPO rewriting for graphs with interfaces (DPOI) and PROP rewriting. Section 3 presents the main technical result, namely, local confluence for DPOI rewriting. Thanks to the correspondence results established in the previous papers of the series, this is exploited to prove local confluence for PROP rewriting with a Frobenius structure in Section 4 and for two different proposals of PROP rewriting without Frobenius in Section 5. Section 6 provides two case studies to support the relevance of our results, while Section 7 wraps up the paper with some final considerations.

Much of the content of this article is based on a paper published in the proceedings of ESOP 2017 (Bonchi et al. 2017). In addition to updating the paper, extending with more examples and explanation, and adapting for consistency with the previous two *String Diagram Rewrite Theory* papers, this version goes beyond the conference paper in two directions. First, it makes precise the distinction between pre-critical and critical pairs, providing an equivalent of the parallel independence theorem for DPOI rewriting (see Proposition 16). Second, and more substantial, the technique for proving local confluence for generic convex rewriting systems using formal path extensions (see Section 5.5) is completely new. This technique is put to work on a new case study (see Section 6.2).

Related work. Confluence is a classical topic for both term and graph rewriting, and it has been studied for quite some time. The key observation is always the same: Identifying a set of rewrite instances whose check could ensure the Knuth-Bendix property. Classically, this is the set of critical pairs. For DPO rewriting, despite the undecidability result recalled before, local confluence has been shown to hold with respect to a stronger notion of joinability for critical pairs (Plump 1993), and confluence is decidable whenever all critical pairs satisfy a syntactic condition, coverability (Plump 2010). More recently, we mention the work on confluence up-to garbage, whose intuition is to check if the rewriting system is confluent on a subclass of graphs that are of interest (Campbell and Plump 2020). And, on a similar note, the work on confluence for DPO with applications conditions (Ehrig et al. 2010) seems also relevant for our investigation. Indeed, both proposals are reminiscent of our restriction to monogamous acyclic hypergraphs and convex

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

rewriting, as introduced in Section 5, and establishing a precise correspondence is left for future work. Despite the introduction of interfaces, our approach to critical pairs is rather classical. A different proposal concerns *initial conflicts*, a restricted class of critical pairs that still guarantees the Knuth-Bendix property (Lambers et al. 2020): Also pursuing the adaptation of this notion in the context of DPOI is left for future work. Instead, our same notion of confluence has been studied in Sander Bruggink et al. (2011) in the setting of Milner's reactive systems. By instantiating Proposition 22 in Sander Bruggink et al. (2011) to the category of input-linear cospans (of hypergraphs) and by using the results relating borrowed context DPO rewriting with reactive systems over cospans in Sobociński (2004), one obtains a variant of our Theorem 2. One restriction of that approach is that the matches are required to be mono, which rules out our applications to SMTs.

2. Background

Notation 1. The composition of arrows $f : a \to b$, $g : b \to c$ in a category $\mathbb{C}$ is written as $f ; g$. For $\mathbb{C}$ symmetric monoidal, $\oplus$ is its monoidal product and $\sigma_{a,b} : a \oplus b \to b \oplus a$ is the symmetry for objects $a, b \in \mathbb{C}$.

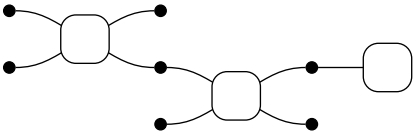
2.1 DPO rewriting

2.1.1 Adhesive categories and (typed) hypergraphs.

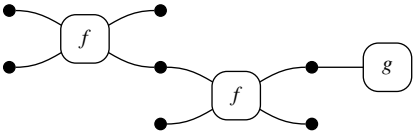
In order not to restrict ourselves to any concrete model of graphs, we work with adhesive categories (Lack and Sobociński 2005). Adhesive categories are relevant because they have well-behaved pushouts along monomorphisms, and for this reason, they are convenient as ambient categories for DPO rewriting.

An important example is the category of finite directed hypergraphs **Hyp**. An object G of **Hyp** is a hypergraph with a finite set of *nodes* $G_\star$ and for each $k, l \in \mathbb{N}$ a finite set of *hyperedges* $G_{k,l}$ with k (ordered) sources and l (ordered) targets, that is, for each $0 \leq i < k$ there is the i^{th} source map $s_i : G_{k,l} \to G_\star$ and for each $0 \leq j < l$ the j^{th} target map $t_j : G_{k,l} \to G_\star$. The arrows of **Hyp** are homomorphisms: functions $G_\star \to H_\star$ such that for each k, l, $G_{k,l} \to H_{k,l}$ they respect the source and target maps in the obvious way. The seasoned reader will recognise **Hyp** as a presheaf topos, and as such, it is adhesive (Lack and Sobociński 2005).

We shall visualise hypergraphs as follows: • is a node and  is a hyperedge, with ordered tentacles attached to the left boundary linking to sources and those on the right linking to targets

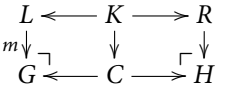


A signature Σ consists of a set of *generators* $o : n \to m$ with arity n and coarity m where $m, n \in \mathbb{N}$. Any signature Σ can be considered as a hypergraph G_Σ with a single node, in the obvious way. We can then express Σ*-labelled hypergraphs* (briefly, Σ-hypergraphs) as the objects of the slice category $\mathbf{Hyp} \downarrow G_\Sigma$, denoted by $\mathbf{Hyp}_\Sigma$, which is adhesive, since adhesive categories are closed under slice (Lack and Sobociński 2005). Σ-hypergraphs are drawn by labelling hyperedges with generators in Σ



https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

2.1.2 DPO rewriting.
We recall the DPO approach (Ehrig and Kreowski 1976) to rewriting in an adhesive category $\mathbb{C}$. A *DPO rule* is a span $L \leftarrow K \rightarrow R$ in $\mathbb{C}$. A *DPO system* $\mathcal{R}$ is a finite set of DPO rules. Given objects G and H in $\mathbb{C}$, we say that G *rewrites* into H – notation $G \rightsquigarrow_{\mathcal{R}} H$– if there exist $L \leftarrow K \rightarrow R$ in $\mathcal{R}$, object C and morphisms such that the squares below are pushouts



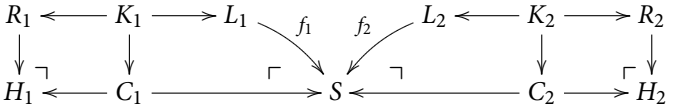
A *derivation* from G into H is a sequence of such rewriting steps. The arrow $m : L \rightarrow G$ is called a *match* of L in G. A rule $L \leftarrow K \rightarrow R$ is said to be *left-linear* if the morphism $K \rightarrow L$ is mono. In this case, the matching m fully determines the graphs C and H; that is, for a fixed rule and a matching, there is a unique H such that $G \rightsquigarrow_{\mathcal{R}} H$, if it exists. Here, by unique, we mean unique up-to isomorphism. More generally, the rewriting steps will always be *up-to iso*: in a step $G \rightsquigarrow_{\mathcal{R}} H$, G and H should not be thought of as single graphs but rather as equivalence classes of isomorphic graphs.

2.1.2.1 Undecidability of confluence. In DPO rewriting, the confluence of terminating systems is not decidable, even if we restrict to left-linear rules.

Theorem 1 (Plump 1993)**.** *Confluence of terminating DPO systems over* $\mathbf{Hyp}_{\Sigma}$ *is undecidable.*

Indeed, critical pair analysis for traditional DPO systems fails: for terminating DPO systems, joinability of critical pairs does not necessarily imply confluence.

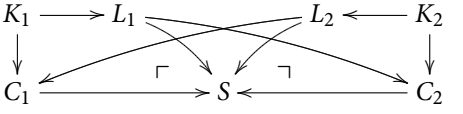
Definition 2 (Pre-critical pair and joinability)**.** *Let* $\mathcal{R}$ *be a DPO system with rules* $L_1 \leftarrow K_1 \rightarrow R_1$ *and* $L_2 \leftarrow K_2 \rightarrow R_2$*. Consider two derivations with common source* S



We say that $H_1 \leftsquigarrow S \rightsquigarrow H_2$ *is a* pre-critical pair *if* $[f_1, f_2] : L_1 + L_2 \rightarrow S$ *is epi; it is* joinable *if there exists* W *such that* $H_1 \rightsquigarrow^* W \ {}^*\!\!\leftsquigarrow H_2$.

We use the standard notation $\rightsquigarrow^*$ for defining a possibly empty sequence of rewriting steps. Intuitively, in a pre-critical pair S should not be bigger than $L_1 + L_2$. In a critical pair, L_1 and L_2 must overlap in S, so that the two rewriting steps do not form a *parallel pair*.

Definition 3 (Parallel pair)**.** *Let* $\mathcal{R}$ *be a DPO system with rules* $L_1 \leftarrow K_1 \rightarrow R_1$ *and* $L_2 \leftarrow K_2 \rightarrow R_2$*. Consider two derivations with common source* S *as in Definition 2. We say that* $H_1 \leftsquigarrow S \rightsquigarrow H_2$ *is a* parallel pair *if there exist* $g_1 : L_1 \rightarrow C_2$ *and* $g_2 : L_2 \rightarrow C_1$ *making the diagram below commute*



The key result for us is that parallel pairs are always joinable (see e.g., Corradini 2016), and a pre-critical pair is thus *critical* if it is not a parallel one. However, for the purposes of this paper, this restriction is immaterial, and we will mostly stick to pre-critical pairs in our results, as proofs are less tedious. For the sake of succinctness, most of the examples will instead display only the

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

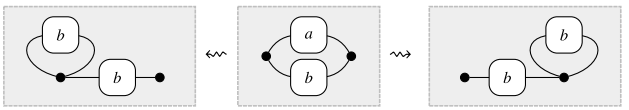
critical pairs. For a pre-critical pair which is also a parallel pair, see for instance the first picture of Section 6.1.

A notable feature of DPO rewriting is that, unlike in the case of term rewriting, joinability of all critical pairs is not enough to guarantee confluence, even for a terminating rewriting system.

Example 4 (Plump 1993)**.** Consider a DPO system $\mathcal{R}$ consisting of the following two rules, where we labelled nodes with numbers in order to make the graph morphisms explicit

Amongst the several pre-critical pairs, only the following two have non-trivial overlap

Both are obviously joinable. However, $\mathcal{R}$ is not confluent, as witnessed by the following



However, this "bug" in DPO rewriting can be fixed by considering graphs with interfaces, and DPO rules that respect the interface.

2.2 DPO rewriting with interfaces.

Morphisms $G \leftarrow J$ will play a special role in our exposition. When $\mathbb{C}$ is $\mathbf{Hyp}_\Sigma$, we will call them *(hyper)graphs with interface*. The intuition is that G is a hypergraph and J is an interface that allows G to be "glued" to a context. Note however that such morphisms are not necessarily mono, even if this will be the case in most of our examples.

Given $G \leftarrow J$ and $H \leftarrow J$ in $\mathbb{C}$, *G rewrites into H with interface J* –notation $(G \leftarrow J) \rightsquigarrow_{\mathcal{R}} (H \leftarrow J)$– if there exist rule $L \leftarrow K \rightarrow R$ in $\mathcal{R}$, object C, and morphisms such that the diagram below commutes and the squares are pushouts

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

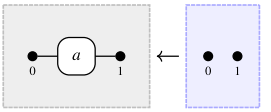
Hence, the interface J is preserved by individual rewriting steps.

When $\mathbb{C}$ has an initial object 0 (for instance, in $\mathbf{Hyp}_\Sigma$ 0 is the empty hypergraph), ordinary DPO rewriting can be considered as a special case, by taking J to be 0.

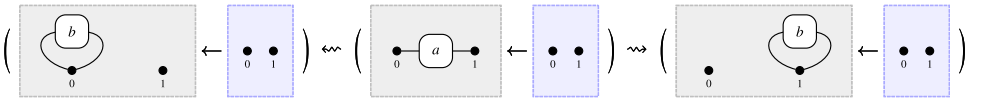
Like for traditional DPO, rewriting steps are modulo isomorphism: $G_1 \leftarrow J : f_1$ and $G_2 \leftarrow J : f_2$ are isomorphic if there is an isomorphism $\varphi : G_1 \rightarrow G_2$ with $f_1 ; \varphi = f_2$.

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

Example 5. Consider the system $\mathcal{R}$ from Example 4 and the graph with interface below



It can be rewritten in two different ways



(1)

which, unlike as in Example 4, results in two distinct hypergraphs with interface. Notably, the interface $\{0, 1\}$ maintains the distinct identities of the two nodes initially connected to the hyperedge labelled by a, even after that hyperedge is removed. Notice that if (1) were considered as a critical pair, it would not be joinable. Hence, the counterexample of Plump (1993) (Example 4) would not work. This is the starting observation for our formulation of critical pair analysis: in Section 3 we will introduce pre-critical pairs for rewriting with interfaces and we will show that, as in term rewriting, joinability of pre-critical pairs entails confluence.

2.3 PROP rewriting

2.3.1 SMTs and PROPs.

A uniform way to express an algebraic structure within a symmetric monoidal category is with a *symmetric monoidal theory* (SMT). A (one-sorted) SMT is a pair $(\Sigma, \mathcal{E})$ where Σ is a *signature* defined as in Section 2.1. The set of Σ-terms is obtained by combining generators in Σ, the *unit* $\mathrm{id} : 1 \to 1$ and the *symmetry* $\sigma_{1,1} : 2 \to 2$ with ; and $\oplus$. That means, given Σ-terms $t : k \to l$, $u : l \to m$, $v : m \to n$, one constructs new Σ-terms $t ; u : k \to m$ and $t \oplus v : k + m \to l + n$. The set $\mathcal{E}$ of *equations* contains pairs (t, t') of Σ-terms, with the requirement that t and t' have the same arity and coarity.

Just as ordinary (cartesian) algebraic theories have a categorical rendition as Lawvere categories (Hyland and Power 2007), the corresponding linear notion (i.e., in the sense that variables can neither be copied, nor discarded) for SMTs is a PROP (Mac Lane 1965) (**pro**duct and **p**ermutation category). A PROP is a symmetric strict monoidal category with objects the natural numbers, where $\oplus$ on objects is addition. Morphisms between PROPs are identity-on-objects strict symmetric monoidal functors. PROPs and their morphisms form a category PROP. Any SMT $(\Sigma, \mathcal{E})$ freely generates a PROP by letting the arrows $n \to m$ be the Σ-terms $n \to m$ modulo the laws of symmetric monoidal categories and the (smallest congruence containing the) equations $t = t'$ for any $(t, t') \in \mathcal{E}$.

We write $\mathbf{S}_\Sigma$ to denote the PROP freely generated by $(\Sigma, \varnothing)$. There is a graphical representation of the arrows of $\mathbf{S}_\Sigma$ as string diagrams, which we now sketch, referring to Selinger (2011) for the details. A Σ-term $n \to m$ is pictured as a box with n ports on the left and m ports on the right, which are ordered and referred to with top-down enumerations $1, \ldots, n$ and $1, \ldots, m$. Compositions via ; and $\oplus$ are drawn, respectively, as horizontal and vertical juxtaposition, that means, $t ; s$ is drawn  and $t \oplus s$ is drawn  . There are specific diagrams for the Σ-terms responsible for the symmetries: these are $\mathrm{id}_1 : 1 \to 1$, represented as  , the symmetry $\sigma_{1,1} : 1 + 1 \to 1 + 1$, represented as  , and the unit object for $\oplus$, that is, $\mathrm{id}_0 : 0 \to 0$, whose representation is an empty diagram  . Graphical representation for arbitrary identities id_n and

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

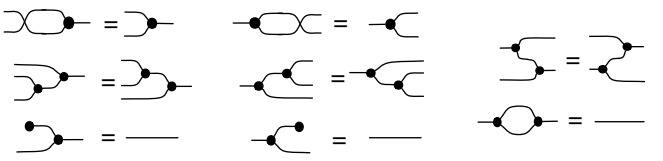
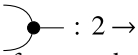


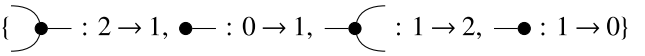
Figure 1. The equations $\mathcal{E}_{\mathbf{Frob}}$ of special Frobenius monoids.

symmetries $\sigma_{n,m}$ are generated using the pasting rules for ; and $\oplus$. It will be sometimes convenient to represent id_n with the shorthand diagram $\boxed{n}$ and, similarly, $t : n \to m$ with $\overset{n}{\boxed{t}}\overset{m}{}$.

Example 6.

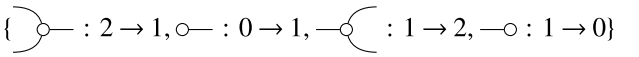
(a) A basic example is the theory $(\Sigma_{\mathbf{Mon}}, \mathcal{E}_{\mathbf{Mon}})$ of *commutative monoids*. The signature $\Sigma_{\mathbf{Mon}}$ contains two generators: *multiplication* – which we depict  – and *unit*, represented as  . Equations in $\mathcal{E}_{\mathbf{Mon}}$ are given in the leftmost column of Figure 1: they assert commutativity, associativity and unitality.

