Bialgebraic Semantics for String Diagrams

Filippo Bonchi
University of Pisa

Robin Piedeleu
University College London

Pawel Sobocinski
University of Southampton

Fabio Zanasi
University College London

Abstract

Turi and Plotkin's bialgebraic semantics is an abstract approach to specifying the operational semantics of a system, by means of a distributive law between its syntax (encoded as a monad) and its dynamics (an endofunctor). This setup is instrumental in showing that a semantic specification (a coalgebra) satisfies desirable properties: in particular, that it is compositional.

In this work, we use the bialgebraic approach to derive well-behaved structural operational semantics of *string diagrams*, a graphical syntax that is increasingly used in the study of interacting systems across different disciplines. Our analysis relies on representing the two-dimensional operations underlying string diagrams in various categories as a monad, and their bialgebraic semantics in terms of a distributive law for that monad.

As a proof of concept, we provide bialgebraic compositional semantics for a versatile string diagrammatic language which has been used to model both signal flow graphs (control theory) and Petri nets (concurrency theory). Moreover, our approach reveals a correspondence between two different interpretations of the Frobenius equations on string diagrams and two synchronisation mechanisms for processes, à la Hoare and à la Milner.

2012 ACM Subject Classification Theory of Computation → Categorical Semantics

Keywords and phrases String Diagram, Structural Operational Semantics, Bialgebraic semantics

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2019.33

Acknowledgements RP and FZ acknowledge support from EPSRC grant EP/R020604/1.

1 Introduction

Starting from the seminal works of Hoare and Milner, there was an explosion [16,17,27,36,42] of interest in process calculi: formal languages for specifying and reasoning about concurrent systems. The beauty of the approach, and often the focus of research, lies in *compositionality*: essentially, the behaviour of composite systems—usually understood as some kind of process equivalence, the most famous of which is bisimilarity—ought to be a function of the behaviour of its components. The central place of compositionality led to the study of syntactic formats for semantic specifications [4,19,25]; succinctly stated, syntactic operations with semantics defined using such formats are homomorphic wrt the semantic space of behaviours.

Another thread of concurrency theory research [26,30,40] uses graphical formalisms, such as Petri nets. These often have the advantage of highlighting connectivity, distribution and the communication topology of systems. They tend to be popular with practitioners in part because of their intuitive and human-readable depictions, an aspect that should not be underestimated: indeed, pedagogical texts introducing CCS [27] and CSP [36] often resort to pictures that give intuition about topological aspects of syntactic specifications. However, compositionality has not—historically—been a principal focus of research.

30th International Conference on Concurrency Theory (CONCUR 2019).
Editors: Wan Fokkink and Rob van Glabbeek; Article No. 33; pp. 33:1–33:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this paper we propose a framework that allows us to eat our cake and have it too. We use *string diagrams* [43] which have an intuitive graphical rendering, but also come with algebraic operations for composition. String diagrams combine the best of both worlds: they are a (2-dimensional) syntax, but also convey important topological information about the systems they specify. They have been used in recent years to give compositional accounts of quantum circuits [1,18], signal flow graphs [2,10,21], Petri nets [6], and electrical circuits [3,24], amongst several other applications.

Our main contribution is the adaptation of Turi and Plotkin's bialgebraic semantics (*abstract GSOS*) [32,45] for string diagrams. By doing so, we provide a principled justification and theoretical framework for giving definitions of operational semantics to the generators and operations of string diagrams, which are those of monoidal categories. More precisely we deal with string diagrams for symmetric monoidal categories which organise themselves as arrows of a particularly simple and well-behaved class known as *props*. Similar operational definitions have been used in the work on the algebra of Span(Graph) [31], tile logic [23], the wire calculus [44] and recent work on modelling signal flow graphs and Petri nets [6,10]. In each case, semantics was given either monolithically or via a set of SOS rules. The link with bialgebraic framework—developed in this paper—provides us a powerful theoretical tool that (i) justifies these operational definitions and (ii) guarantees compositionality.

In a nutshell, in the bialgebraic approach, the syntax of a language is the initial algebra (the algebra of terms) T_Σ for a signature functor Σ. A certain kind of distributive law, an *abstract GSOS* specification [45], induces a coalgebra (a state machine) $\beta\colon T_\Sigma \to \mathcal{F}T_\Sigma$ capturing the operational semantics of the language. The final $\mathcal{F}$-coalgebra Ω provides the denotational universe: intuitively, the space of all possible behaviours. The unique coalgebra map $[\![\cdot]\!]_\beta\colon T_\Sigma \to \Omega$ represents the denotational semantics assigning to each term its behaviour.

$$\begin{tikzcd} T_\Sigma \arrow[r, dashed, "{[\![\cdot]\!]_\beta}"] \arrow[d, "\beta"'] & \Omega \arrow[d] \\ \mathcal{F}(T_\Sigma) \arrow[r, "{\mathcal{F}([\![\cdot]\!]_\beta)}"'] & \mathcal{F}(\Omega) \end{tikzcd} \tag{1}$$

The crucial observation is that (1) lives in the category of Σ-algebras: Ω also carries a Σ-algebra structure and the denotational semantics is an algebra homomorphism. This means that the behaviour of a composite system is determined by the behaviour of the components, e.g. $[\![s + t]\!] = [\![s]\!] + [\![t]\!]$, for an operation $+$ in Σ.

We show that the above framework can be adapted to the algebra of string diagrams. The end result is a picture analogous to (1), but living in the category of props and prop morphism. As a result, the denotational map is a prop morphism, and thus guarantees compositionality with respect to sequential and parallel composition of string diagrams.