(b) An SMT that plays a key role in our exposition is the theory $(\Sigma_{\mathbf{Frob}}, \mathcal{E}_{\mathbf{Frob}})$ of *special Frobenius monoids*. The signature $\Sigma_{\mathbf{Frob}}$ is as follows and $\mathcal{E}_{\mathbf{Frob}}$ is depicted in Figure 1

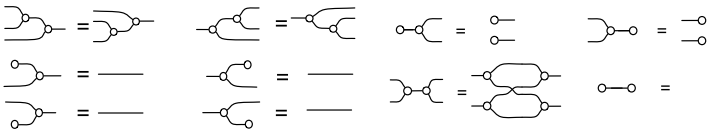


E_F includes the theory of commutative monoids in the leftmost column. Dually, the equations in the middle column assert that  and  form a cocommutative comonoid. Finally, the two rightmost equations describe an interaction between these two structures. We call **Frob** the PROP freely generated by $(\Sigma_{\mathbf{Frob}}, \mathcal{E}_{\mathbf{Frob}})$.

(c) The theory of *non-commutative bimonoids* has signature $\Sigma_{\mathbf{NBiM}}$



and the following equations $\mathcal{E}_{\mathbf{NBiM}}$



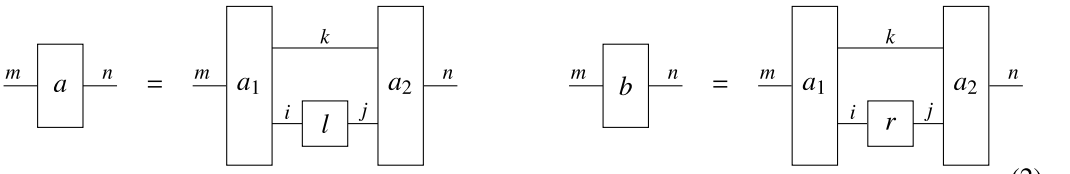
We call **NB** the PROP freely generated from $(\Sigma_{\mathbf{NBiM}}, \mathcal{E}_{\mathbf{NBiM}})$. In Bonchi et al. (2020) we showed that the rewriting system that is obtained by orienting the equalities from left to right terminates. In this paper, we will show that is also confluent. For this, it will be convenient to use μ, η, ν, ϵ, respectively, to refer to the generators in $\Sigma_{\mathbf{NBiM}}$.

2.3.2 Rewriting in a PROP.

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.

Definition 8. *A* rewriting system $\mathcal{R}$ *in a PROP* $\mathbb{A}$ *consists of a set of* rewriting rules*, that is, pairs* $\langle l, r \rangle$ *of arrows* $l, r : i \to j$ *in* $\mathbb{A}$ *with the same arities and coarities. Given* $a, b : m \to n$ *in* $\mathbb{A}$*, a rewrites into b via* $\mathcal{R}$*, written* $a \Rightarrow_{\mathcal{R}} b$*, if they are decomposable as follows, for some rule* $\langle l, r \rangle \in \mathcal{R}$

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

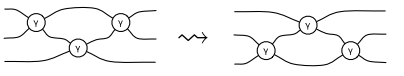


(2)

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

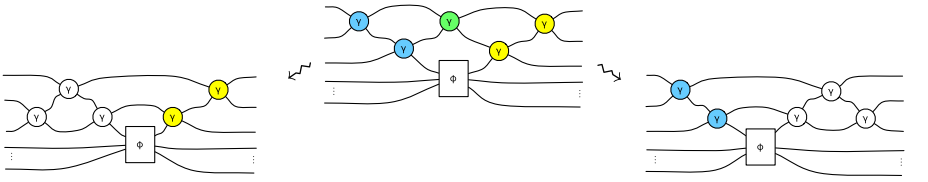
The following well-known example illustrates the subtlety of critical pair analysis when rewriting in monoidal categories.

Example 9 (From Lafont 2003, see also Mimram 2014)**.** Fix $\Sigma = \{\gamma : 2 \to 2\}$ and consider the rewriting system on $\mathbf{S}_\Sigma$ consisting of the following rule



(3)

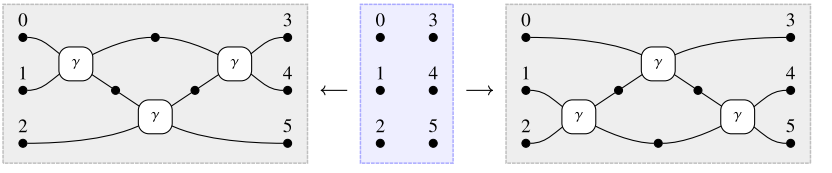
A critical pair analysis yields an infinite number of critical pairs. Indeed, as shown in Lafont (2003); Mimram (2014), any diagram $\phi : 1 + m \to 1 + n$ that does not decompose non-trivially into $\phi = \mu + \nu$ for some μ, ν yields a critical pair



in which clearly there are two embeddings of the left-hand side of (3) (depicted in blue and yellow, respectively, in a colour version of the paper) with an overlap (in green).

In Mimram (2010) this problem was solved by adding duals to monoidal categories. In Section 4, we will show another solution based on Bonchi et al. (2020): a translation from PROPs to DPO rewriting with interfaces. The example below anticipates this encoding. It will be useful as a running example for the next section, which is devoted to critical pair analysis and confluence in DPO rewriting with interfaces.

Example 10. Treating the rewriting system of Example 9 as DPO system over $\mathbf{Hyp}_\Sigma$ with $\gamma : 2 \to 2 \in \Sigma$ yields the following DPO rule

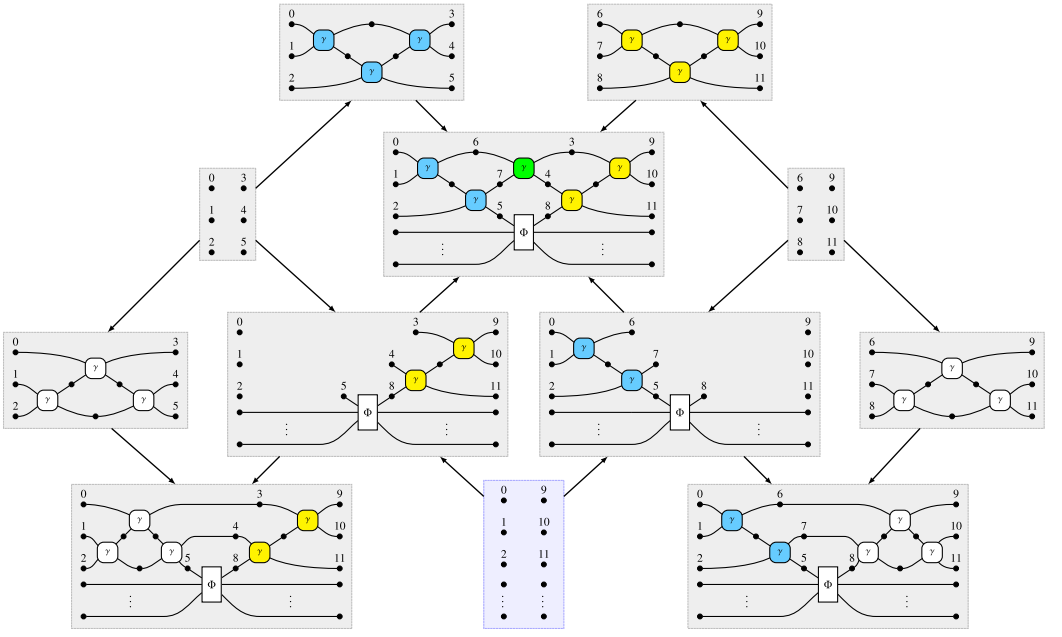


The formal correspondence between PROPs and DPO rewriting with interfaces will be explained in Section 4. For the time being, the reader can observe the similarities between the left-hand side of (3) and the left-hand side of the above rule: each γ in (3) corresponds to an hyperedge (labelled with γ) in the hypergraph above; moreover, each wires in (3) corresponds to a node above; finally,

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

dangling wires in (3) are exactly the numbered nodes. A similar correspondence holds for the right-hand side, while the interface of the DPO rule, depicted in light blue, just collects all the numbered nodes.

Below, we give a DPO derivation with interface (in light blue), corresponding to a critical pair from the family identified in Example 9



3. Confluence for DPO Rewriting with Interfaces

Differently from Definition 2, the interface of the pre-critical pair plays a crucial role when considering the setting of DPO with interfaces.

Definition 11 (Pre-critical pair with interface). *Let $\mathcal{R}$ be a DPO system with rules $L_1 \leftarrow K_1 \rightarrow R_1$ and $L_2 \leftarrow K_2 \rightarrow R_2$. Consider two derivations with source $S \leftarrow J$*

$$\begin{array}{ccccccccc} R_1 & \leftarrow & K_1 & \rightarrow & L_1 & & L_2 & \leftarrow & K_2 & \rightarrow & R_2 \\ \downarrow & \urcorner & \downarrow & & & {\scriptstyle f_1}\searrow \quad \swarrow{\scriptstyle f_2} & & & \downarrow & \urcorner & \downarrow \\ H_1 & \leftarrow & C_1 & & & \urcorner \quad S \quad \urcorner & & & C_2 & & H_2 \\ & & & \searrow & & (\dagger) & & \swarrow & & & \\ & & & & & J & & & & & \end{array} \tag{4}$$

We say that $(H_1 \leftarrow J) \leftsquigarrow (S \leftarrow J) \rightsquigarrow (H_2 \leftarrow J)$ is a pre-critical pair *if $[f_1, f_2] : L_1 + L_2 \rightarrow S$ is epi and $(\dagger)$ is a pullback; it is* joinable *if there exists $W \leftarrow J$ such that $(H_1 \leftarrow J) \rightsquigarrow^* (W \leftarrow J) \ {}^*\!\!\leftsquigarrow (H_2 \leftarrow J)$.*

Definition 11 augments Definition 2 with the interface J, given by "intersecting" C_1 and C_2. Intuitively, J is the largest interface that allows both rewriting steps.

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

Example 12. Consider the pair of rewriting steps (1) in Example 5. This is a pre-critical pair: the reader can check that the interface is indeed a pullback, constructed as in (†). Observe moreover that this pair is *not* joinable. Should we consider rewriting without interfaces, that is, should J be the empty graph, the pair $(H_1 \leftarrow J) \leftsquigarrow (S \leftarrow J) \rightsquigarrow (H_2 \leftarrow J)$ would not be a pre-critical pair anymore.

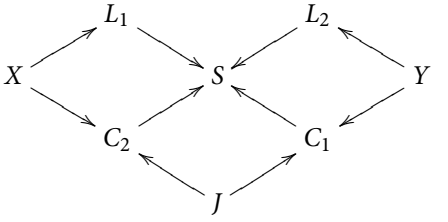
Plump's Example 4 shows that in ordinary DPO, joinability of pre-critical pairs does not imply confluence. Our Example 12 shows that the argument does not work for DPO with interfaces. Indeed, as we shall see in Theorem 2, in the presence of interfaces joinability suffices for confluence. To prove it, we assume the following property.

Assumption 13. *Our ambient category $\mathbb{C}$ is assumed (1) to possess an epi-mono factorisation system, (2) to have binary coproducts, pushouts and pullbacks, and (3) to be adhesive with (4) all the pushouts stable under pullbacks.*

The above conditions hold in any presheaf category. Additionally, they are closed under slice. It follows that $\mathbf{Hyp}_\Sigma$ is an example of such a category.

We could now mimic the definition of parallel pairs given in Definition 3. However, the existence of pullbacks in property (2) allows for a simpler characterisation.

Definition 14 (Parallel pair with interface)**.** *Let $\mathcal{R}$ be a DPO system with rules $L_1 \leftarrow K_1 \rightarrow R_1$ and $L_2 \leftarrow K_2 \rightarrow R_2$. Consider two derivations with common source $S \leftarrow J$ as in Definition 11 and the diagram below formed by pullbacks*



We say that $(H_1 \leftarrow J) \leftsquigarrow (S \leftarrow J) \rightsquigarrow (H_2 \leftarrow J)$ is a parallel pair *if $X \rightarrow L_1$ and $Y \rightarrow L_2$ are iso and $C_1 \rightarrow S$ and $C_2 \rightarrow S$ are mono.*

The definition is slightly stronger than the one for parallel independence for DPO rewriting without interfaces shown in Definition 3, even if it coincides whenever rules are left-linear (Corradini et al. 2018, Definition 5 and Proposition 1). However, this formulation is better suited for our notion of rewriting *with* interfaces, and indeed, it is easy to see that parallel pairs are joinable.

Before moving to the proof, though, we need a technical lemma, the following simple pushout decomposition result (aka "mixed decomposition" from Baldan et al. (2011)). The proof uses only stability of pushouts under pullbacks, which is encompassed by our Assumption 13.

Lemma 15. *Suppose that in the diagram below m is mono, (†) + (‡) is a pushout, and (‡) is a pullback. Then both (†) and (‡) are pushouts.*

$$\begin{array}{ccccc} K & \longrightarrow & C' & \longrightarrow & C \\ \downarrow & (\dagger) & \downarrow & (\ddagger) & \downarrow \\ L & \longrightarrow & G' & \xrightarrow[m]{} & G \end{array} \tag{5}$$

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

Proposition 16. *Let $\mathcal{R}$ be a DPO system with rules $L_1 \leftarrow K_1 \rightarrow R_1$ and $L_2 \leftarrow K_2 \rightarrow R_2$. Consider two derivations with common source $S \leftarrow J$ as in Definition 11. If $(H_1 \leftarrow J) \leftsquigarrow (S \leftarrow J) \rightsquigarrow (H_2 \leftarrow J)$ is a parallel pair then it is joinable.*

Proof. Let us assume that $X = L_1$ and $Y = L_2$, so that the arrows $L_1 \rightarrow C_2$ and $L_2 \rightarrow C_1$ are obviously defined, and the conditions of Definition 3 satisfied. The existence of a parallel pair for graph without interfaces is a standard result, see for example the survey (Habel et al. 2001).

Note also that $C_i \rightarrow S$ are both mono, even if $K_i \rightarrow L_i$ might not be so. Hence, we can retrace the steps of the classical proof presented in Ehrig (1978, Section 9.7). So, consider the two diagrams below

$$\begin{array}{ccccc} & & K_2 & \longrightarrow & L_2 \\ & & \downarrow & (3) & \downarrow \\ K_1 & \longrightarrow & J & \longrightarrow & C_1 \\ \downarrow & (2) & \downarrow & (1) & \downarrow \\ L_1 & \longrightarrow & C_2 & \longrightarrow & S \end{array} \qquad \begin{array}{ccccc} & & K_2 & \longrightarrow & R_2 \\ & & \downarrow & (6) & \downarrow \\ K_1 & \longrightarrow & J & \longrightarrow & E_2 \\ \downarrow & (5) & \downarrow & (4) & \downarrow \\ R_1 & \longrightarrow & E_1 & \longrightarrow & W \end{array} \tag{6}$$

The square (1) is a pullback; hence, $K_i \rightarrow J$ are obtained by the universal property. By construction, the squares (2)+(1) and (3)+(1) are pushouts; hence, by the mixed decomposition lemma the squares (1), (2), and (3) are also pushouts. Squares (4), (5), and (6) are now obtained by pushout.

Consider then the diagrams below

$$\begin{array}{ccc} K_1 & \longrightarrow & R_1 \\ \downarrow & (5) & \downarrow \\ J & \longrightarrow & E_1 \\ \downarrow & (7) & \downarrow \\ C_1 & \longrightarrow & H_1 \end{array} \qquad \begin{array}{ccc} K_2 & \longrightarrow & R_2 \\ \downarrow & (6) & \downarrow \\ J & \longrightarrow & E_2 \\ \downarrow & (8) & \downarrow \\ C_2 & \longrightarrow & H_2 \end{array} \qquad \begin{array}{ccccc} L_2 & \longleftarrow & K_2 & \longrightarrow & R_2 \\ \downarrow & (3) & \downarrow & (6) & \downarrow \\ C_1 & \longleftarrow & J & \longrightarrow & E_2 \\ \downarrow & (7) & \downarrow & (4) & \downarrow \\ H_1 & \longleftarrow & E_1 & \longrightarrow & W \end{array} \tag{7}$$

Squares (5), (6), (5)+(7), and (6)+(8) are pushouts; hence, $E_i \rightarrow H_i$ are obtained by the universal property and also squares (7) and (8) are pushouts.

We then have all in place to obtain two derivations $(H_1 \leftarrow J) \rightsquigarrow (W \leftarrow J) \leftsquigarrow (H_2 \leftarrow J)$: The derivation $(H_1 \leftarrow J) \rightsquigarrow (W \leftarrow J)$ is depicted on the right in the picture above. □

As for the rewriting without interfaces, in our results we stick to pre-critical pairs, the distinction being immaterial, even if in the examples we usually show just the critical ones.

The following construction mimics (Ehrig et al. 2004). It allows us to restrict –or "clip"– a DPO rewriting step with match $f : L \rightarrow G$ to any subobject of G' through which f factors.

Construction 17 (One-step clipping)**.** Suppose we have a DPO rewriting step as below left, together with factorisation $L \rightarrow G' \xrightarrow{m} G$ where m is mono. As shown below right, we get C' by pulling back $G' \rightarrow G \leftarrow C$ and $K \rightarrow C'$ by the universal property

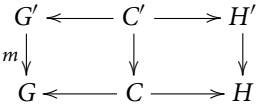
$$\begin{array}{ccccc} L & \longleftarrow & K & \longrightarrow & R \\ & \searrow & \downarrow & & \downarrow \\ G' & & & & \\ & \searrow^{m} & \downarrow & & \downarrow \\ & & G & \longleftarrow & C \longrightarrow H \end{array} \qquad \begin{array}{ccccc} L & \longleftarrow & K & \longrightarrow & R \\ & \searrow & \vdots & & \downarrow \\ G' & \longleftarrow & C' & & \\ & \searrow^{m} & \downarrow & \searrow & \downarrow \\ & & G & \longleftarrow & C \longrightarrow H \end{array}$$

By Lemma 15 the two leftmost squares are both pushouts. Next, H' is the pushout of $C' \leftarrow K \rightarrow R$ and $H' \rightarrow H$ follows from its universal property

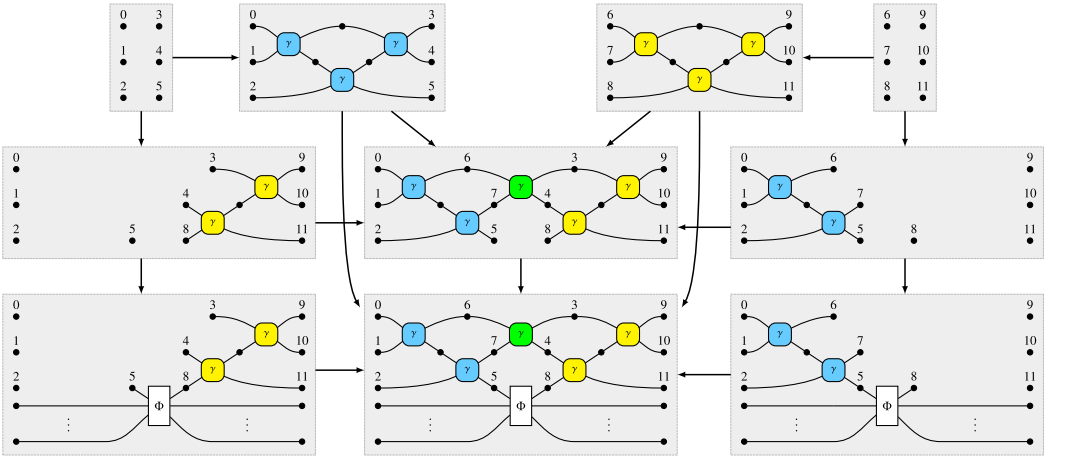
$$\begin{array}{ccccccc} L & \longleftarrow & K & \longrightarrow & R & & \\ & \searrow & \vdots & & \vdots & \searrow & \\ G' & \longleftarrow & C' & \longrightarrow & H' & & \\ & \searrow^{m} & \downarrow & \searrow & \downarrow & & \downarrow \\ & & G & \longleftarrow & C & \longrightarrow & H \end{array}$$

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

By pushout pasting also the bottom-rightmost square is a pushout. Finally, observe that $C' \to C$ is mono since it is the pullback of m along $C \to G$. This means that each of the two squares in diagram below is, as well as being a pushout, also a pullback, since each is a pushout along a mono in an adhesive category

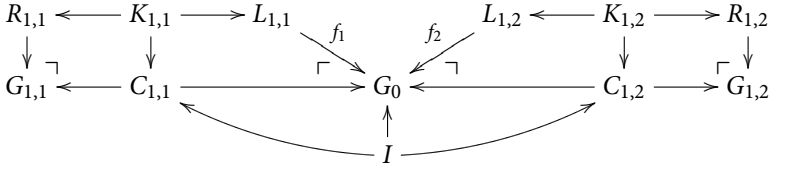


Example 18. We use the clipping construction to restrict pairs of derivations with common source into pre-critical pairs. For example, consider the two DPO rewriting rules illustrated in Example 10. We can factorise the two matches through their common image, and clip, as illustrated below

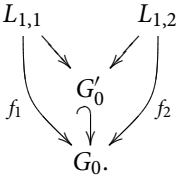


Note that the clipped derivations result with the two matches being jointly epi, which is one of the properties of a pre-critical pair. This generalises: given two rewriting steps with common source $(G_{1,1} \leftarrow I) \rightsquigarrow (G_0 \leftarrow I) \rightsquigarrow (G_{1,2} \leftarrow I)$, the next construction produces a pre-critical pair $(G'_{1,1} \leftarrow J') \rightsquigarrow (G_0{}' \leftarrow J') \rightsquigarrow (G'_{1,2} \leftarrow J')$ using clipping.

Construction 19 (Pre-critical pair extraction)**.** Start with two rewrites from $G_0 \leftarrow I$

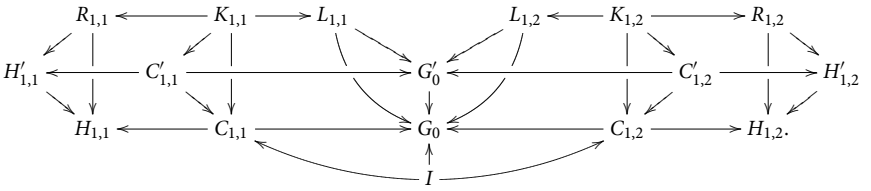


and factorise $[f_1, f_2] : L_{1,1} + L_{1,2} \to G_0$ to obtain

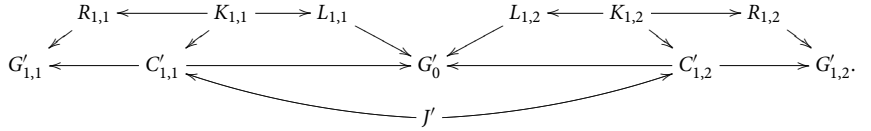


https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

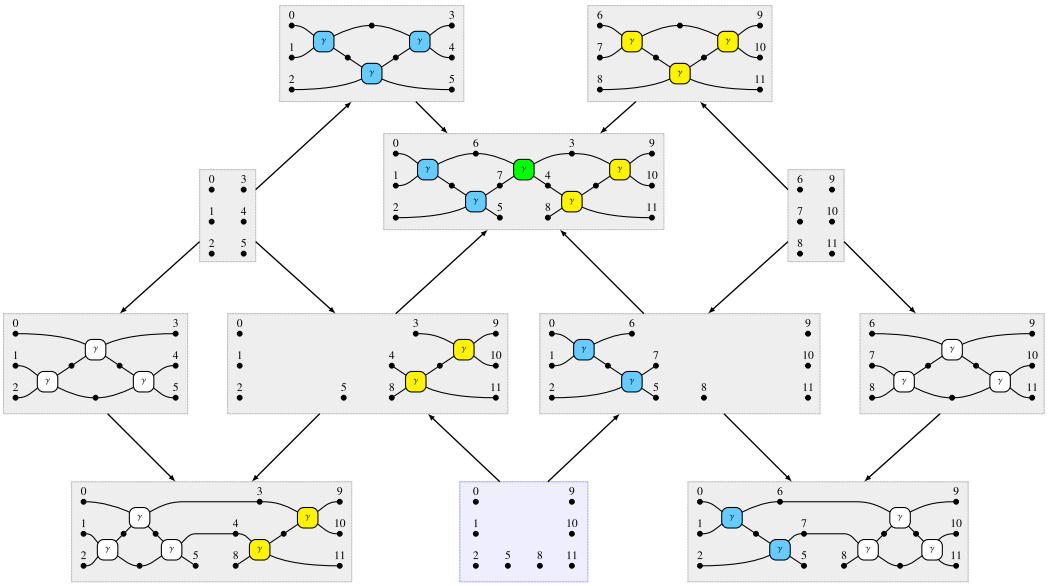
Next apply Construction 17 twice, obtaining



Finally, pull back $C'_{1,1} \to G'_0 \leftarrow C'_{1,2}$ to obtain the pre-critical pair



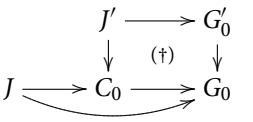
Example 20. We can now complete the pre-critical pair extraction process, commenced in Example 18, following the steps of Construction 19



Construction 19 means that we are able to extract a pre-critical pair from two rewriting steps with common source. If the pre-critical pair is joinable, we would then like to embed the joining derivations to the original context.

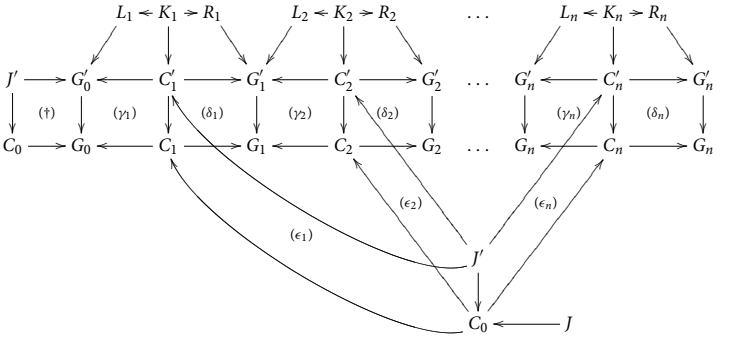
The following is a useful step in this direction. Assuming a mono $G'_0 \to G_0$, it allows us to extend a derivation from $G'_0 \leftarrow J'$ to a corresponding one from $G_0 \leftarrow J$, if we can obtain G_0 by glueing G'_0 and some context C_0 along J'. Stated more formally, we want the following diagram commute and (†) be a pushout

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press



(8)

Construction 21 (Embedding)**.** The extended derivation is constructed as in the commuting diagram below, where each square is a pushout diagram

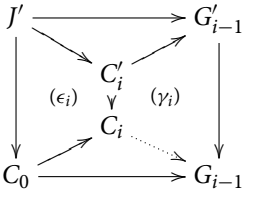


We shall now explain each of the components. The upper row of pushouts together with morphisms $J' \to C'_i$ witnesses the original derivation $(G'_0 \leftarrow J') \rightsquigarrow^* (G'_n \leftarrow J')$.

For $i = 1 \ldots n$, (ϵ_i) is formed as the pushout of $C_0 \leftarrow J' \to C'_i$ and (δ_i) as the pushout of $C_i \leftarrow C'_i \to G'_i$, as shown in the diagram below

$$\begin{array}{ccccc} J' & \longrightarrow & C'_i & \longrightarrow & G'_i \\ \downarrow & & \downarrow{\scriptstyle (\epsilon_i)} & & \downarrow{\scriptstyle (\delta_i)} \\ C_0 & \longrightarrow & C_i & \longrightarrow & G_i \end{array} \tag{9}$$

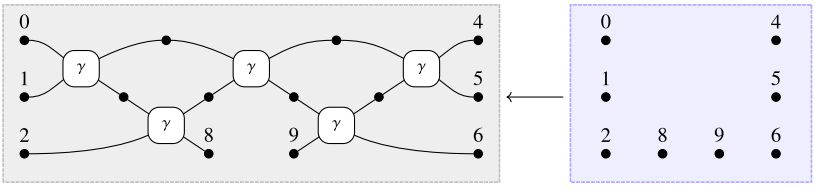
It remains to construct pushouts (γ_i), which is done in the following diagram



(10)

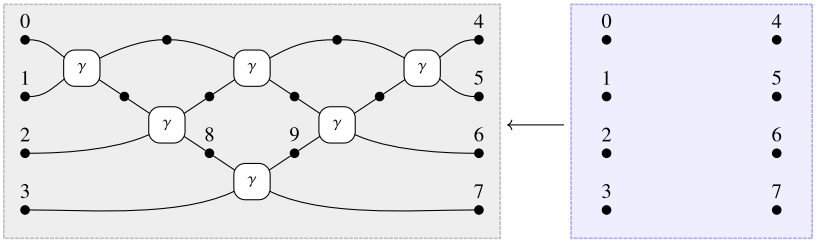
The exterior square in (10) is a pushout: for $i = 1$ it is (†) from (8), while for $i \geq 2$ it is obtained by composing (ϵ_{i-1}) and (δ_{i-1}) from (9). The universal property of (ϵ_i) yields the morphism $C_i \to G_{i-1}$. By pushout decomposition, the diagram (γ_i) is a pushout.

Example 22. In Example 20 we saw two derivations from

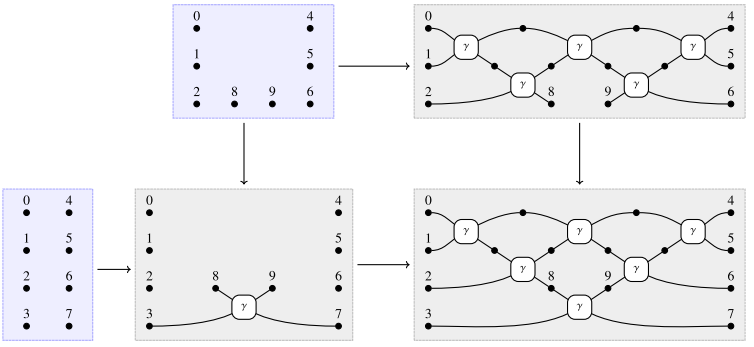


These can be extended to

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

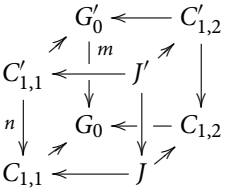


following the steps in Construction 21 because the square below is a pushout

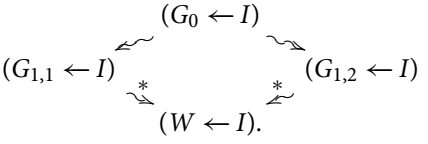


Constructions 19 and 21 are the main ingredients for showing the Knuth-Bendix property for DPOI. Before we prove it, we need one technical lemma from the theory of adhesive categories.

Lemma 23. *Consider the cube below, where the top and bottom faces are pullbacks, the rear faces are both pullbacks and pushouts, and m is mono. Then, the front faces are also pushouts.*



Theorem 2 (Local confluence)**.** *For a DPO system with interfaces, if all pre-critical pairs are joinable then rewriting is locally confluent: given* $(G_{1,1} \leftarrow I) \leftsquigarrow (G_0 \leftarrow I) \rightsquigarrow (G_{1,2} \leftarrow I)$*, there exists* $W \leftarrow I$ *such that*



Proof. Following the steps of Construction 19, we obtain a pre-critical pair

$$(G'_{1,1} \leftarrow J') \leftsquigarrow (G'_0 \leftarrow J') \rightsquigarrow (G'_{1,2} \leftarrow J')$$

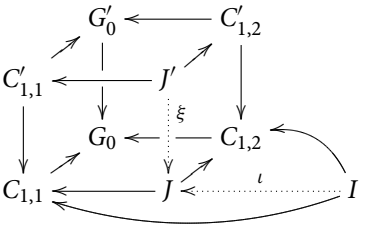
Because pre-critical pairs are by assumption joinable, we have derivations

$$(G'_{1,1} \leftarrow J') \rightsquigarrow^* (W' \xleftarrow{\beta'} J') \,{}^*\!\leftsquigarrow (G'_{1,2} \leftarrow J').$$

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

Suppose that the leftmost derivation requires n steps and the rightmost m. To keep the notation consistent with Construction 21, we fix notation $G'_{n,1} := W' =: G'_{m,2}$.

Now let J be the pullback object of $C_{1,1} \to G_0 \leftarrow C_{1,2}$. By the universal property, we obtain maps $\iota : I \to J$ and $\xi : J' \to J$



(11)

Recall by Construction 17 that the rear faces of (11) are both pullbacks and pushouts. Then, by Lemma 23, the square below is a pushout

$$\begin{array}{ccc} J' & \to & G'_0 \\ \downarrow & (\dagger) & \downarrow \\ J & \to & G_0 \end{array}$$

We are now in position to apply Construction 21 by taking $C_0 = J$, which yields

$$(G_0 \leftarrow J) \rightsquigarrow (G_{1,1} \leftarrow J) \rightsquigarrow^* (G_{n,1} \xleftarrow{\beta_1} J)$$

extending $(G'_0 \leftarrow J') \rightsquigarrow (G'_{1,1} \leftarrow J') \rightsquigarrow^* (G'_{n,1} \xleftarrow{\beta'} J')$ and

$$(G_0 \leftarrow J) \rightsquigarrow (G_{1,2} \leftarrow J) \rightsquigarrow^* (G_{m,2} \xleftarrow{\beta_2} J)$$

extending $(G'_0 \leftarrow J') \rightsquigarrow (G'_{1,1} \leftarrow J') \rightsquigarrow^* (G'_{m,2} \xleftarrow{\beta'} J')$.

The next step is to prove that $(G_{n,1} \xleftarrow{\beta_1} J) \cong (G_{m,2} \xleftarrow{\beta_2} J)$. To see this, it is enough to observe that both the following squares are pushouts of $J \xleftarrow{\xi} J' \xrightarrow{\beta'} W' = G'_{n,1} = G'_{m,2}$

$$\begin{array}{ccc} J' & \xrightarrow{\beta'} & G'_{n,1} \\ {\scriptstyle\xi}\downarrow & & \downarrow \\ J & \xrightarrow[\beta_1]{} & G_{n,1} \end{array} \qquad \begin{array}{ccc} J' & \xrightarrow{\beta'} & G'_{m,2} \\ {\scriptstyle\xi}\downarrow & & \downarrow \\ J & \xrightarrow[\beta_2]{} & G_{m,2} \end{array}$$

Indeed, the leftmost is a pushout by composition of squares (ϵ_n) and (δ_n) in the embedding construction and the rightmost by composition of (ϵ_m) and (δ_m).

To complete the proof, it remains to show that, in the above derivations, interface J extends to interface I as in the statement of the theorem. But this trivially holds by precomposing with $\iota : I \to J$. □

We are now ready to give our decidability result. To formulate it at the level of generality of adhesive categories, we need some additional definitions.