Adapting the bialgebraic approach to the 2-dimensional syntax of props requires some technical work since, e.g. the composition operation of monoidal categories is a dependent operation. For this reason we need to adapt the usual notion of a syntax endofunctor on the category of sets; instead we work in a category Sig whose objects are spans $\mathbb{N} \leftarrow \Sigma \to \mathbb{N}$, with the two legs giving the number of dangling wires on the left and right of each diagram. The operations of props are captured as a Sig-endofunctor, which yields string-diagrams-as-initial-algebra, and a quotient of the resulting free monad, whose algebras are precisely props.

In addition to the basic laws of props, we also consider the further imposition of the equations of special Frobenius algebras. We illustrate the role of this algebraic structure

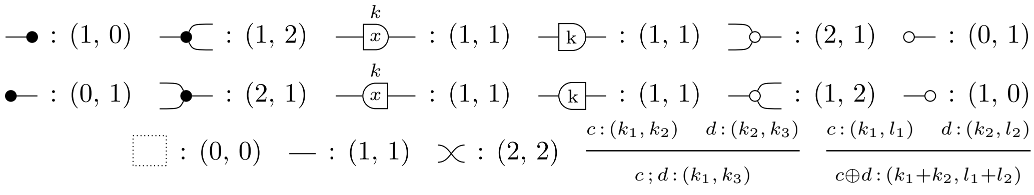


Figure 1 Sorting discipline for Circ_R

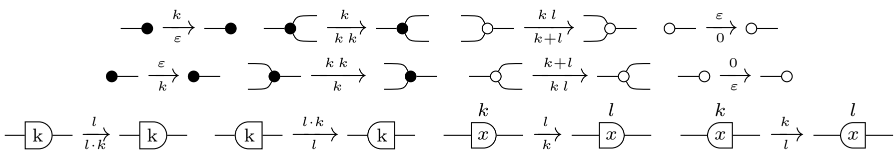


Figure 2 Structural Operational Semantics for the generators of Circ_R. Intuitively, from left to right, these are elementary connectors modelling discard, copy, one-place register, multiplication by a scalar, addition, and the constant zero.

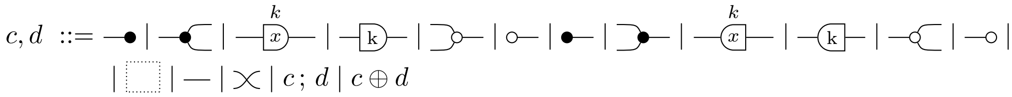
with our running example, a string diagrammatic process calculus Circ_R that has two Frobenius structures and can be equipped with two different semantics, one which provides a compositional account of signal flow graphs for linear time invariant dynamical systems [10], and one which is a compositional account of Petri nets [6].

We conclude with an observation that ties our work back to classical concepts of process calculi and show that the two Frobenius structures of Circ_R are closely related to two different, well-known synchronisation patterns, namely those of Hoare's CSP [27] and Milner's CCS [36].

Structure of the paper. In §2 we introduce our main example and recall some preliminaries, followed by a recapitulation of bialgebraic approach in §3. We develop the technical aspects of string-diagrams-as-syntax in §4 and adapt the bialgebraic approach in §5. Finally, we exhibit the connection with classical synchronisation mechanisms in §6 and conclude in §7.

2 Motivating Example

As our motivating example, we recall from [6, 9, 11] a basic language Circ_R given by the grammar below. Values k in and range over elements of a given semiring R.


(2)
(3)

The language does not feature variables; on the other hand, a simple sorting discipline is necessary. A sort is a pair (n, m), with $n, m \in \mathbb{N}$. Henceforth we will consider only terms sortable according to the rules in Figure 1. An easy induction confirms uniqueness of sorting.

The operational meaning of terms is defined recursively by the structural rules in Figs. 2 and 3 where k, l range over R and $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ over $\mathsf{R}^\star$, the set of words over R. We denote the empty word by ε and concatenation of $\boldsymbol{a}, \boldsymbol{b}$ by $\boldsymbol{ab}$. As expected $+, \cdot$ and 0 denote respectively the sum, the product and zero of the semiring R. For any term $c\colon (n, m)$, the rules yield

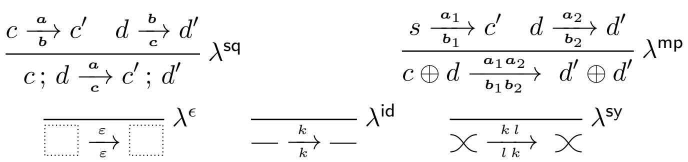
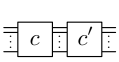
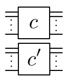
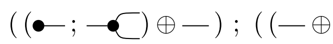
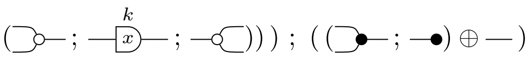


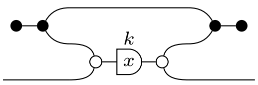
Figure 3 Structural operational semantics for the operations of Circ_R.

a labelled transition system where each transition has form $c \xrightarrow[\boldsymbol{b}]{\boldsymbol{a}} d$. By induction, it is immediate that d has the same sort as c, the word $\boldsymbol{a}$ has length n, and $\boldsymbol{b}$ has length m.

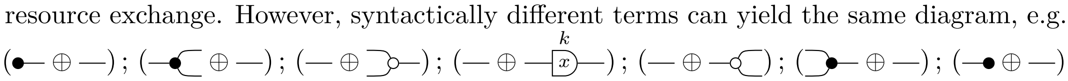
Our chief focus in this paper is the study of semantics specifications of the kind given in Figs. 2 and 3. So far, the technical difference with typical GSOS examples [4] is the presence of a sorting discipline. A more significant difference, which we will now highlight, is that sorted terms are considered up-to the laws of symmetric monoidal categories. As such, they are "2-dimensional syntax" and enjoy a pictorial representation in terms of string diagrams.

2.1 From Terms to String Diagrams

In (2)-(3) we purposefully used a graphical rendering of the components. Indeed, terms of Circ_R are usually represented graphically, according to the convention that $c\,;\,c'$ is drawn  and $c \oplus c'$ is drawn . For instance, the term   is depicted as the following diagram.

p_k ::=  (4)

Given this graphical convention, a sort gives the number of dangling wires on each side of the diagram induced by a term. A transition $c \xrightarrow[\boldsymbol{b}]{\boldsymbol{a}} d$ means that c may evolve to d when the values on the dangling wires on the left are $\boldsymbol{a}$ and those on the right are $\boldsymbol{b}$. When R is the natural numbers, the diagram in (4) behaves as a place of a Petri nets containing k tokens: any number of tokens can be inserted from its left and at most k tokens can be removed from its right. Indeed, by the rules in Figs. 2 and 3, $p_k \xrightarrow[o]{i} p_{k'}$ iff $o \leq k$ and $k' = i + k - o$.

The graphical notation is appealing as it highlights connectivity and the capability for resource exchange. However, syntactically different terms can yield the same diagram, e.g.  also yields (4). Indeed, one defines diagrams to be terms modulo *structural congruence*, denoted by $\equiv$. This is the smallest congruence over terms generated by the equations of strict symmetric monoidal categories (SMCs):

$$(f \oplus g) \oplus h \;\equiv\; f \oplus (g \oplus h) \qquad (\epsilon \oplus f) \;\equiv\; f \qquad (f \oplus \epsilon) \;\equiv\; f \qquad \sigma_{1,1}\,;\,\sigma_{1,1} \;\equiv\; id_2 \quad (5)$$

$$(f\,;\,g) \oplus (h\,;\,i) \;\equiv\; (f \oplus h)\,;\,(g \oplus i) \qquad (f\,;\,g)\,;\,h \;\equiv\; f\,;\,(g\,;\,h) \qquad (f\,;\,id_m) \;\equiv\; f \quad (6)$$

$$(id_n; f) \;\equiv\; f \qquad (\sigma_{1,n}; (f \oplus id_1)) \equiv (id_1 \oplus f); \sigma_{1,m} \qquad (\sigma_{n,1}; (id_1 \oplus g)) \equiv (g \oplus id_1); \sigma_{m,1} \quad (7)$$

where identities $id_n : (n,\, n)$ and symmetries $\sigma_{n,m} : (n + m,\, m + n)$ can be recursively defined starting from id_0 :=  and $\sigma_{1,1}$:= . Therefore, sorted diagrams $c : (n,\, m)$ are the arrows $n \to m$ of an SMC with objects the natural numbers, also called a prop [35].

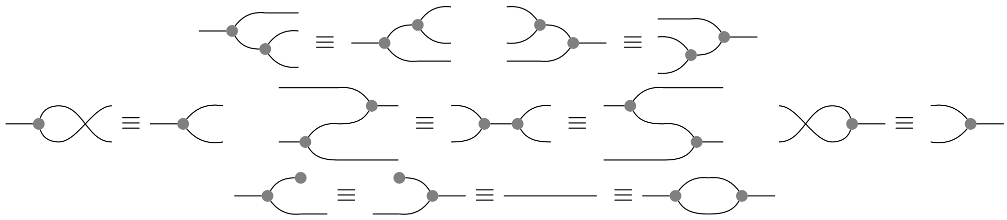


Figure 4 Axioms of special Frobenius bimonoids

2.2 Frobenius Bimonoids

We will also consider additional algebraic structure for the black (, , ,) and the white (, , ,) components. When R is the field of reals, $\mathsf{Circ}_{\mathsf{R}}$ models linear dynamical systems [2, 11, 21] and both the black and the white structures form special Frobenius bimonoids, meaning the axioms of Fig. 4 hold, replacing the gray circles by either black or white. When R is the semiring of natural numbers, $\mathsf{Circ}_{\mathsf{R}}$ models Petri nets [6] and only the black structure satisfies these equations. In § 6, we shall see that the black Frobenius structure gives rise to the synchronisation mechanism used by Hoare in CSP [28], while the white Frobenius structure to that used by Milner in CCS [36].

3 Background: Bialgebras and GSOS Specifications

For more detailed background and simple examples showcasing the notions recalled below, we refer the reader to the extended version of the present work [7].

Distributive laws and bialgebras. A *distributive law* of a monad $(\mathcal{T}, \eta, \mu)$ over an endofunctor $\mathcal{F}$ is a natural transformation $\lambda \colon \mathcal{T}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{T}$ s.t. $\lambda \circ \eta_{\mathcal{F}} = \mathcal{F}\eta$ and $\lambda \circ \mu_{\mathcal{F}} = \mathcal{F}\mu \circ \lambda_{\mathcal{T}} \circ \mathcal{T}\lambda$. A λ-*bialgebra* is a triple (X, α, β) s.t. (X, α) is an Eilenberg-Moore algebra for $\mathcal{T}$, (X, β) is a $\mathcal{F}$-coalgebra and $\mathcal{F}\alpha \circ \lambda_X \circ \mathcal{T}\beta = \beta \circ \alpha$. Bialgebra morphisms are both $\mathcal{T}$-algebra and $\mathcal{F}$-coalgebra morphisms.