A *quotient* of an object X is an equivalence class of epis with domain X. Two epis $e_1 : X \to X_1$, $e_2 : X \to X_2$ are equivalent when there exists an isomorphism $\varphi : X_1 \to X_2$ such that $e_1 ; \varphi = e_2$. Note that quotient is the dual of *subobject*.

A DPO rewriting system with interfaces is *computable* when

- pullbacks are computable,
- for every pair of rules $L_i \leftarrow K_i \to R_i$, $L_j \leftarrow K_j \to R_j$, the set of quotients of $L_i + L_j$ is finite and computable,
- for all $G \leftarrow J$, it is possible to compute every $H \leftarrow J$ such that $(G \leftarrow J) \rightsquigarrow (H \leftarrow J)$.

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

Computability refers to the possibility of effectively computing each rewriting step as well as to have a finite number of pre-critical pairs. More precisely, the first two conditions ensure that the set of all pre-critical pairs is finite (since every object has finitely many quotients) and each of them can be computed, while the last one ensures that any possible rewriting step can also be computed. Thus, these assumptions rule out the rewriting of infinite structures, singleing out instead those structures where it is reasonable to apply the DPO mechanism, like finite hypergraphs in $\mathbf{Hyp}_\Sigma$, which are exactly what is needed for implementing rewriting of SMTs.

Corollary 24. *For a computable terminating DPO system with interfaces, confluence is decidable.*

Proof. We first observe that an arbitrary DPO system with interface is confluent if and only if all pre-critical pairs are joinable.

(1) If all pre-critical pairs are joinable then, by Theorem 2, the system is confluent.
(2) If not all pre-critical pairs are joinable, then at least one pair witnesses the fact that the system is not confluent.

Therefore, to decide confluence, it suffices to check that all pre-critical pairs are joinable.

Since the system is computable, there are only finitely many pre-critical pairs and these can be computed. For each pair, one can decide joinability: Indeed, each rewriting step can be computed (since the system is computable) and there are only finitely many $(H \leftarrow J)$ such that $(G \leftarrow J) \rightsquigarrow^* (H \leftarrow J)$ (since the system is terminating). □

It is worth to remark that this result is not in conflict with Theorem 1: Corollary 24 refers to the confluence of all hypergraphs with interfaces $G \leftarrow J$. The property that Theorem 1 states as undecidable is whether the rewriting is confluent for all *hypergraphs with empty interface* $G \leftarrow 0$. Observe that the restriction to hypergraphs with empty interface would make the above proof fail in point (2): indeed, thanks to Theorem 2, point (1) would hold also for hypergraphs with empty interface, but a non-joinable pre-critical pair originating from $S \leftarrow J$ with J non-empty does not necessarily witness that rewriting is not confluent for all $G \leftarrow 0$.

A similar problem arises with term rewriting, when restricting to the confluence of *ground terms* (Kapur et al. 1990). As an example, consider the following term rewriting system defined on the signature with two unary symbols, f and g, and one constant c

$$f(g(f(x))) \to x \qquad f(c) \to c \qquad g(c) \to c$$

The critical pair $f(g(x)) \leftarrow f(g(f(g(f(x))))) \to g(f(x))$ is not joinable, but the system is obviously ground confluent, as every ground term will eventually rewrite into c.

Our work therefore allows one to view Theorem 1 in a new light: as hypergraphs with empty interface are morally the graphical analogous of ground terms, we can say that ground confluence is not decidable for DPO rewriting with interfaces.

4. Confluence for PROP Rewriting with Frobenius Structure

As emphasised in the introduction, a major reason for interest in DPO rewriting with interfaces is that PROP rewriting (Section 2.3) may be interpreted therein. In this section, we investigate how our confluence result behaves with respect to this interpretation. The outcome is that confluence is decidable for terminating PROP rewriting systems, where terms are taken modulo a chosen special Frobenius structure (Corollary 30). For arbitrary symmetric monoidal theories, confluence is also decidable, provided that certain additional conditions hold (Corollary 43).

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

4.1 From PROPs to hypergraphs with interfaces

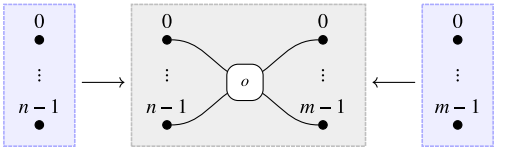
In this subsection, we report a result from Bonchi et al. (2022) that is crucial for the encoding of PROP rewriting into DPO rewriting with interfaces in $\mathbf{Hyp}_\Sigma$ (cf. Section 2.1).

First, we obtain our domain of interpretation by restricting the category $\mathsf{Csp}(\mathbf{Hyp}_\Sigma)$ whose objects are hypergraphs and arrows cospans, that is pairs $G_1 \to G_2 \leftarrow G_3$ of Σ-hypergraphs morphisms, up-to isomorphism in the choice of G_2.

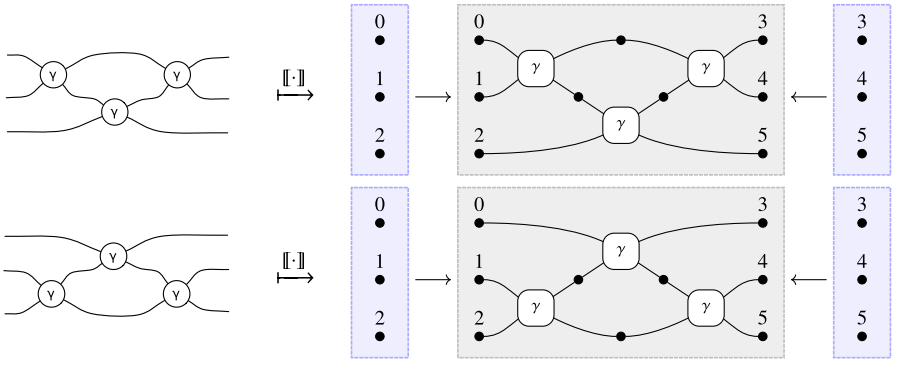
Definition 25 (Hypergraphs with interfaces)**.** *Any $k \in \mathbb{N}$ can be seen as a discrete hypergraph (i.e., with an empty set of edges) with k vertices. The objects of the PROP $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ are natural numbers and arrows $n \to m$ are cospans $n \to G \leftarrow m$ in $\mathbf{Hyp}_\Sigma$ (where n, m are considered as hypergraphs). $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$, therefore, is a full subcategory of $\mathsf{Csp}(\mathbf{Hyp}_\Sigma)$.*

Explicitly, composition in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is defined by pushout as in $\mathsf{Csp}(\mathbf{Hyp}_\Sigma)$ and the monoidal product $\oplus$ by the coproduct in $\mathbf{Hyp}_\Sigma$. The idea behind the discreteness restriction is that the arrows of the cospan tell what are the "left and right dangling wires" in the string diagram encoded by G. In pictures, we shall represent n and m as actual discrete graphs –with n and m nodes respectively– and use number labels (and sometimes colours, whenever available to the reader) to help visualise how they get mapped to nodes of G.

Given a signature Σ, we define a PROP morphism $[\![\cdot]\!] : \mathbf{S}_\Sigma \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$. Since $\mathbf{S}_\Sigma$ is the PROP freely generated by an SMT with no equations, it suffices to define $[\![\cdot]\!]$ on the generators: for each $o : n \to m$ in Σ, we let $[\![o]\!]$ be the following cospan of type $n \to m$



Example 26. The two sides of the PROP rewriting rule (3) (Example 9) get interpreted as the following cospans in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$



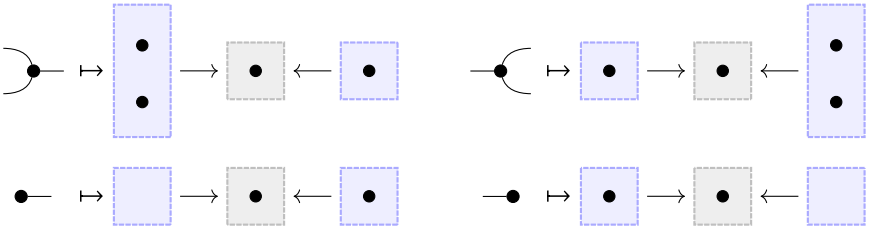
Proposition 27 (Bonchi et al. 2022)**.** $[\![\cdot]\!] : \mathbf{S}_\Sigma \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ *is faithful.*

The encoding $[\![\cdot]\!]$ is an important part of Theorem 3 below. This is a pivotal result in our exposition, as it serves as a bridge between algebraic and combinatorial structures. Indeed, it provides a presentation, by means of generators and equations, for the PROP $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$: the disjoint union of the SMTs of $\mathbf{S}_\Sigma$ and **Frob**.

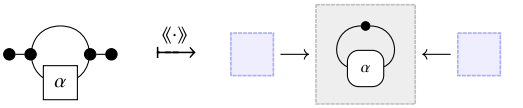
https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

Theorem 3 (Bonchi et al. 2022). *There is an isomorphism of PROPs* $\langle\!\langle \cdot \rangle\!\rangle \to \mathbf{S}_\Sigma + \mathbf{Frob} \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$.

The isomorphism $\langle\!\langle \cdot \rangle\!\rangle$ is given as the pairing $[[\![\cdot]\!], [\ \cdot\ ]] : \mathbf{S}_\Sigma + \mathbf{Frob} \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$, where $[\ \cdot\ ] : \mathbf{Frob} \to \mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is the PROP morphism mapping the generators of **Frob** as follows

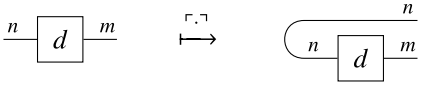


Here, **Frob** is used to model those features of the graph domain that are not part of the syntactic domain, for example the ability of building a "feedback loop" around some $\alpha : 1 \to 1$ in Σ



4.2 Confluence for rewriting in $\mathbf{S}_\Sigma + \mathbf{Frob}$

We can use Theorem 3 to apply results for graphs with interfaces to $\mathbf{S}_\Sigma + \mathbf{Frob}$. To this aim, first we need to interpret string diagrams as graphs with a *single* interface, instead of two as in their usual cospan interpretation. This can be easily achieved by applying the transformation $\ulcorner d \urcorner$ (introduced in Bonchi et al. 2022), which "rewires" a syntactic term d of $\mathbf{S}_\Sigma + \mathbf{Frob}$ by turning all of the inputs into outputs



Syntactic rewriting with "rewired" graphs is equivalent to rewriting 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: if $\langle\!\langle d \rangle\!\rangle = i \to G \leftarrow j$, then $\langle\!\langle \ulcorner d \urcorner \rangle\!\rangle = 0 \to G \leftarrow i + j$, which we may simply write as the graph with interface $G \leftarrow i + j$.

Example 28. The PROP rewriting system of Example 9 consists of just a single rule, let us call it $\langle d, e \rangle$. The resulting DPO rewriting system with interfaces is then presented in Example 10. Also, Example 26 is an intermediate step of this transformation, as it shows the cospans $[[\![c]\!]] = \langle\!\langle c \rangle\!\rangle$ and $[[\![d]\!]] = \langle\!\langle d \rangle\!\rangle$. One can obtain both graphs with interfaces $\langle\!\langle \ulcorner c \urcorner \rangle\!\rangle$ and $\langle\!\langle \ulcorner d \urcorner \rangle\!\rangle$ by "folding" the domain/codomain into the interface of Example 10.

Observe that a rule in the rewrite system $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle$ (defined as $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle = \{\langle\!\langle \ulcorner l \urcorner \rangle\!\rangle, \langle\!\langle \ulcorner r \urcorner \rangle\!\rangle \rangle \mid \langle l, r \rangle \in \mathcal{R}\}$) just consists of a pair of hypergraphs with a common interface; that is, it is a DPO rule of the form $L \leftarrow n + m \to R$. Thus, PROP rewriting in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ coincides with DPOI rewriting: together with Theorem 3, this correspondence yields the following result.

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

Theorem 4 (Bonchi et al. 2022)**.** *Let $\mathcal{R}$ be a rewriting system on* $\mathbf{S}_\Sigma + \mathbf{Frob}$*. Then*

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

One can read Theorem 4 as: DPO rewriting with interfaces is sound and complete for any symmetric monoidal theory with a chosen special Frobenius structure, that is one of shape $(\Sigma + \Sigma_F, \mathcal{E} + \mathcal{E}_F)$, with $(\Sigma_F, \mathcal{E}_F)$ the SMT of **Frob**. There are various relevant such theories in the literature, such as the ZX-calculus (Coecke and Duncan 2008), the calculus of signal flow graphs (Bonchi et al. 2014), the calculus of stateless connectors (Bruni et al. 2006) and monoidal computer (Pavlovic 2013).

The combination of the result above with Theorem 2 is however not sufficient for ensuring the decidability of the confluence for a terminating rewriting system $\mathcal{R}$ on $\mathbf{S}_\Sigma + \mathbf{Frob}$. Indeed, Theorem 2 and Theorem 4 ensure that if all the pre-critical pairs in $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle$ are joinable, then the rewriting in $\mathcal{R}$ is confluent. However, for the decidability of confluence in $\mathcal{R}$ the reverse is also needed: if one pre-critical pair in $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle$ is not joinable, then $\mathcal{R}$ should not be confluent. To conclude this fact, it is enough to check that all pre-critical pairs of $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle$ lay in the image of $\langle\!\langle \ulcorner \cdot \urcorner \rangle\!\rangle$, that is, that they all have discrete interfaces. The key observation is given by the lemma below.

Lemma 29 (Pre-critical pair with discrete interface)**.** *Consider a pre-critical pair in* $\mathbf{Hyp}_\Sigma$ *as in* (4)*, Definition 11. If both K_1 and K_2 are discrete, so is the interface J.*

Proof. For $i = 1, 2$, since K_i is discrete, the hyperedges of C_i are exactly those of G_i that are not in $f_i(L_i)$. Since $[f_1, f_2] : L_1 + L_2 \to S$ is epi, all the hyperedges of G are either in $f_1(L_1)$ or $f_2(L_2)$. Therefore, J cannot contain any hyperedge. □

Since in every rule $L \leftarrow K \rightarrow R$ in $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle$, K is discrete, from Lemma 29 and Theorem 2 we derive the following result.

Corollary 30. *Confluence is decidable for terminating rewriting systems on* $\mathbf{S}_\Sigma + \mathbf{Frob}$.

Proof. To decide confluence of a rewriting system $\mathcal{R}$ on $\mathbf{S}_\Sigma + \mathbf{Frob}$, it is enough to check whether all pre-critical pairs in $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle$ are joinable. Indeed, if all pre-critical pairs are joinable, then $\rightsquigarrow_{\mathcal{R}}$ is confluent by Theorems 2 and 4. For the other direction, suppose that there exists a pre-critical pair $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle \leftsquigarrow (S \leftarrow J) \rightsquigarrow_{\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle}$ that is not joinable. By construction, in every rule $L \leftarrow K \rightarrow R$ in $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle$, K is discrete. Therefore, by Lemma 29, also J is discrete. This is the key fact to entail that there exists d in $\mathbf{S}_\Sigma + \mathbf{Frob}$, such that $\langle\!\langle \ulcorner d \urcorner \rangle\!\rangle = (S \leftarrow J)$. By Theorem 4, d witnesses that $\rightsquigarrow_{\mathcal{R}}$ is not confluent.

Now, if $\mathcal{R}$ is terminating, then by Theorem 4 also $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle$ is terminating. The latter is also computable and therefore joinability of pre-critical pairs of $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle$ can easily be decided by following the steps in the second part of the proof of Corollary 24. □

5. Confluence for PROP Rewriting without Frobenius Structure

The presence of a chosen Frobenius structure simplifies the connection between syntactic PROP rewriting and hypergraph rewriting in two ways. The first is that Frobenius algebras give a natural, syntactic analogue to hypergraph vertices that are connected to many different hyperedges at once. These correspond to wires "splitting" and "merging," which are exactly captured by the Frobenius algebra. The second and perhaps more notable freedom provided by Frobenius algebras is the ability to interpret feedback loops and, in particular, hypergraphs and hypergraph rewriting that ignores any kind of acyclicity constraint. We have shown in Part II, and the reason why is shortly

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

recalled here in Section 5.2, that the absence of Frobenius algebras requires us not only to restrict the types of hypergraphs we consider but also which matches correspond to syntactically sound rewriting steps. This restriction on allowed matches has significant consequences in proving local confluence by critical pair analysis, which we will address in this section.

5.1 Monogamous acyclic hypergraphs

We first recall from Bonchi et al. (2020) a combinatorial characterisation of the image of $[\![\cdot]\!]$. It is based on a few preliminary definitions. We call a sequence of hyperedges $e_1, e_2, \ldots, e_n$ a *(directed) path* if at least one target of e_k is a source for e_{k+1} and a *(directed) cycle* if additionally at least one target of e_n is a source for e_1. The *in-degree* of a node v in an hypergraph G is the number of pairs (h, i) where h is an hyperedge with v as its i-th target. Similarly, the *out-degree* of v is the number of pairs (h, j) where h is an hyperedge with v as its j-th source. We call *input* nodes those with in-degree 0, *output* nodes those with out-degree 0, and *internal* nodes the others. We write $\mathsf{in}(G)$ for the set of inputs and $\mathsf{out}(G)$ for the set of outputs.

Definition 31. *A hypergraph G is monogamous acyclic (ma-hypergraph) if*
(1) it contains no cycle (acyclicity) and
(2) every node has at most in- and out-degree 1 *(monogamy).*

A cospan $n \xrightarrow{f} G \xleftarrow{g} m$ in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is monogamous acyclic (ma-cospan) when G is an ma-hypergraph, f is mono and its image is $\mathsf{in}(G)$, and g is mono and its image is $\mathsf{out}(G)$.