Given a coalgebra $\beta \colon X \to \mathcal{F}\mathcal{T}X$ for a monad $(\mathcal{T}, \eta, \mu)$ and a functor $\mathcal{F}$, if there exists a distributive law $\lambda \colon \mathcal{T}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{T}$, one can form a coalgebra $\beta^\sharp \colon \mathcal{T}X \to \mathcal{F}\mathcal{T}X$ defined as $\mathcal{T}X \xrightarrow{\mathcal{T}\beta} \mathcal{T}\mathcal{F}\mathcal{T}X \xrightarrow{\lambda_{\mathcal{T}X}} \mathcal{F}\mathcal{T}\mathcal{T}X \xrightarrow{\mathcal{F}\mu} \mathcal{F}\mathcal{T}X$. Most importantly, $(\mathcal{T}X, \mu, \beta^\sharp)$ is a λ-bialgebra.
Free monads. We recall the construction of the monad $\mathcal{F}^\dagger \colon \mathcal{C} \to \mathcal{C}$ *freely generated* by a functor $\mathcal{F} \colon \mathcal{C} \to \mathcal{C}$. Assume that $\mathcal{C}$ has coproducts and that, for all objects X of $\mathcal{C}$, there exists an initial $X + \mathcal{F}$-algebra that we denote as $X + \mathcal{F}(\mathcal{F}^\dagger X) \xrightarrow{[\eta_X, \kappa_X]} \mathcal{F}^\dagger X$. It is easy to check that the assignment $X \mapsto \mathcal{F}^\dagger X$ induces a functor $\mathcal{F}^\dagger \colon \mathcal{C} \to \mathcal{C}$. The map $\eta_X \colon X \to \mathcal{F}^\dagger X$ gives rise to the unit of the monad; the multiplication $\mu_X \colon \mathcal{F}^\dagger \mathcal{F}^\dagger X \to \mathcal{F}^\dagger X$ is the unique algebra morphism from the initial $\mathcal{F}^\dagger X + \mathcal{F}$-algebra to the algebra $\mathcal{F}^\dagger X + \mathcal{F}(\mathcal{F}^\dagger X) \xrightarrow{[id, \kappa_X]} \mathcal{F}^\dagger X$.

GSOS specifications. An *abstract GSOS specification* is a natural transformation $\lambda \colon \mathcal{S}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}^\dagger$, where $\mathcal{F}$ is a functor representing the coalgebraic behaviour, $\mathcal{S}$ is a functor representing the syntax. It is important to recall the following fact.

▶ **Proposition 1** ([34]). *Any GSOS spec.* $\lambda \colon \mathcal{S}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}^\dagger$ *yields a distrib. law* $\lambda^\dagger \colon \mathcal{S}^\dagger\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}^\dagger$.

Coproduct of GSOS specifications. Suppose we have two functors $\mathcal{S}_1, \mathcal{S}_2 \colon \mathcal{C} \to \mathcal{C}$ capturing two syntaxes, a functor $\mathcal{F} \colon \mathcal{C} \to \mathcal{C}$ for the coalgebraic behaviour, and two GSOS specifications $\lambda_1 \colon \mathcal{S}_1\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_1^\dagger$ and $\lambda_2 \colon \mathcal{S}_2\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_2^\dagger$. Then we can construct a GSOS specification $\lambda_1 \cdot \lambda_2 \colon (\mathcal{S}_1 + \mathcal{S}_2)\mathcal{F} \Rightarrow \mathcal{F}(\mathcal{S}_1 + \mathcal{S}_2)^\dagger$. The details are in [7].

Quotients of monads and distributive laws. Given the correspondence between finitary monads and algebraic theories [29], it natural to consider *quotients* of monads by additional equations. Following [13, 15, 41], for a monad $\mathcal{T}$ on a category $\mathcal{C}$, $\mathcal{T}$*-equations* can be defined as a tuple $\mathbb{E} = (\mathcal{A}, l, r)$ consisting of a functor $\mathcal{A} \colon \mathcal{C} \to \mathcal{C}$ and natural transformations $l, r \colon \mathcal{A} \Rightarrow \mathcal{T}$. The intuition is that $\mathcal{A}$ acts as the variables of each equation, whose left- and right-hand sides are l and r, respectively. Assuming mild conditions that generalise the properties of Set (see [41, Ass. 7.1.2]), one constructs the *quotient* of $\mathcal{T}$ by $\mathcal{T}$-equations. The conditions hold in our setting: categories of presheaves over a discrete index category.

▶ **Proposition 2** (*cf.* [41])**.** *If $\mathcal{C} = \mathsf{Set}^{\mathcal{D}}$ for discrete $\mathcal{D}$, $\mathcal{T}$-equations $\mathbb{E}$ yield a monad $\mathcal{T}_{/\mathbb{E}} \colon \mathcal{C} \to \mathcal{C}$ with algebras precisely $\mathcal{T}$-algebras $\mathcal{T}A \xrightarrow{\alpha} A$ that satisfy $\mathbb{E}$, in the sense that $\alpha \circ l_A = \alpha \circ r_A$. Moreover, there exists a monad morphism $q^{\mathbb{E}} \colon \mathcal{T} \to \mathcal{T}_{/\mathbb{E}}$ with epi components.*

One may also quotient distributive laws, provided these are compatible with the new equations. Fix an endofunctor $\mathcal{F}$ and a monad $\mathcal{T}$ on $\mathsf{Set}^{\mathcal{D}}$, together with $\mathcal{T}$-equations $\mathbb{E} = (\mathcal{A}, l, r)$. We say that a distributive law $\lambda \colon \mathcal{T}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{T}$ *preserves equations* $\mathbb{E}$ if, for all $A \in \mathcal{C}$, the following diagram commutes: $\mathcal{A}\mathcal{F}A \underset{r_{\mathcal{F}A}}{\overset{l_{\mathcal{F}A}}{\rightrightarrows}} \mathcal{T}\mathcal{F}A \xrightarrow{\lambda_A} \mathcal{F}\mathcal{T}A \xrightarrow{\mathcal{F}q^{\mathbb{E}}_A} \mathcal{F}\mathcal{T}_{/\mathbb{E}}A$.