Theorem 5 (Bonchi et al. 2020)**.** *$n \to G \leftarrow m$ in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is in the image of $[\![\cdot]\!]$ iff it is an ma-cospan.*

We call a hypergraph with interface $G \leftarrow J$ monogamous acyclic (ma-hypergraph with interface) if it is of the form $G \xleftarrow{[i,o]} n + m$ for an ma-cospan $n \xrightarrow{i} G \xleftarrow{o} m$, up to isomorphism between J and $n + m$. Note that such an ma-cospan (and hence the ma-hypergraph with interface) may be uniquely fixed, so we can use the two representations interchangeably.

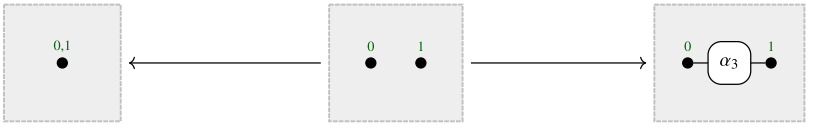
Finally, we say that a rule $L \leftarrow i + j \to R$ is an ma-rule if $i \to L \leftarrow j$ and $i \to R \leftarrow j$ are ma-cospans.

5.2 Convex rewriting and soundness

We are now in position to interpret PROP rewriting for $\mathbf{S}_\Sigma$ in DPO rewriting for ma-hypergraphs with interfaces, via the mapping that takes string diagrams to ma-hypergraphs with interfaces. Unfortunately, as shown in Part II (Bonchi et al. 2020), this interpretation is generally *unsound*. There are several things that can go wrong in the absence of Frobenius structure, as illustrated in the next two examples from Part II. These motivate our restrictions to PROP rewriting systems that make the interpretation sound, as presented in the sections below.

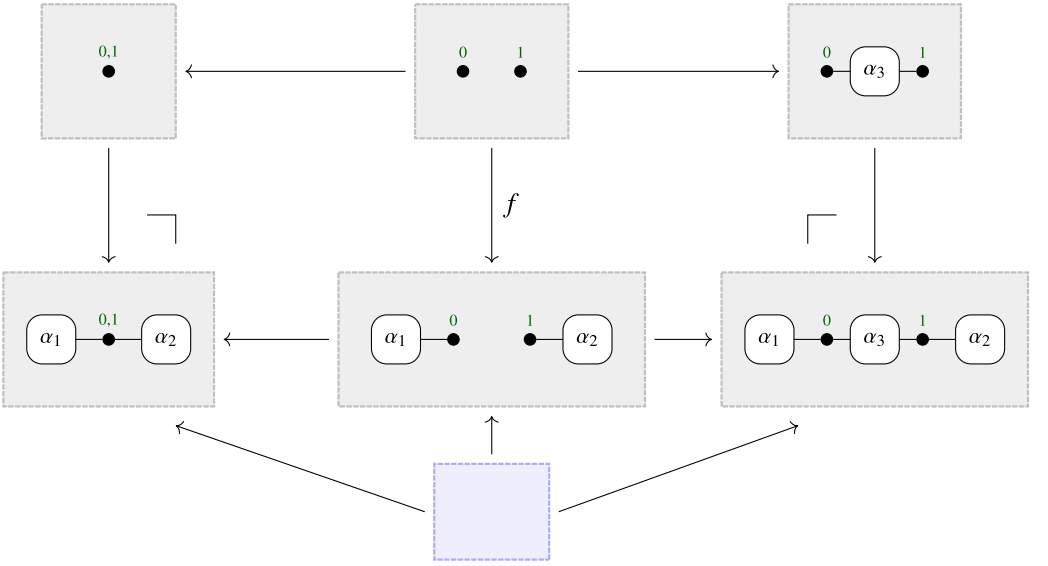
First, as noted in Part I (Bonchi et al. 2022), a DPOI rule can have multiple pushout complements when it is not left-linear, only some of which make sense without Frobenius structure.

Example 32. Consider $\Sigma = \{\alpha_1 : 0 \to 1, \alpha_2 : 1 \to 0, \alpha_3 : 1 \to 1\}$ and the PROP rewriting system $\mathcal{R} = \left\{ \text{———} \Rightarrow \text{—}\boxed{\alpha_3}\text{—} \right\}$ on $\mathbf{S}_\Sigma$. Its interpretation in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ is given by the rule

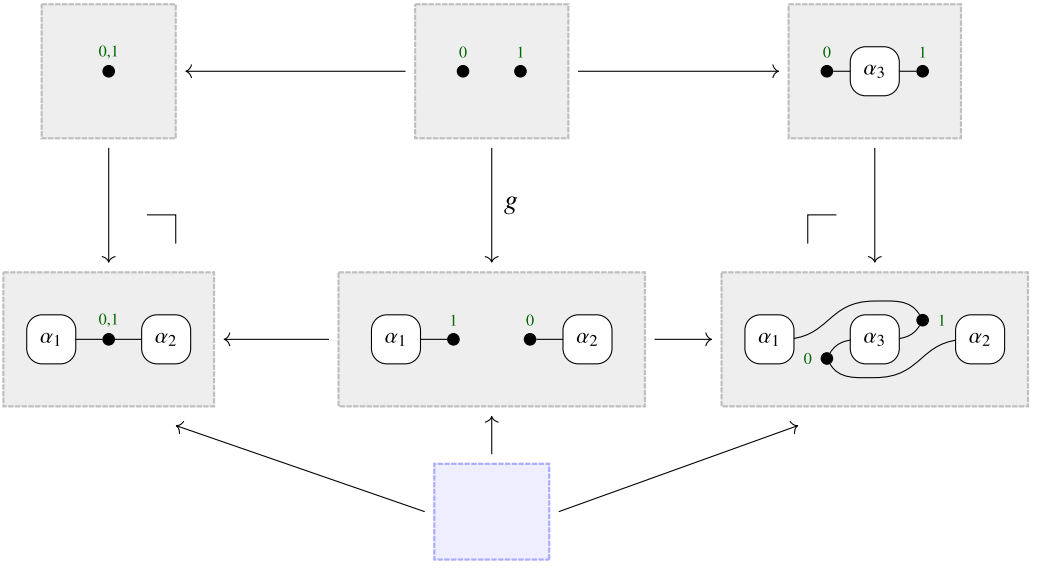


https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

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



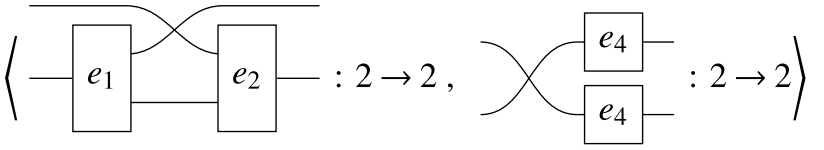
The different outcome is due to the fact that f maps 0 to the leftmost and 1 to the rightmost node, whereas g swaps the assignments. Even though both rewriting steps could be mimicked at the syntactic level in $\mathbf{S}_\Sigma + \mathbf{Frob}$, the second hypergraph rewrite yields a hypergraph that is illegal

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

for $\mathcal{R}$ in $\mathbf{S}_\Sigma$. 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.

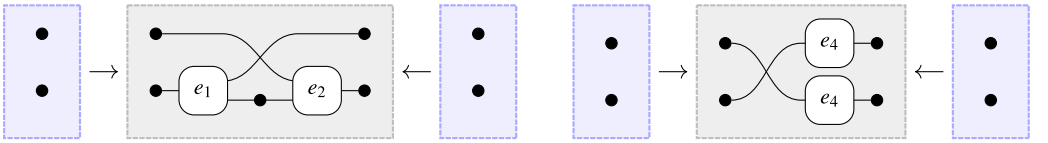
Next we can see that, even if a DPOI rewriting step yields a string diagram that can be expressed without Frobenius structure, it could be the case that equation itself cannot be proven in the SMT without introducing a feedback loop.

Example 33. Consider a $\Sigma = \{e_1 : 1 \to 2, e_2 : 2 \to 1, e_3 : 1 \to 1, e_4 : 1 \to 1\}$ and the following rewriting rule in $\mathbf{S}_\Sigma$

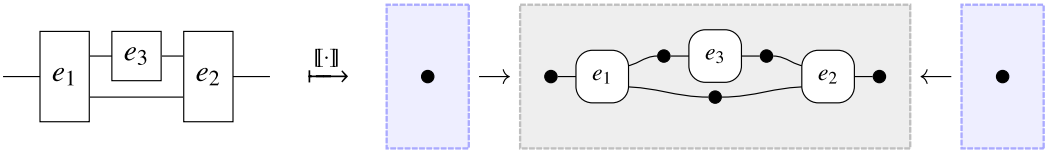


(12)

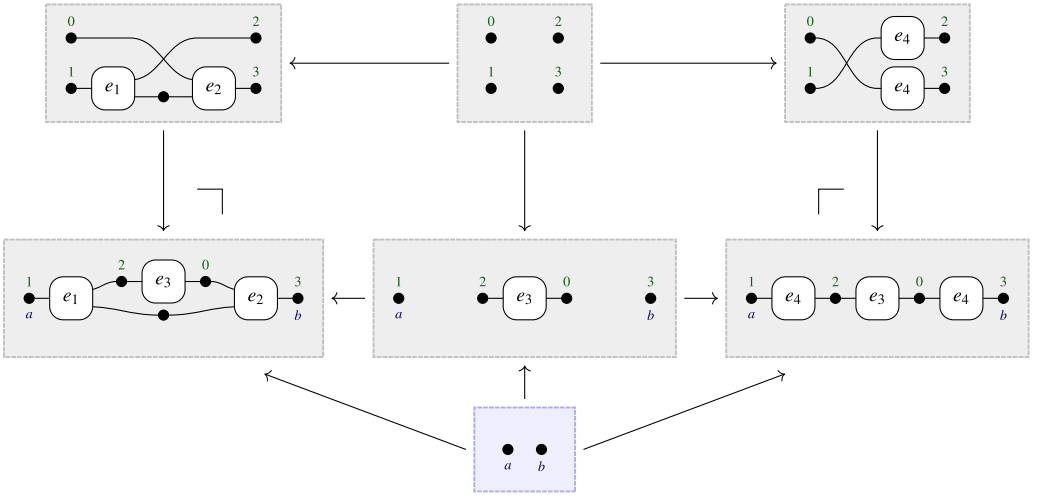
Left and right side are interpreted in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ as cospans



We introduce another diagram $c : 1 \to 1$ in $\mathbf{S}_\Sigma$ and its interpretation in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$

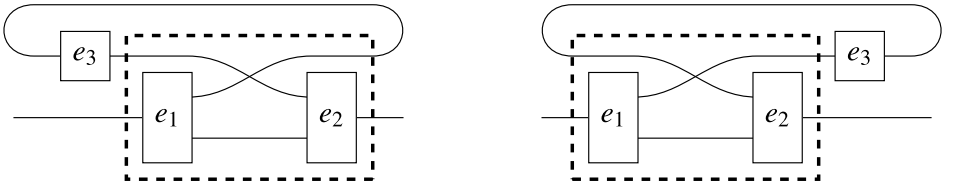


Now, rule (12) cannot be applied to c, even modulo the SMC equations. However, their interpretation yields a DPO rewriting step in $\mathsf{Csp}_D(\mathbf{Hyp}_\Sigma)$ as below



https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

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 occurence 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 two examples motivate the definition of *convex* rewriting, as a restriction of DPOI rewriting that rules out the above counterexamples and ensures soundness. We briefly recall the relevant definitions from Part II (Bonchi et al. 2020), referring to the discussion therein for more examples and properties of convex rewriting. As a preliminary step, we need to introduce a suitable restriction of the notion of pushout complement, called *boundary complement*.

Definition 34 (Boundary complement)**.** *For ma-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*

$$\begin{array}{ccc} L & \xleftarrow{a=[a_1,a_2]} & i+j \\ {\scriptstyle f}\downarrow & (\dagger) & \downarrow{\scriptstyle c=[c_1,c_2]} \\ G & \xleftarrow{\quad g \quad} & L^{\perp} \\ & {\scriptstyle [b_1,b_2]}\searrow & \uparrow{\scriptstyle [d_1,d_2]} \\ & & n+m \end{array}$$

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*

$$j + n \xrightarrow{[c_2,d_1]} L^{\perp} \xleftarrow{[c_1,d_2]} m + i \tag{13}$$

is an ma-cospan.

Note that boundary complements are unique when they exist (Bonchi et al. 2020) and that the pushout complement of Example 32 is not a boundary complement.

The next definition is a restriction on the possible matches, and rules out the other counterexample (Example 33).

Definition 35 (Convex match)**.** *We call* $m : L \to G$ *in* $\mathbf{Hyp}_{\Sigma}$ *a convex match if it is mono and its image* $m[L]$ *is convex, that is, for any nodes* v, v' *in* $m[L]$ *and any path p from v to* v' *in G, every hyperedge in p is also in* $m[L]$*.*

We now have all the ingredients to recall the notion of convex rewriting step, which is essentially a DPOI rewriting step relying on a boundary complement and a convex match.

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

Definition 36. *Given $n \to D \leftarrow m$ and $n \to E \leftarrow m$ ma-cospans, D* rewrites convexely into *E* with interface $n+m$ *–notation $(D \leftarrow n+m) \Rrightarrow_{\mathcal{R}} (E \leftarrow n+m)$– if there exist ma-rule $L \leftarrow i+j \to R$ in $\mathcal{R}$, object C, and morphisms such that the diagram below commutes and the squares are pushouts*

$$\begin{array}{ccccc} L & \xleftarrow{[a_1,a_2]} & i+j & \xrightarrow{[b_1,b_2]} & R \\ {\scriptstyle f}\downarrow & \urcorner & \downarrow & \ulcorner & \downarrow \\ D & \xleftarrow{\quad} & C & \xrightarrow{\quad} & E \\ & {\scriptstyle [q_1,q_2]}\nwarrow & \uparrow & \nearrow{\scriptstyle [p_1,p_2]} & \\ & & n+m & & \end{array} \tag{14}$$

and the following conditions hold

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

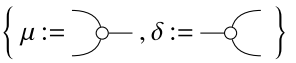
Note that in the definition above we implicitly assume that $i \to L \leftarrow j$ is an ma-cospan.

5.3 Failure of naive critical pair analysis for convex rewriting

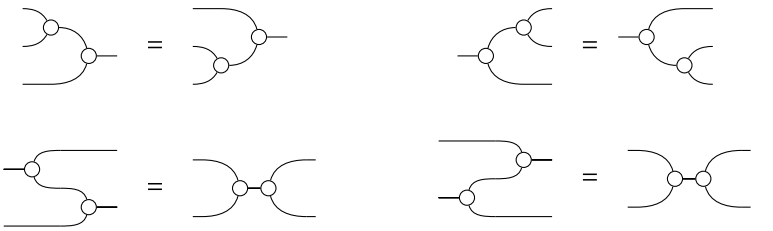
The main issue with using the critical pair analysis technique delineated before is that convexity is not preserved by clipping. That is, we can have critical overlaps H which form non-convex subgraphs $H \subseteq G$. Hence, it could be the case that a branching $H_1 \leftsquigarrow H \rightsquigarrow H_2$ is joinable using convex rewriting starting from H, but the lifted branching $G_1 \leftsquigarrow G \rightsquigarrow G_2$ will not be joinable.

This is what happens in the following example, taken from Part II (Bonchi et al. 2020).