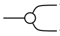
▶ **Proposition 3** (*cf.* [41])**.** *If $\lambda \colon \mathcal{T}\mathcal{F} \to \mathcal{F}\mathcal{T}$ preserves equations $\mathbb{E}$ then there exists a (unique) distributive law $\lambda_{/\mathbb{E}} \colon \mathcal{T}_{/\mathbb{E}}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{T}_{/\mathbb{E}}$ such that $\lambda_{/\mathbb{E}} \circ q^{\mathbb{E}}\mathcal{F} = \mathcal{F}q^{\mathbb{E}} \circ \lambda$.*

4 Diagrammatic Syntax as Monads

4.1 The Category of Signatures

Syntax and semantics of string diagrams will be specified in the category $\mathsf{Sig} := \mathsf{Span}(\mathsf{Set})(\mathbb{N}, \mathbb{N})$, where objects are spans $\mathbb{N} \leftarrow \Sigma \rightarrow \mathbb{N}$ in Set and arrows are span morphisms: given objects $\mathbb{N} \xleftarrow{s} X \xrightarrow{t} \mathbb{N}$ and $\mathbb{N} \xleftarrow{s'} \Sigma' \xrightarrow{t'} \mathbb{N}$, an arrow is a function $f \colon \Sigma \to \Sigma'$ such that $t' \circ f = t$ and $s' \circ f = s$. We think of an object of Sig as a *signature*, i.e. a set of symbols Σ equipped with arity and coarity functions $a, c \colon \Sigma \to \mathbb{N}$. We write $\Sigma(n, m)$ for the set $\{d \in \Sigma \mid \langle a, c\rangle(d) = (n, m)\}$ of operations with arity n and coarity m. Note that we allow coarities different from 1: this is because string diagrams express *monoidal* algebraic theories, not merely *cartesian* ones.

Since the objects in Sig are spans with identical domain and codomain, we will often write Σ for the entire span $\mathbb{N} \xleftarrow{a} \Sigma \xrightarrow{c} \mathbb{N}$. In particular, $\mathbb{N}$ means the identity span $\mathbb{N} \xleftarrow{id} \mathbb{N} \xrightarrow{id} \mathbb{N}$.

▶ **Example 4.** Recall the language $\mathsf{Circ}_{\mathsf{R}}$ from § 2. Line (2) of its syntax together with the first two lines of the sorting discipline in Fig. 1 define a signature Σ: every axiom $d : (n, m)$ gives the symbol d arity n and coarity m. For instance, $\Sigma(1, 2) = \{$, $\}$.

For computing (co)limits, it is useful to observe that Sig is isomorphic to the presheaf category $\mathsf{Set}^{\mathbb{N} \times \mathbb{N}}$, where $\mathbb{N} \times \mathbb{N}$ is the discrete category with objects pairs $(n, m) \in \mathbb{N} \times \mathbb{N}$.

4.2 Functors on Signatures

We turn to (co)algebras of endofunctors $\mathcal{F} \colon \mathsf{Sig} \to \mathsf{Sig}$ generated by the following grammar:

$$\mathcal{F} \quad ::= \quad Id \mid \Sigma \mid \mathbb{N} \mid \mathcal{F} ; \mathcal{F} \mid \mathcal{F} \oplus \mathcal{F} \mid \mathcal{F} + \mathcal{F} \mid \mathcal{F} \times \mathcal{F} \mid \overline{\mathcal{G}}$$

where $\mathcal{G}$ ranges over functors $\mathcal{G} : \mathsf{Set} \to \mathsf{Set}$ and Σ is a span $\mathbb{N} \leftarrow \Sigma \rightarrow \mathbb{N}$. In more detail:

- $Id\colon \mathsf{Sig} \to \mathsf{Sig}$ is the identity functor.
- $\Sigma\colon \mathsf{Sig} \to \mathsf{Sig}$ is the constant functor mapping every object to $\mathbb{N} \leftarrow \Sigma \to \mathbb{N}$ and every arrow to id_Σ; an important special case is $\mathbb{N}\colon \mathsf{Sig} \to \mathsf{Sig}$ the constant functor to $\mathbb{N} \xleftarrow{id} \mathbb{N} \xrightarrow{id} \mathbb{N}$.
- $(\cdot)\,;(\cdot)\colon \mathsf{Sig}^2 \to \mathsf{Sig}$ is *sequential composition* for signatures. On objects, $\Sigma_1\,;\Sigma_2$ is

$$\mathbb{N} \xleftarrow{s_1\circ\pi_1} \{(d_1,d_2) \in \Sigma_1 \times \Sigma_2 \mid t_1(d_1) = s_2(d_2)\} \xrightarrow{t_2\circ\pi_2} \mathbb{N}.$$

  Since the above is a Set-pullback, the action on arrows is inducted by the universal property. Note that, up to iso, $(\cdot)\,;(\cdot)\colon \mathsf{Sig}^2 \to \mathsf{Sig}$ is associative with unit $\mathbb{N}\colon \mathsf{Sig} \to \mathsf{Sig}$.
- $(\cdot)\oplus(\cdot)\colon \mathsf{Sig}^2 \to \mathsf{Sig}$ is *parallel composition* for signatures, with $\Sigma_1 \oplus \Sigma_2$ given by:

$$\mathbb{N} \xleftarrow{+\circ(s_1\times s_2)} \Sigma_1 \times \Sigma_2 \xrightarrow{+\circ(t_1\times t_2)} \mathbb{N}$$