Example 37. *Frobenius semi-algebras* are Frobenius algebras lacking the unit and counit equations. That is, they are the free PROP generated by the signature

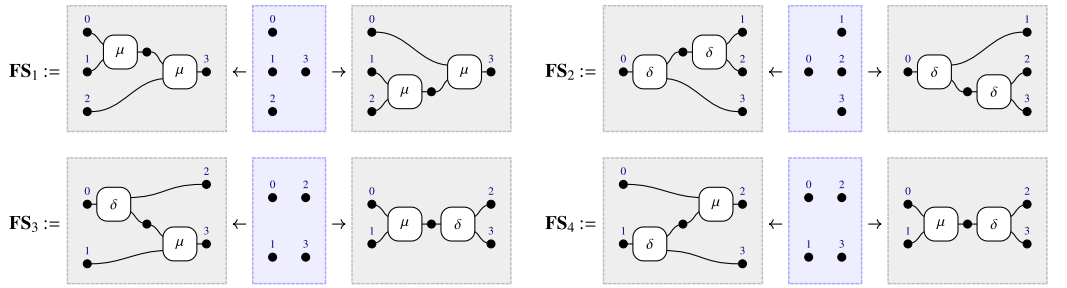


modulo the following equations



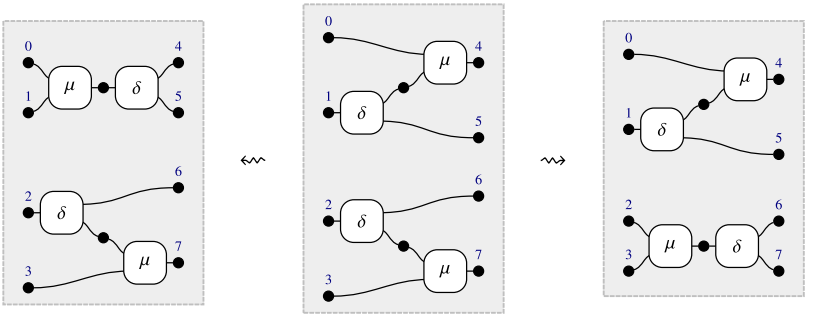
(15)

which can be represented as the following rewriting system

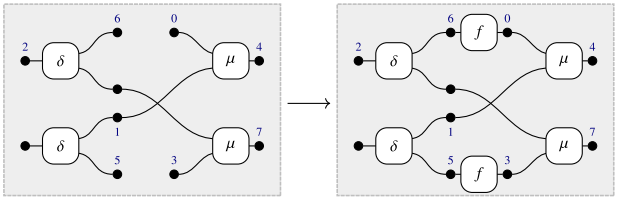


https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

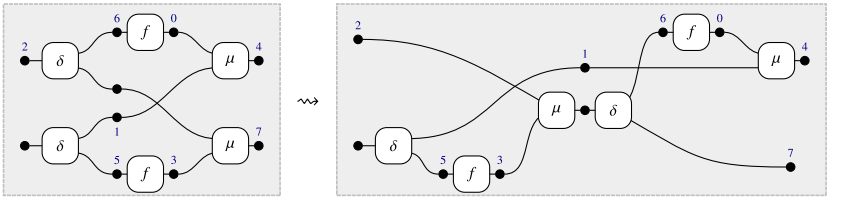
Suppose we start to consider the pre-critical pairs for this system. The following one is clearly joinable, since it involves parallel (and in fact totally disjoint) applications of $\mathbf{FS}_3$ and $\mathbf{FS}_4$



However, if we consider the middle graph in a larger context, for example



it no longer becomes joinable by convex rewriting. For example, suppose we apply $\mathbf{FS}_3$ on the larger graph above



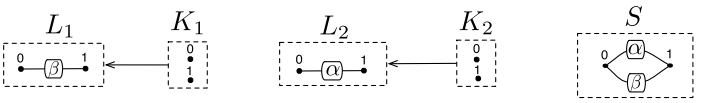
Then we are stuck: $\mathbf{FS}_4$ no longer has a convex matching, because applying $\mathbf{FS}_3$ introduced a new path from the output 5 to the input 0. Convexity guarantees we will not introduce cycles, and indeed in this case applying $\mathbf{FS}_4$ would introduce a new path from 0 to 5, and hence a cycle.

We present two solutions to this problem. The first is to put a strong restriction, called *left-connectedness*, on the rewriting systems being considered. The second is to develop a more in-depth notion of critical pair analysis, which accounts for the context-sensitivity of convex rewriting using *formal path extensions*.

They both rely on a more refined notion of pre-critical pair. One cannot simply reuse Definition 11, as we want to enforce that the common source $S \leftarrow J$ (cf. (4)) of the two derivations is an ma-hypergraph with interfaces, so that it is in the image of $\langle\!\langle \ulcorner \cdot \urcorner \rangle\!\rangle$ and we can reason about pre-critical pairs "syntactically" in $\mathbf{S}_\Sigma$. However, while Lemma 29 guarantees that this is always the case for rewriting systems on $\mathbf{S}_\Sigma + \mathbf{Frob}$, with Definition 11, this is not guaranteed for $\mathbf{S}_\Sigma$, as shown by the example below.

Example 38. We concoct a pre-critical pair by instantiating (4) as shown below

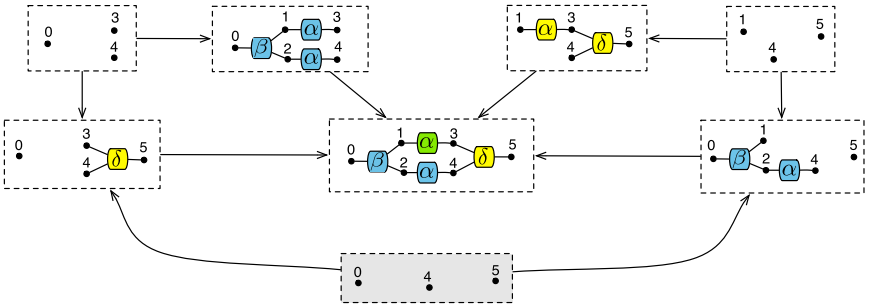
https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press



Although $L_1 \leftarrow K_1$ and $L_2 \leftarrow K_2$ are left-hand sides of left-connected rules, S is *not* monogamous, thus this pre-critical pair does not correspond to anything syntactic in $\mathbf{S}_\Sigma$.

Recall from the end of Section 5.1 that an ma-hypergraph with interface is required to have an interface corresponding exactly to the inputs and outputs of the ma-hypergraph. Because of this, using the notion of pre-critical pair from Definition 11 may yield too many nodes in the interface.

Example 39. Here is an example, where two rules match in an ma-hypergraph G, but the interface contains one extra node 4 which is neither an input nor an output of G



Motivated by these two examples, we give the following definition.

Definition 40 (Ma-pre-critical pair)**.** *Let $\mathcal{R}$ be a rewrite system consisting of ma-rules $L_1 \leftarrow K_1 \rightarrow R_1$ and $L_2 \leftarrow K_2 \rightarrow R_2$. Consider two derivations with source $S \leftarrow J$*

$$\begin{array}{ccccccccc} R_1 & \longleftarrow & K_1 & \longrightarrow & L_1 & & L_2 & \longleftarrow & K_2 & \longrightarrow & R_2 \\ \downarrow & & \downarrow & & & {\scriptstyle f_1}\searrow \quad \swarrow{\scriptstyle f_2} & & & \downarrow & & \downarrow \\ H_1 & \longleftarrow & C_1 & & & S & & & C_2 & & H_2 \\ & & & \searrow & & \uparrow & & \swarrow & & & \\ & & & & & J & & & & & \end{array} \qquad (16)$$

We say that $(H_1 \leftarrow J) \leftsquigarrow (S \leftarrow J) \rightsquigarrow (H_2 \leftarrow J)$ is an ma-pre-critical pair *if $[f_1, f_2] : L_1 + L_2 \rightarrow S$ is epi, (†) is a commuting diagram, and $S \leftarrow J$ is an ma-hypergraph with interface; it is* joinable *if there exists an ma-hypergraph with interface $W \leftarrow J$ such that $(H_1 \leftarrow J) \rightsquigarrow^* (W \leftarrow J) \ {}^*\!\!\leftsquigarrow (H_2 \leftarrow J)$.*

Comparing it with Definition 11, we are dropping the requirement that (†) is a pulback. However, note that up to an isomorphic choice of J, there is at most one ma-hypergraph with interface $S \leftarrow J$. Indeed, all ma-pre-critical pairs are also pre-critical pairs, and if I is the pullback along $C_1 \rightarrow S \leftarrow C_2$, the uniquely induced monomorphism $I \leftarrow J$ just weeds out from I those items whose image is neither an input nor an output of S.

5.4 Confluence for left-connected rewriting in $\mathbf{S}_\Sigma$

Definition 41. *An ma-hypergraph G is* strongly connected *if for every input $x \in \mathsf{in}(G)$ and output $y \in \mathsf{out}(G)$ there exists a path from x to y. A DPO system with interfaces is called left-connected if it*

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

is left-linear, every rule is an ma-rule and its left-hand side is strongly connected. We call a PROP rewriting system $\mathcal{R}$ on $\mathbf{S}_\Sigma$ left-connected if $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle$ is left-connected.

Non-commutative bimonoids (Example 6(c), see also Section 6.1 below) and the Yang-Baxter rule of Example 9 are examples of left-connected rewriting systems.

Intuitively, in Definition 41 strong connectedness prevents matches leaving "holes," as in Example 33, whereas left-linearity guarantees uniqueness of the pushout complements, and prevents the problem in Example 32. We are then able to prove the following.

Theorem 6. (Bonchi et al. 2020)**.** *Let $\mathcal{R}$ be a left-connected rewriting system on $\mathbf{S}_\Sigma$. Then,*
(1) if $d \Rightarrow_{\mathcal{R}} e$ then $\langle\!\langle \ulcorner d \urcorner \rangle\!\rangle \Rrightarrow_{\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle} \langle\!\langle \ulcorner e \urcorner \rangle\!\rangle$;
(2) if $\langle\!\langle \ulcorner d \urcorner \rangle\!\rangle \Rrightarrow_{\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle} \langle\!\langle \ulcorner e \urcorner \rangle\!\rangle$ then $d \Rightarrow_{\mathcal{R}} e$.

Remark 42. Note that for such rewriting systems, the further restriction of left-linearity is not particularly harmful, confluence-wise. Indeed, an ma-hypergraph with interface $G \leftarrow J$ is not mono iff G has one node that is both input and output, that is, an isolated node. A rule with a strongly connected $L \leftarrow K$ is not left-linear precisely when L is discrete, with a single node. Such a rule cannot be part of a terminating system, that is, one where local confluence implies confluence.

The above theorem allows us to use DPOI rewriting as a mechanism for rewriting $\mathbf{S}_\Sigma$.

We could now recast in this setting the considerations on parallel and critical pairs, as well as on joinability, as given in Definition 14 and Proposition 16, respectively. We move instead directly to state the confluence theorem for left-connected systems.

Theorem 7. (Local confluence for left-connected systems)**.** *For a left-connected DPO system with interfaces, if all ma-pre-critical pairs are joinable then rewriting is locally confluent: given an ma-hypergraph with interface $G_0 \leftarrow I$ and $(G_{1,1} \leftarrow I) \leftsquigarrow (G_0 \leftarrow I) \rightsquigarrow (G_{1,2} \leftarrow I)$, there exists an ma-hypergraph with interface $W \leftarrow I$ such that*

$$\begin{array}{ccccc} & & (G_0 \leftarrow I) & & \\ & \swarrow & & \searrow & \\ (G_{1,1} \leftarrow I) & & & & (G_{1,2} \leftarrow I) \\ & \searrow^{*} & & \swarrow^{*} & \\ & & (W \leftarrow I). & & \end{array}$$

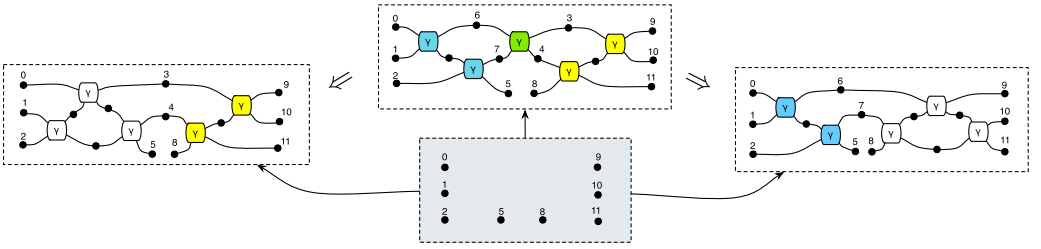
The proof of Theorem 7 follows steps analogous to the one of Theorem 2. The essential difference is that ma-pre-critical pairs now have interfaces that are not necessarily pullbacks. The assumption of left-connectedness is nevertheless enough to ensure that the fundamental pieces, namely Constructions 19 and 21, can be reproduced.

Corollary 43. *Let $\mathcal{R}$ be a terminating left-connected rewriting system on $\mathbf{S}_\Sigma$. Then confluence of $\rightsquigarrow_{\mathcal{R}}$ is decidable.*

Proof. By Theorem 6 and 7, it is enough to check whether pre-critical pairs in $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle$ are joinable. This is decidable since $\mathcal{R}$ is terminating and $\langle\!\langle \ulcorner \mathcal{R} \urcorner \rangle\!\rangle$ is computable. □

Example 44. The PROP rewriting system $\mathcal{R}$ of Example 9 is left-connected. Once interpreted as the DPO rewriting system with interfaces of Example 10, we can do critical pair analysis. The ma-pre-critical pair below (where the middle grey graph acts as the interface for the rewriting steps) is not joinable, meaning that $\mathcal{R}$ is not confluent

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press



We emphasise that the decision procedure relies on the fact that there are only *finitely many* pre-critical pairs to consider, the above one being the only one to feature a non-trivial overlap of rule applications. This is in contrast with a naive, "syntactic" analysis, which as we observe in Example 9 yields infinitely many pre-critical pairs for $\mathcal{R}$.

5.5 Convex critical pair analysis via formal path extensions

It is natural to ask whether we can extend critical pair analysis for convex rewriting beyond left-connected systems. It turns out that this is true, but the usual checking of critical pairs does not suffice: they need to be checked in a variety of contexts to account for the possible existence of paths from an output of the critical pair to an input. However, while it might seem necessary to check infinitely many contexts to account for every way a critical pair can be embedded in a larger graph, we get around this problem by considering *formal path contexts*. These abstract over the particular graph in which a critical pair is embedded, and only capture whether certain paths exist.

Definition 45. *For an ma-hypergraph G and a mono $m : G \to H$, the path relation of m, $R_m \subseteq \mathit{out}(G) \times \mathit{in}(G)$ is defined by letting $(y, x) \in R_m$ if and only if there is a path from the image of y to the image of x in H. We say a mono $m' : G \to H'$ path-covers m, written $m \lesssim m'$, if $R_m \subseteq R_{m'}$.*

Any morphism path-covers itself and path-covering is transitive, so $\lesssim$ is a pre-order (but not a partial order). We will write $m \sim m'$ for $m \lesssim m'$ and $m' \lesssim m$.

Lemma 46. *For an ma-hypergraph G and monos $m : G \to H, m' : G \to H'$ such that $m \lesssim m'$, we have for any mono $k : K \to G$, $m \circ k \lesssim m' \circ k$.*

Proof. If there is a path from an output to an input of $m \circ k[K]$ in H, it must split into 3 parts: a path from an output of $m \circ k[K]$ to an output of $m[G]$, a path from an output of $m[G]$ to an input of $m[G]$, and a path from an input of $m[G]$ to an input of $m \circ k[K]$. The first and third parts will also be present in the image of $m' \circ k$, and the second part will be whenever $m \lesssim m'$. Hence $m \circ k \lesssim m' \circ k$. □

We extend Σ with 3 new formal path generators $\mathcal{P} = \{$  $\}$, and introduce a family of monos which can produce any path relation.

Definition 47. *For an ma-hypergraph G and a binary relation $R \subseteq \mathsf{out}(G) \times \mathsf{in}(G)$, a mono $p : G \to P$ is called a path extension if P consists of $p[G] \cong G$, augmented by additional vertices and $\mathcal{P}$-labelled hyperedges such that there is a path from an output $y \in j$ to an input $x \in i$ if and only if $(y, x) \in R$.*

It follows by construction that $R = R_p$. Note that there is more than one way to construct a path extension for a given R, but if $R = R_p = R_q$ then $p \sim q$.

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

Lemma 48. *Let $L \leftarrow K \rightarrow R$ be a left-linear ma-rule in $\mathbf{Hyp}_\Sigma$, $p : G' \rightarrow P$ a path extension, and $m : L \rightarrow P$ a convex match. Then, m factors as $L \xrightarrow{m'} G' \xrightarrow{p} P$, and the convex rewrite of G' at m' extends to a convex rewriting step of P at m as follows*

$$\begin{array}{ccccc} L & \longleftarrow & K & \longrightarrow & R \\ {\scriptstyle m'}\downarrow & & \downarrow & & \downarrow{\scriptstyle n'} \\ G' & \longleftarrow & C' & \longrightarrow & H' \\ {\scriptstyle p}\downarrow & & \downarrow & & \downarrow{\scriptstyle q} \\ P & \longleftarrow & D & \longrightarrow & Q \end{array} \tag{17}$$

and furthermore q is a path extension.

Proof. Because L contains no $\mathcal{P}$-hyperedges, the image of every hyperedge in L under m must be in the image of G' under p. Furthermore, by left-linearity L contains no isolated vertex, so every vertex in the image of m is in the image of G' under p. Hence m factors as $p \circ m'$, as required.

The top pushouts in (17) are constructed as a convex DPO rewriting step. The bottom-left pushout is constructed as a pushout complement, which exists because m satisfies the gluing conditions with respect to K, so p satisfies the gluing conditions with respect to C' (which contains K). The bottom-right square is a pushout. This corresponds to the original rewrite $P \rightsquigarrow Q$ by uniqueness of the pushout complement D.

It only remains to show that q is a path extension. This follows from the fact that any $\mathcal{P}$-hyperedges in the pushout yielding Q must come from D. □

Lemma 49. *Let $L \leftarrow K \rightarrow R$, p, q, m', and n' be given as in Lemma 48. Then, for any mono $k : G' \rightarrow G$ such that $k \lesssim p$ the convex rewriting of G' at m' extends to a convex rewriting of G at $k \circ m'$ as follows, where $l \lesssim q$*

$$\begin{array}{ccccc} L & \longleftarrow & K & \longrightarrow & R \\ {\scriptstyle m'}\downarrow & & \downarrow & & \downarrow{\scriptstyle n'} \\ G' & \longleftarrow & C' & \longrightarrow & H' \\ {\scriptstyle k}\downarrow & & \downarrow & & \downarrow{\scriptstyle l} \\ G & \longleftarrow & C & \longrightarrow & H \end{array} \tag{18}$$

Proof. The bottom pushout squares are constructed from the rewrite of G' as in Construction 21. We first need to show that $k \circ m'$ is convex. If that were not the case, there would be a path from an output of the image of L to an input in G. But then, since $k \lesssim p$, we have by Lemma 46 that $k \circ m' \lesssim p \circ m'$. But then there is a path from an output of the image of L in P to an input, which contradicts convexity of $m = p \circ m'$. Hence $k \circ m'$ is convex.