  where $+\colon \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ is usual $\mathbb{N}$-addition. Again $(\cdot)\oplus(\cdot)\colon \mathsf{Sig}^2 \to \mathsf{Sig}$ associates up to iso.
- For the remaining functors, we use the fact that $\mathsf{Sig} \cong \mathsf{Set}^{\mathbb{N}\times\mathbb{N}}$, which guarantees (co)completeness, with limits and colimits constructed pointwise in Set. Thus, for spans Σ_1 and Σ_2, their coproduct is $\mathbb{N} \xleftarrow{[s_1,s_2]} \Sigma_1 + \Sigma_2 \xrightarrow{[t_1,t_2]} \mathbb{N}$ and the carrier of the product is $\{(d_1,d_2) \mid s_1(d_1) = s_2(d_2) \text{ and } t_1(d_1) = t_2(d_2)\}$, with the two obvious morphisms to $\mathbb{N}$.
- The isomorphism $\mathsf{Sig} \cong \mathsf{Set}^{\mathbb{N}\times\mathbb{N}}$ also yields the extension of an arbitrary endofunctor $\mathcal{G}\colon \mathsf{Set} \to \mathsf{Set}$ to a functor $\bar{\mathcal{G}}\colon \mathsf{Sig} \to \mathsf{Sig}$ defined by post-composition with $\mathcal{G}$, that is $\bar{\mathcal{G}}(\Sigma) = \mathcal{G}\circ\Sigma$ for all $\Sigma\colon \mathbb{N}\times\mathbb{N} \to \mathsf{Set}$. In particular, we shall often use the functor $\overline{\mathcal{P}_\kappa}$ obtained by post-composition with the κ-bounded powerset functor $\mathcal{P}_\kappa\colon \mathsf{Set} \to \mathsf{Set}$.[1]

Next we use these endofunctors to construct monads that capture the two-dimensional algebraic structure of string diagrams. In § 4.3 we construct the monad encoding the symmetric monoidal structure of props and in § 4.4 we construct the monad for the Frobenius structure of Carboni-Walters props. Later, in § 5, we shall use these monads to define compositional bialgebraic semantics for string diagrams of each of these categorical structures.

4.3 The Prop Monad

Here we define a monad on Sig with algebras precisely props: symmetric strict monoidal categories with objects the natural numbers, where the monoidal product on objects is addition. Together with identity-on-objects symmetric monoidal functors they form a category **PROP**. The first step is to encapsulate the operations of props as a Sig-endofunctor.

$$\mathcal{S}_{\mathsf{SM}} := (Id\,;Id) + \iota + (Id \oplus Id) + \epsilon + \sigma\colon \mathsf{Sig} \to \mathsf{Sig}. \tag{8}$$

In the type of $\mathcal{S}_{\mathsf{SM}}$, $Id\,;Id\colon \mathsf{Sig} \to \mathsf{Sig}$ is sequential composition and ι the identity arrow on object 1, i.e. the constant functor to $\mathbb{N} \xleftarrow{h} \{id_1\} \xrightarrow{h} \mathbb{N}$, with $h\colon id_1 \mapsto 1$. Similarly, $Id \oplus Id$ is the monoidal product with unit ϵ, i.e. the constant functor to $\mathbb{N} \xleftarrow{q} \{0\} \xrightarrow{q} \mathbb{N}$, with $q\colon 0 \mapsto 0$. Finally, σ is the basic symmetry: the constant functor to $\mathbb{N} \xleftarrow{f} \{\sigma_{1,1}\} \xrightarrow{f} \mathbb{N}$, with $f\colon \sigma_{1,1} \mapsto 2$.

The free monad $\mathcal{S}^\dagger_{\mathsf{SM}}$ on $\mathcal{S}_{\mathsf{SM}}$ is the functor mapping a span Σ to the span of Σ-terms obtained by sequential and parallel composition, together with symmetries and identities —with the identity id_n defined by parallel composition of n copies of id_1.

Algebras for this monad are spans Σ together with span morphisms *identity*: $\iota \to \Sigma$, *composition*: $\Sigma\,;\Sigma \to \Sigma$, *parallel*: $\Sigma \oplus \Sigma \to \Sigma$, *unit*: $\epsilon \to \Sigma$, and *swap*: $\sigma \to \Sigma$. This

[1] Boundedness is needed to ensure the existence of a final coalgebra, see § 5.1. In our leading example Circ_R, κ can be taken to be the cardinality of the semiring R.

information *almost* defines a prop $\mathcal{C}_\Sigma$: the carrier Σ of the span is the set of arrows of $\mathcal{C}_\Sigma$, containing special arrows id_n and $\sigma_{n,m}$ for identities and symmetries, *compose* assigns to every pairs of composable arrows their composition, and $\oplus$ assigns to every pair of arrows their monoidal product. The missing data is the usual equations (5)-(7) of symmetric monoidal categories. Thus, in order to obtain props as algebras, we quotient the monad $\mathcal{S}^{\dagger}_{\mathsf{SM}}$ by those equation, expressed abstractly as a triple $\mathbb{E}_{\mathsf{SM}} = (\mathcal{A}, l, r)$, as described in § 3. The functor $\mathcal{A}\colon \mathsf{Sig} \to \mathsf{Sig}$, defined below, has summands following the order (5)-(7):

$$(Id \oplus Id \oplus Id) + Id + Id + \sigma + ((Id\,;Id) \oplus (Id\,;Id)) + (Id\,;Id\,;Id) + Id + Id + Id^{+1} + Id^{+1} \quad (9)$$

Here, Id^{+1} is the functor adding 1 to the arity/coarity of each element of a given span $\mathbb{N} \xleftarrow{a} \Sigma \xrightarrow{c} \mathbb{N}$. We also need natural transformations $l, r\colon \mathcal{A} \to \mathcal{S}^{\dagger}_{\mathsf{SM}}$ that define the left- and right-hand side of each equation. For instance, for fixed $\Sigma \in \mathsf{Sig}$ and $(n, m) \in \mathbb{N} \times \mathbb{N}$:

- an element of $\Sigma\,;\Sigma\,;\Sigma$ (sixth summand of (9)) is a tuple (f, g, h) of Σ-elements, where f is of type (n, w), g of type (w, v), and h of type (v, m), for arbitrary $w, v \in \mathbb{N}$. We let l_Σ map (f, g, h) to the term $(f\,; g)\,; h$ of type (n, m) in $\mathcal{S}^{\dagger}_{\mathsf{SM}}(\Sigma)$, and r_Σ to the term $f\,; (g\,; h)$. Thus this component gives the second equation in (6) (associativity).
- the seventh summand Id in (9) yields a Σ-term f, which $l_\Sigma\colon \Sigma \to \mathcal{S}^{\dagger}_{\mathsf{SM}}(\Sigma)$ maps to $f\,; id_m$ and $r_\sigma\colon \Sigma \to \mathcal{S}^{\dagger}_{\mathsf{SM}}(\Sigma)$ maps to f, thus yielding the final equation in (6).
- an element in Σ^{+1} (last summand of (9)) of type $(n + 1, m + 1)$ is a Σ-term g of type (n, m), which is mapped by l_Σ to $(\sigma_{n,1}\,; (id_1 \oplus g))$ and by r_σ to $(g \oplus id_1)\,; \sigma_{m,1}$, both elements of $\mathcal{S}^{\dagger}_{\mathsf{SM}}(\Sigma)$ of type $(n + 1, m + 1)$, thus giving the final equation in (7).

The remainder of the definition of $l, r\colon \mathcal{A} \to \mathcal{S}^{\dagger}_{\mathsf{SM}}$, handles the remaining equations in (5)-(7), and should be clear from the above. Now, using Proposition 2, we quotient the monad $\mathcal{S}^{\dagger}_{\mathsf{SM}}$ by $(\mathcal{A}, l, r)$, obtaining a monad that we call $\mathcal{S}_{\mathsf{PROP}}$. We can then conclude by construction that the Eilenberg-Moore category $EM(\mathcal{S}_{\mathsf{PROP}})$ for the monad $\mathcal{S}_{\mathsf{PROP}}$ (with objects the $\mathcal{S}_{\mathsf{PROP}}$-algebras, and arrows the $\mathcal{S}_{\mathsf{PROP}}$-algebra homomorphisms) is precisely **PROP**.

▶ **Proposition 5.** *$EM(\mathcal{S}_{PROP}) \cong \mathbf{PROP}$.*

▶ **Example 6.** The monad $\mathcal{S}_{\mathsf{PROP}}$ takes Σ to the prop freely generated by Σ. Taking Σ as in Example 4, one obtains $\mathcal{S}_{\mathsf{PROP}}(\Sigma)$ with arrows $n \to m$ string diagrams of $\mathsf{Circ}_{\mathsf{R}}$ of sort $(n,\, m)$.

4.4 The Carboni-Walters Monad

The treatment we gave to props may be applied to other categorical structures. For space reasons, we only consider one additional such structure: *Carboni-Walters* (CW) props, also called 'hypergraph categories' [22]. Here each object n carries a distinguished special Frobenius bimonoid compatible with the monoidal product: it can be defined recursively using parallel compositions of the Frobenius structure on the generating object 1.

▶ **Definition 7.** *A* CW prop *is a prop with morphisms* $: 1 \to 2$, $: 1 \to 0$, $: 2 \to 1$, $: 0 \to 1$ *satisfying the equations of special Frobenius bimonoids (Fig. 4).*

CW props with prop morphisms preserving the Frobenius bimonoid form a subcategory **CW** of **PROP**. We can now extend the prop monad of § 4.3 to obtain a monad with algebras CW props. The signature is that of a prop with the additional Frobenius structure. Let $: \mathsf{Sig} \to \mathsf{Sig}$ be the functor constant at $\mathbb{N} \xleftarrow{s} \{\} \xrightarrow{t} \mathbb{N}$ with $s() = 1$ and $t() = 2$. Similarly, we introduce the constant functors $: \mathsf{Sig} \to \mathsf{Sig}$, $: \mathsf{Sig} \to \mathsf{Sig}$ and $: \mathsf{Sig} \to \mathsf{Sig}$ for the other generators. Let $\mathcal{S}_{\mathsf{FR}} := \mathcal{S}_{\mathsf{PROP}} + + + + $.

We now need to quotient $\mathcal{S}_{\mathsf{FR}}$ by the defining equations of special Frobenius bimonoids (Fig. 4). We omit the detailed encoding of these equations as a triple $\mathbb{E}_{\mathsf{CW}} = (\mathcal{A}_{\mathsf{CW}}, l_{\mathsf{CW}}, r_{\mathsf{CW}})$

since it presents no conceptual difficulty. Let $\mathcal{S}_{\mathtt{CW}}$ be the quotient of $\mathcal{S}_{\mathtt{FR}}$ by these equations. As for props, we obtain $EM(\mathcal{S}_{\mathtt{CW}}) \cong \mathbf{CW}$ by construction.

5 Bialgebraic Semantics for String Diagrams

Now that we have established monads for our categorical structures of interest, we study coalgebras that capture behaviour for string diagrams in these categories, and distributive laws that yield the desired bialgebraic semantics. We fix our 'behaviour' functor to

$$\mathcal{F} := \overline{\mathcal{P}_\kappa}(L\,;\,Id\,;\,L)\colon \mathsf{Sig} \to \mathsf{Sig}$$

where $L\colon \mathsf{Sig} \to \mathsf{Sig}$ is the *label functor* constant at the span $\mathbb{N} \xleftarrow{|\cdot|} A^* \xrightarrow{|\cdot|} \mathbb{N}$, with A^* the set of words on some set of labels A. The map $|\cdot|\colon A^* \to \mathbb{N}$ takes $w \in A^*$ to its length $|w| \in \mathbb{N}$. An $\mathcal{F}$-coalgebra is a span morphism $\Sigma \to \overline{\mathcal{P}_\kappa}(L\,;\,\Sigma\,;\,L)$; a function that takes $f \in \Sigma(n,m)$ to a set of transitions (v,g,w) with the appropriate sorts, i.e. $g \in \Sigma(n,m)$, $|v| = n$ and $|w| = m$.