It only remains to show that $l \lesssim q$. Inspecting the bottom pushout squares of (17), we note that, because of monogamy of G', it must be the case that any path from an output to an input of the image G' in P must be in C'. Hence, the same path will be in the image of H' in Q, so $R_p \subseteq R_q$. By the symmetry of the DPO construction, it is also the case that $R_q \subseteq R_p$ so $R_p = R_q$.

Applying the same argument to the bottom pushout squares of (18), we conclude that $R_l = R_k$. Since $k \lesssim p$, we have $R_l = R_k \subseteq R_p = R_q$, so $l \lesssim q$. □

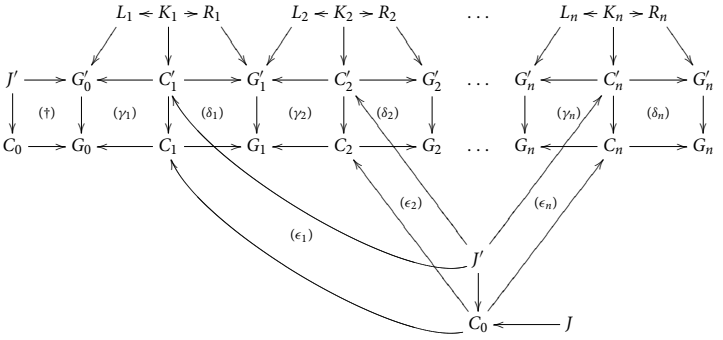
https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

Lemma 50. *For ma-rules $L_1 \leftarrow K_1 \rightarrow R_1$ and $L_2 \leftarrow K_2 \rightarrow R_2$ and an ma-pre-critical pair $(G'_{1,1} \leftarrow J') \leftsquigarrow (G'_0 \leftarrow J') \rightsquigarrow (G'_{1,2} \leftarrow J')$, let $k : G'_0 \rightarrow G_0$ be a mono such that the induced matches $L_1 \rightarrow G'_0 \rightarrow G_0$ and $L_2 \rightarrow G'_0 \rightarrow G_0$ are convex and $p : G'_0 \rightarrow P_0$ a path extension such that $k \lesssim p$. Then, we can obtain the following rewrites by extending the critical pair along k and p, respectively*

$$(G_{1,1} \leftarrow J) \leftsquigarrow (G_0 \leftarrow J) \rightsquigarrow (G_{2,1} \leftarrow J) \qquad (P_{1,1} \leftarrow K) \leftsquigarrow (P_0 \leftarrow K) \rightsquigarrow (P_{2,1} \leftarrow K)$$

If the right branching is joinable by convex rewriting, then so is the left one.

Proof. We apply essentially the same technique as the proof of Theorem 2, except that we additionally need to show that, when we extend derivations from the critical pair G'_0 to the full graph G_0 following Construction 21



each of the matches $L_i \rightarrow G'_i \rightarrow G_i$ is convex.

If all the critical pairs in $\mathcal{R}$ are path-joinable, then there exists a path extension $p : G'_0 \rightarrow P_0$ that path-covers $G'_0 \rightarrow G_0$ and $(P_{1,1} \leftarrow I) \leftsquigarrow (P_0 \leftarrow I) \rightsquigarrow (P_{1,2} \leftarrow I)$ is joinable. First, we can apply Lemma 48 to translate a convex rewrite $(P_0 \leftarrow I) \rightsquigarrow (P_{1,i} \leftarrow I)$ to a convex rewrite $(G'_0 \leftarrow J) \rightsquigarrow (G'_{1,i} \leftarrow J)$. We can then apply Lemma 49 to extend this to a convex rewrite $(G_0 \leftarrow J) \rightsquigarrow (G_{1,i} \leftarrow J)$. This yields a path extension $q : G'_{1,i} \rightarrow P_{2,i}$ that path-covers $G'_{1,i} \rightarrow G_{1,i}$, hence we can iterate this process to get a convex rewrite $(G_{1,i} \leftarrow J) \rightsquigarrow (G_{2,i} \leftarrow J)$, and so on.

When we path-join the critical pair, we obtain $P_{m,1} \cong P_{n,2}$. If we remove all of the $\mathcal{P}$-hyperedges (and nodes connected only to $\mathcal{P}$-hyperedges), this will restrict to an isomorphism $G'_{m,1} \cong G'_{n,2}$, which in turn yields an isomorphism $G_{m,1} \cong G_{n,2}$. Hence the branching $(G_{1,1} \leftarrow J) \leftsquigarrow (G_0 \leftarrow J) \rightsquigarrow (G_{1,2} \leftarrow J)$ is joinable by convex rewriting. □

Definition 51. *Given ma-rules $L_1 \leftarrow K_1 \rightarrow R_1$ and $L_2 \leftarrow K_2 \rightarrow R_2$ and an ma-pre-critical pair $(G'_0 \leftarrow J)$, a maximal path relation is a binary relation $R \subseteq \mathsf{out}(G'_0) \times \mathsf{in}(G'_0)$ such that*

(1) a mono $m : G'_0 \rightarrow H$ exists for an ma-hypergraph H with $R_m = R$,
(2) the induced matchings $L_1 \rightarrow G'_0 \rightarrow H$ and $L_2 \rightarrow G'_0 \rightarrow H$ are convex, and
(3) no relation satisfying (1) and (2) is a proper superset of R.

Definition 52. *Given ma-rules $L_1 \leftarrow K_1 \rightarrow R_1$ and $L_2 \leftarrow K_2 \rightarrow R_2$, an ma-pre-critical pair $(G'_{1,1} \leftarrow J') \leftsquigarrow (G'_0 \leftarrow J') \rightsquigarrow (G'_{1,2} \leftarrow J')$ is called path-joinable if for any maximal path relation R, there exists a path extension $p : G'_0 \rightarrow P$ with $R_p = R$ such that the branching $(P_{1,1} \leftarrow K) \leftsquigarrow (P_0 \leftarrow K) \rightsquigarrow (P_{1,2} \leftarrow K)$ obtained by lifting the two rewriting steps in the critical pair along p is joinable by convex rewriting.*

Theorem 8. *Let $\mathcal{R}$ be a convex DPOI rewriting system. If all ma-pre-critical pairs are path-joinable, then $\mathcal{R}$ is locally confluent.*

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

Proof. For any branching $(G_{1,1} \leftarrow J) \leftsquigarrow (G_0 \leftarrow J) \rightsquigarrow (G_{1,2} \leftarrow J)$, we can find an ma-pre-critical pair based at G_0' where the embedding $e : G_0' \to G_0$ has a path relation R_e satisfying conditions (1) and (2) in Definition 51. Hence, there exists a path extension $p : G_0' \to P_0$ where $R_e \subseteq R_p$ and the associated branching is joinable by convex rewriting. Hence by Lemma 50, the branching based at G_0 is also joinable by convex rewriting. Therefore $\mathcal{R}$ is locally confluent. □

The converse of this theorem is almost true, but with a small caveat that one needs to consider local confluence of ma-hypergraphs over the full signature $\Sigma + \mathcal{P}$ containing the formal path generators, rather than just Σ.

Theorem 9. *Let $\mathcal{R}$ be a convex DPOI rewriting system. If it is locally confluent for all ma-hypergraphs labelled by $\Sigma + \mathcal{P}$, then all ma-pre-critical pairs are path-joinable.*

The proof is immediate, since failing to join the path extension of a ma-pre-critical pair witnesses a failure of local confluence. There is no reason *a priori* that the above theorem would hold just for ma-hypergraphs. Hence, one can see the inclusion of the formal path generators as a sort of "stabilisation" of the theory that rules out certain degenerate cases of convex rewriting systems, such as those where certain paths never exist or can always be broken by rewriting.

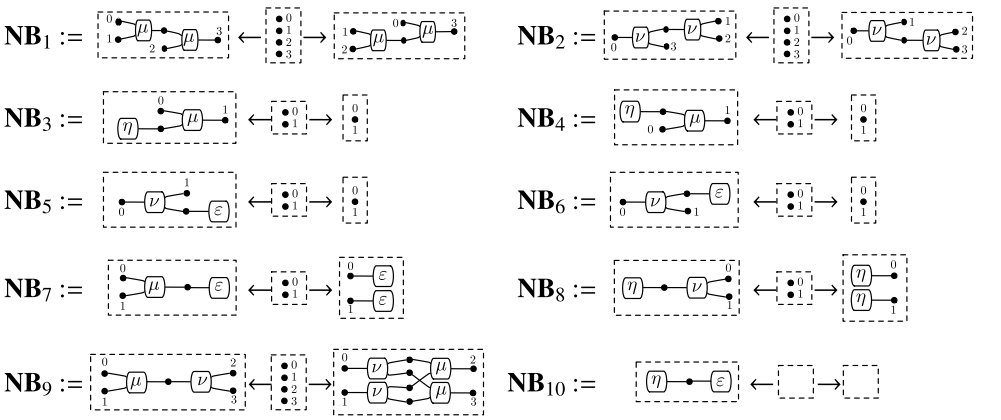
Theorem 8 gives us an effective way to check local confluence. For an ma-pre-critical pair, we need to enumerate all the maximal path relations, and for each one, construct a path extension and check if it is joinable. While there could in principle be exponentially many of these for each critical pair, conditions (1)–(3) in Definition 51 rule many of them out. We will see this process in action in the case study in Section 6.2.

6. Case Studies

We close the paper by providing two positive examples of our confluence results. Both of them concern ma-hypergraphs, distinguishing between left-connected systems and convex rewriting.

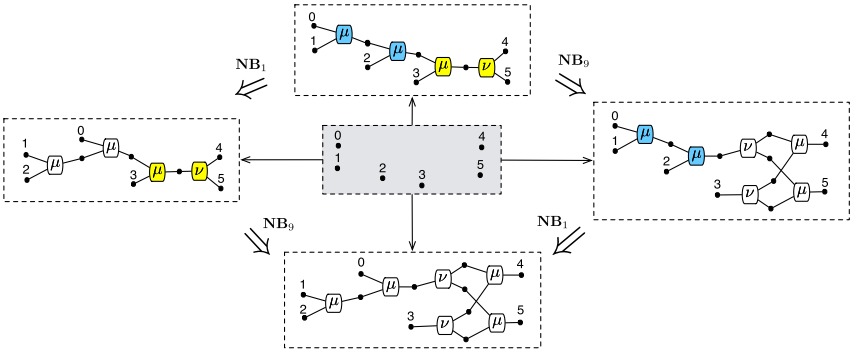
6.1 Left-connected and confluent: Non-commutative bimonoids

First case study is an application of the results on left-connected systems, showing confluence of the theory **NB** of non-commutative bimonoids (Example 6(c)). Below is the interpretation of the theory as a DPO system $\langle\!\langle \ulcorner \mathcal{R}_{\mathbf{NBiM}} \urcorner \rangle\!\rangle$, which was shown to be terminating in Bonchi et al. (2020)

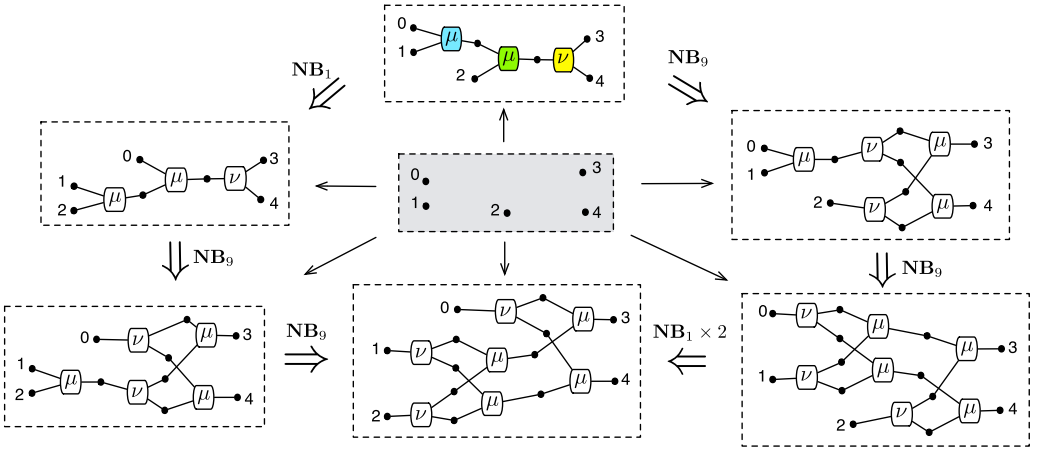


https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

Given that the system is terminating, it suffices to show local confluence. Observe that $\langle\!\langle \ulcorner \mathcal{R}_{\mathbf{NBiM}} \urcorner \rangle\!\rangle$ is left-connected: monogamy is ensured by the fact that it is in the image of $\langle\!\langle \ulcorner \cdot \urcorner \rangle\!\rangle$; strong connectedness and left-linearity hold by inspection of the set of rules. We can thus use Theorem 7 and local confluence follows from joinability of the ma-pre-critical pairs. Amongst them, the pairs without overlap of rule applications pose no problem: they are trivially joinable. One example is given below, with the middle grey graph acting as the interface for all depicted derivation steps

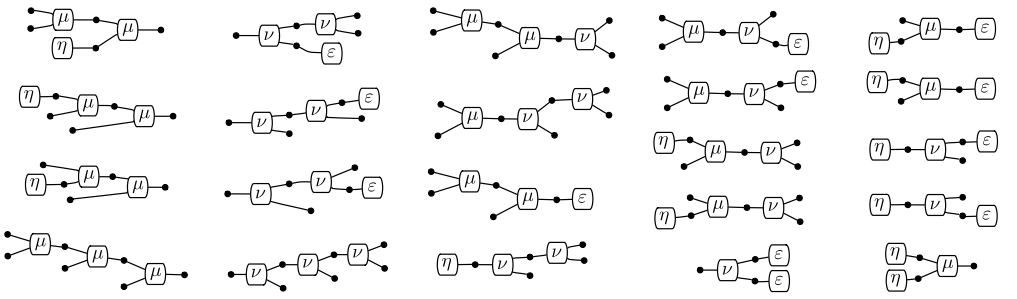


Thus, we confine ourselves to analysing actual critical pairs, with overlapping rule applications. One such pair is given below, also involving rules $\mathbf{NB}_1$ and $\mathbf{NB}_9$. Again, we show how it is joined, with the interface of each step drawn in the centre



Overall, there are 22 critical pairs to consider. For each of them, we only show the graph exhibiting the overlap. It is straightforward to check that the corresponding pairs are all joinable

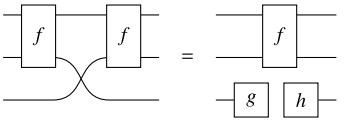
https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press



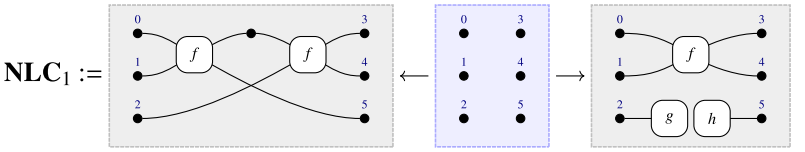
We can thereby conclude that **NB** is a confluent rewriting system. Since it is also terminating, equivalence of terms in **NB** is decidable by means of rewriting. Note that, by virtue of Corollary 43, the above pre-critical pair analysis can be automated.

6.2 A confluent, non-left-connected example

We now consider an example of a rewriting system that is not left-connected, and demonstrate a proof of confluence by means of path extensions. Let $\Sigma = \{f : 2 \to 2, g : 1 \to 0, h : 0 \to 1\}$, satisfying one equation

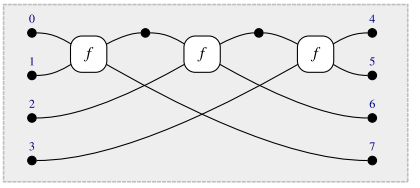


which translates into the following ma-rule



The rule strictly decreases the number of f-labelled hyperedges in a graph, so **NLC** is clearly terminating. Hence, it suffices to check local confluence to prove that **NLC** is confluent.

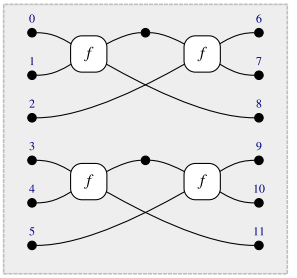
The rule **NLC** has two types of ma-pre-critical pairs. The first type is a genuine critical pair



(19)

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

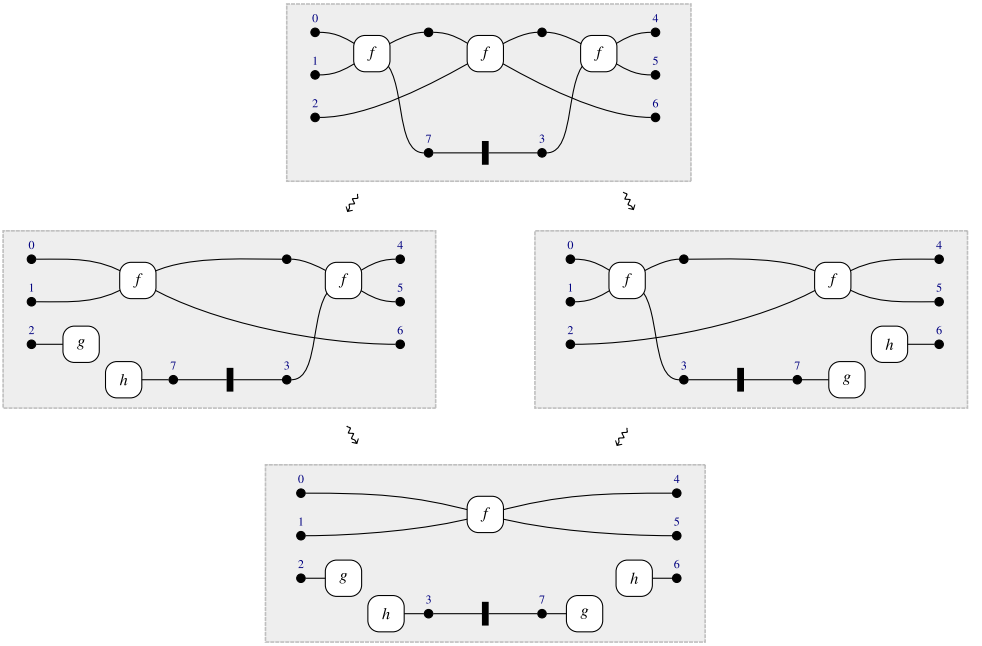
and the second is a parallel pair



(20)

All of the other ma-pre-critical pairs are variations of (19) and (20), obtained by joining some outputs to some inputs in such a way that the two matchings of $\mathbf{NLC}_1$ remain convex.

The maximal path relations for (19) and (20) can be computed by exhaustive enumeration (see Appendix B). For (19), there is only one maximal path relation $R_p = \{(7, 3)\}$. Hence, we can form the extension by adjoining a $\mathcal{P}$-hyperedge that creates a path from output 7 to input 3. This is then joinable

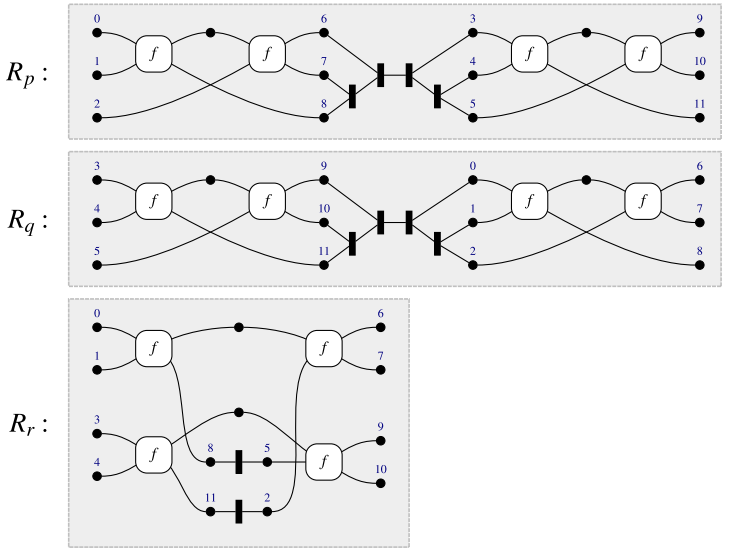


The ma-pre-critical pair (20) has three maximal path relations

$$R_p = \{6, 7, 8\} \times \{3, 4, 5\}$$
$$R_q = \{9, 10, 11\} \times \{0, 1, 2\}$$
$$R_r = \{(11, 2), (8, 5)\}$$

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

We can construct path extensions for each of these as follows



Each of these is joinable by convex rewriting. The 24 variations of (19) and (20) are all basically identical to these two cases, with the only difference being that one or more paths in the path extension is replaced by an output connected directly to an input. These are also all joinable, and hence, the system **NLC** is confluent.

7. Conclusions

The starting observation of this paper (Theorem 2) is that the Knuth-Bendix property holds for DPOI rewriting. As a consequence (Corollary 24), confluence is decidable for terminating systems.

The relevance of this is two-fold. On the conceptual side, it puts graph rewriting in tight correspondence with term rewriting: When considering rewriting with interfaces, confluence is decidable for graphs as it is for terms (Knuth and Bendix 1970), while the appropriate notion of ground confluence is undecidable in both cases (Kapur et al. 1990; Plump 1993).

On the side of applications, our result allows one to study confluence for string diagrams. One consequence of Theorem 2 and of our previous work in Bonchi et al. (2016) is that, for all those symmetric monoidal theories including a special Frobenius structure, which are already commonplace in computer science (Baez and Erbele 2015; Bonchi et al. 2015; Bruni et al. 2006, 2011; Coecke and Duncan 2008; Coecke et al. 2012; Fong et al. 2016; Sobociński and Stephens 2014), local confluence can be checked by means of critical pair analysis. Moreover, confluence can be decided automatically when termination is guaranteed (Corollary 30). Analogous results on critical pairs hold for those theories that do not include a special Frobenius structure, albeit with a few caveats. More precisely, confluence is accomplished by two kinds of restrictions. The first choice is to curb the family of admissible rules to left-connected systems (Theorem 7): The notion of critical pair is substantially unchanged, so that confluence can be decided automatically along the same lines as in the Frobenius case (Corollary 43). The second option is to restrict the family of admissible rewriting steps to convex matches, at the cost of checking a larger family of critical pairs in order to include path extensions (Theorem 8). The family is still computable, as witnessed by the algorithm proposed in Appendix B, thus obtaining again confluence decidability for terminating systems.

Our results apply to a variety of other non-Frobenius theories, such as those in Lafont (2003), Ghica (2013), Fiore and Campos (2013). In any case, in all the proposed scenarios, these decision

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

procedures are amenable to implementation in string diagram rewriting tools like Quantomatic (Kissinger and Zamdzhiev 2015) (via an encoding of hypergraphs) or directly in hypergraph-based rewriting tools.

Acknowledgements

Fabio Gadducci acknowledges support from MIUR PRIN 2017FTXR7S "IT-MaTTerS," Pawel Sobocinski from ESF funded Estonian IT Academy research measure (project 2014-2020.4.05.19-0001) and Estonian Research Council grant PRG1210, Fabio Zanasi from EPSRC EP/V002376/1.

References

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

Baldan, P., Gadducci, F. and Sobociński, P. (2011). Adhesivity is not enough: Local Church-Rosser revisited. In: Murlak, F. and Sankowski, P. (eds.) *MFCS 2011*, vol. 6907. *LNCS*. Springer, 48–59.

Bauer, G. and Otto, F. (1984). Finite complete rewriting systems and the complexity of the word problem. *Acta Informatica* **21** (5) 521–540.

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. (2017). Confluence of graph rewriting with interfaces. In: H. Yang (ed.), *ESOP 2017*, vol. 10201. *LNCS*. Springer, 141–169.

Bonchi, F., Gadducci, F., Kissinger, A., Sobociński, P. and Zanasi, F. (2020). String diagram rewrite theory II: Rewriting with symmetric monoidal structure. *Preprint available at arXiv:2104.14686*.

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

Bonchi, F., Sobociński, P. and Zanasi, F. (2014). A categorical semantics of signal flow graphs. In: Baldan, P. and Gorla, D. (eds.), *CONCUR 2014*, vol. 8704. *LNCS*. Springer, 435–450.

Bonchi, F., Sobociński, P. and Zanasi, F. (2015). Full abstraction for signal flow graphs. In: *POPL 2015*. ACM, 515–526.

Sander Bruggink, H. J., Cauderlier, R., Hülsbusch, M. and König, B. (2011). Conditional reactive systems. In: Chakraborty, S. and Kumar, A. (eds.) *FSTTCS 2011*, vol. 13. *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 191–203.

Bruni, R., Lanese, I. and Montanari, U. (2006). A basic algebra of stateless connectors. *Theoretical Computer Science* **366** (1–2) 98–120.

Bruni, R., Melgratti, H. C. and Montanari, U. (2011). A connector algebra for P/T nets interactions. In: Katoen, J.-P. and König, B. (eds.) *CONCUR 2011*, vol. 6901. *LNCS*. Springer, 312–326.

Campbell, G. and Plump, D. (2020). Confluence up to garbage. In: Gadducci, F. and Kehrer, T. (eds.) *ICGT 2020*, vol. 12150. *LNCS*. Springer, 20–37.

Coecke, B. and Duncan, R. (2008). Interacting quantum observables. In: Aceto, L., Damgård, I., Goldberg, L. A., Halldórsson, M. M., Ingólfsdóttir, A. and Walukiewicz, I. (eds.) *ICALP 2008*, vol. 5126. *LNCS*. Springer, 298–310.

Coecke, B., Duncan, R., Kissinger, A. and Wang, Q. (2012). Strong complementarity and non-locality in categorical quantum mechanics. In: *LICS 2012*. ACM, 245–254.

Corradini, A. (2016). On the definition of parallel independence in the algebraic approaches to graph transformation. In: Milazzo, P., Varró, D. and Wimmer, M. (eds.) *STAF 2016*, vol. 9946. *LNCS*. Springer.

Corradini, A., Duval, D., Löwe, M., Ribeiro, L., Machado, R., Costa, A., Azzi, G. G., Bezerra, J. S. and Rodrigues, L. M. (2018). On the essence of parallel independence for the double-pushout and sesqui-pushout approaches. In: Heckel, R. and Taentzer, G. (eds.) *Graph Transformation, Specifications, and Nets*, vol. 10800. *LNCS*. Springer, 1–18.

Ehrig, H. (1978). Introduction to the algebraic theory of graph grammars (A survey). In: Claus, V., Ehrig, H. and Rozenberg, G. (eds.) *Graph-Grammars and Their Application to Computer Science and Biology*, vol. 73. *LNCS*. Springer, 1–69.

Ehrig, H., Habel, A., Lambers, L., Orejas, F. and Golas, U. (2010). Local confluence for rules with nested application conditions. In: Ehrig, H., Rensink, A., Rozenberg, G. and Schürr, A. (eds.) *ICGT 2010*, vol. 6372. *LNCS*. Springer, 330–345.

Ehrig, H., Habel, A., Padberg, J. and Prange, U. (2004). Adhesive high-level replacement categories and systems. In: Ehrig, H., Engels, G., Parisi-Presicce, F. and Rozenberg, G. (eds.) *ICGT 2004*, vol. 2987. *LNCS*. Springer, 144–160, 2004.

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

Ehrig, H. and Kreowski, H.-J. (1976). Parallelism of manipulations in multidimensional information structures. In: Mazurkiewicz, A. W. (ed.) *MFCS 1976*, vol. 45. *LNCS*. Springer, 284–293.

Fiore, M. P. and Campos, M. D. (2013). The algebra of directed acyclic graphs. In: Coecke, B., Ong, L. and Panangaden, P. (eds.) *Computation, Logic, Games, and Quantum Foundations*, vol. 7860. *LNCS*. Springer, 37–51.
Fong, B., Sobociński, P. and Rapisarda, P. (2016). A categorical approach to open and interconnected dynamical systems. In: Grohe, M., Koskinen, E. and Shankar, N. (eds.), *LICS 2016*. ACM, 495–504.
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.
Ghica, D. R. (2013). Diagrammatic reasoning for delay-insensitive asynchronous circuits. In: Coecke, B., Ong, L. and Panangaden, P. (eds.) *Computation, Logic, Games, and Quantum Foundations*, vol. 7860. *LLNCS*. Springer, 52–68.
Habel, A., Müller, J. and Plump, D. (2001). Double-pushout graph transformation revisited. *Mathematical Structures in Computer Science* **11** (5), 637–688.
Huet, G. and Lankford, D. (1978). On the uniform halting problem for term rewriting systems. Technical Report 283, IRIA.
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*, vol. 172. *ENTCS*. Elsevier, 437–458.
Joyal, A., Street, R. and Verity, D. (1996). Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society* **119** (3) 447–468.
Kapur, D., Narendran, P. and Otto, F. (1990). On ground-confluence of term rewriting systems. *Information and Computation* **86** (1) 14–31.
Kissinger, A. and Zamdzhiev, V. (2015). Quantomatic: A proof assistant for diagrammatic reasoning. In: Felty, A. P. and Middeldorp, A. (eds.) *CADE 2015*, vol. 9195. *LNCS*. Springer, 326–336.
Knuth, D. E. and Bendix, P. B. (1970). Simple word problems in universal algebras. In: *Computational Problems in Abstract Algebra*. Pergamon Press, 263–297.
Lack, S. and Sobociński, P. (2005). Adhesive and quasiadhesive categories. *Theoretical Informatics and Applications* **39** (3) 511–546.
Lafont, Y. (2003). Towards an algebraic theory of Boolean circuits. *Pure and Applied Algebra* **184** (2–3) 257–310.
Lambers, L. and Orejas, F. (2020). Initial conflicts for transformation rules with nested application conditions. In: Gadducci, F. and Kehrer, T. (eds.) *ICGT 2020*, vol. 12150. *LNCS*. Springer, 109–127.
Mac Lane, S. (1965). Categorical algebra. *Bulletin of the American Mathematical Society* **71** (1) 40–106.
Mimram, S. (2010). Computing critical pairs in 2-dimensional rewriting systems. In: Lynch, C. (ed.) *RTA 2010*, vol. 6. *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 227–242.
Mimram, S. (2014). Towards 3-dimensional rewriting theory. *Logical Methods in Computer Science* **10** (2).
Padawitz, P. (1980). New results on completeness and consistency of abstract data types. In: Dembinski, P. (ed.) *MFCS 1980*, vol. 88. *LNCS*. Springer, 460–473.
Pavlovic, D. (2013). Monoidal computer I: Basic computability by string diagrams. *Information and Computation* **226**, 94–116.
Plump, D. (1993). Hypergraph rewriting: Critical pairs and undecidability of confluence. In: Sleep, M. R., Plasmeijer, M. J. and van Eekele, M. C. J. D. (eds.) *Term Graph Rewriting: Theory and Practice*. Wiley, 201–213.
Plump, D. (2010). Checking graph-transformation systems for confluence. In: Drewes, F., Habel, A., Hoffmann, B. and Plump, D. (eds.), *Manipulation of Graphs, Algebras and Pictures*, vol. 26. *ECEASST*. EASST.
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.
Sobociński, P. (2004). *Deriving Process Congruences from Reaction Rules*. PhD thesis, BRICS, University of Aarhus.
Sobociński, P. and Stephens, O. (2014). A programming language for spatial distribution of net systems. In: Ciardo, G. and Kindler, E. (eds.), *Computation, Logic, Games, and Quantum Foundations*, vol. 8489. *LNCS*. Springer, 150–169.

Appendix A. Competing Interest Declaration

Competing interests: The authors declare none.

Appendix B. Enumeration of Maximal Path Relations

The case study in section 6.2 makes use of the following Python code for enumerating maximal path relations. The main function `find_extensions` takes as arguments

- `inputs`: a set of input vertices,
- `outputs`: a set of output vertices,
- `paths`: a set of pairs indicating that a path connects the input to the output in the pre-critical pair,
- `forbidden_paths`: a set of pairs indicating that a path must *not* exist from an output to an input, due to convexity.

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

```
from itertools import chain, combinations

# iterator over the powerset of a set, ordered from largest
# to smallest subsets
def powerset(iterable):
    s = list(iterable)
    return map(lambda it: set(it),
            chain.from_iterable(combinations(s, r)
                for r in range(len(s)+2,-1,-1)))

def find_extensions(inputs, outputs, paths, forbidden_paths):
    paths_rev = set(map(lambda p: (p[1],p[0]), paths))

    forbidden = (forbidden_paths
                 .union({(i,i) for i in inputs})
                 .union({(i,i) for i in outputs})
                 .union(paths_rev))

    # returns true of the transitive closure of (c UNION paths)
    # contains no forbidden pairs
    def is_sln(c):
        c = c.union(paths)
        while True:
            d = c.copy()
            for (x,y0) in c:
                for (y1, z) in c:
                    if y0 == y1: d.add((x,z))
            if len(d.intersection(forbidden)) > 0:
                return False
            else:
                if len(c) == len(d):
                    return True
                else:
                    c = d

    candidates = set((i,j) for i in outputs for j in inputs) - forbidden

    slns = []
    sz = len(candidates) + 1
    for c in powerset(candidates):
        if len(c) < sz:
            print("%s solutions of size >= %s" % (len(slns), sz))
            sz = len(c)
        if is_sln(c) and not any(c <= s for s in slns):
            slns.append(c)
    return slns

########################################################################
```

https://doi.org/10.1017/S0960129522000123 Published online by Cambridge University Press

```
if __name__ == "__main__":
    exts1 = find_extensions(
        inputs = { 0, 1, 2, 3 },
        outputs = { 4, 5, 6, 7 },
        paths = {
            (0,4),(0,5),(0,6),(0,7),
            (1,4),(1,5),(1,6),(1,7),
            (2,4),(2,5),(2,6),
            (3,4),(3,5)
            },
        forbidden_paths = { (7,2), (6,3) }
    )

    print("\n\nCP1 got %s extensions:"% len(exts1))
    for e in exts1: print(e)

    exts2 = find_extensions(
        inputs = { 0, 1, 2, 3, 4, 5 },
        outputs = { 6, 7, 8, 9, 10, 11 },
        paths = {
            (0,6),(0,7),(0,8),
            (1,6),(1,7),(1,8),
            (2,6),(2,7),
            (3,9),(3,10),(3,11),
            (4,9),(4,10),(4,11),
            (5,9),(5,10)
            },
        forbidden_paths = { (8,2), (11,5) }
    )

    print("\n\nCP2 got %s extensions:"% len(exts2))
    for e in exts2: print(e)
```

Cite this article: 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* **32**, 829–869. https://doi.org/10.1017/S0960129522000123