The data of an $\mathcal{F}$-coalgebra $\beta\colon \Sigma \to \overline{\mathcal{P}_\kappa}(L\,;\,\Sigma\,;\,L)$ is that of a transition relation. For instance, fix labels $A = \{a,b\}$ and let $x,y \in \Sigma(1,2)$ and $z \in \Sigma(1,1)$; suppose also that β maps x to $\{(b\,;\,y\,;\,ab),(a\,;\,x\,;\,aa)\}$, y to $\emptyset$ and z to $\{(b\,;\,z\,;\,a)\}$. Then β can be written:

$$x \xrightarrow[ab]{b} y \qquad x \xrightarrow[aa]{a} x \qquad z \xrightarrow[a]{b} z \tag{10}$$

▶ **Example 8.** In our main example, Fig. 2 defines a coalgebra $\beta\colon \Sigma \to \overline{\mathcal{P}_\kappa}(L\,;\,\Sigma\,;\,L)$ where Σ is the signature from Example 4 and the set of labels is R. For instance $\beta(-\!\!\bullet) = \{(k, -\!\!\bullet, \varepsilon) \mid k \in \mathsf{R}\}$. Note the κ bounding $\mathcal{P}_\kappa$ is the cardinality of R.

In the sequel we shall construct distributive laws between the above behaviour functor and monads encoding the various categorical structures defined in the previous section.

5.1 Bialgebraic Semantics for Props

The modularity of $\mathcal{S}_{\mathtt{PROP}}$ can be exploited to define a distributive law of the $\mathcal{S}_{\mathtt{PROP}}$ over $\mathcal{F}$. Recall from § 4.3 that $\mathcal{S}_{\mathtt{PROP}}$ is a quotient of $\mathcal{S}^{\dagger}_{\mathtt{SM}}$. We start by letting $\mathcal{F} = \overline{\mathcal{P}_\kappa}(L\,;\,Id\,;\,L)$ interact with the individual summands of $\mathcal{S}_{\mathtt{SM}}$ (see (8)), corresponding to the operations of props. This amounts to defining GSOS specifications:

$$\lambda^{\mathsf{sq}}\colon \overline{\mathcal{P}_\kappa}(L\,;\,Id\,;\,L)\,;\,\overline{\mathcal{P}_\kappa}(L\,;\,Id\,;\,L) \Rightarrow \overline{\mathcal{P}_\kappa}(L\,;\,(Id\,;\,Id)^{\dagger}\,;\,L) \qquad \text{(sequential composition)}$$
$$\lambda^{\mathsf{id}}\colon \iota \Rightarrow \overline{\mathcal{P}_\kappa}(L\,;\,\iota^{\dagger}\,;\,L) \qquad \text{(identity)}$$
$$\lambda^{\mathsf{mp}}\colon \overline{\mathcal{P}_\kappa}(L\,;\,Id\,;\,L) \oplus \overline{\mathcal{P}_\kappa}(L\,;\,Id\,;\,L) \Rightarrow \overline{\mathcal{P}_\kappa}(L\,;\,(Id \oplus Id)^{\dagger}\,;\,L) \qquad \text{(monoidal product)}$$
$$\lambda^{\epsilon}\colon \epsilon \Rightarrow \overline{\mathcal{P}_\kappa}(L\,;\,\epsilon^{\dagger}\,;\,L) \qquad \text{(product unit)}$$
$$\lambda^{\mathsf{sy}}\colon \sigma \Rightarrow \overline{\mathcal{P}_\kappa}(L\,;\,\sigma^{\dagger}\,;\,L) \qquad \text{(symmetry)}$$

Definitions of these maps are succinctly given via derivation rules, see Fig. 3.

We explain this in detail for λ^{sq}, the others are similar. Given $\Sigma \in \mathsf{Sig}$, an element of type (n,m) in the domain $\overline{\mathcal{P}_\kappa}(L\,;\,\Sigma\,;\,L)\,;\,\overline{\mathcal{P}_\kappa}(L\,;\,\Sigma\,;\,L)$ is a pair (A,B), where, for some $z \in \mathbb{N}$,

$$A \text{ is a set of triples } (\boldsymbol{a}, c', \boldsymbol{b}) \in L(n,n) \times \Sigma(n,z) \times L(z,z), \text{ and}$$

$$B \text{ is a set of triples } (\boldsymbol{b}, d', \boldsymbol{c}) \in L(z,z) \times \Sigma(z,m) \times L(m,m).$$

Then $\lambda^{\mathsf{sq}}_{\Sigma}(A,B) := \{(\boldsymbol{a}, c'\,;\,d', \boldsymbol{c}) \mid (\boldsymbol{a}, c', \boldsymbol{b}) \in A,\ (\boldsymbol{b}, d', \boldsymbol{c}) \in B\}$. Following the convention (10), we can write this data as: $\boxed{\xrightarrow[c]{a} c'\,;\,d'} \in \lambda^{\mathsf{sq}}_{\Sigma}(A,B)$ if $\boxed{\xrightarrow[b]{a} c'} \in A$ and $\boxed{\xrightarrow[c]{b} d'} \in B$. This leads us to the more compact version of λ^{sq} as the transition rule in Fig.3.

Next, take the coproduct of GSOS specifications λ^{sq}, λ^{id}, λ^{mp}, λ^{ϵ} and λ^{sy} (see [7] for the details) to obtain $\lambda\colon \mathcal{S}_{\mathtt{SM}}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}^{\dagger}_{\mathtt{SM}}$. By Proposition 1, this yields distributive law $\lambda^{\dagger}\colon \mathcal{S}^{\dagger}_{\mathtt{SM}}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}^{\dagger}_{\mathtt{SM}}$.

The last step is to upgrade $\lambda^{\dagger}$ to a distributive law $\lambda^{\dagger}{}_{/\mathsf{SMC}}$ over the quotient $\mathcal{S}_{\mathtt{PROP}}$ of $\mathcal{S}^{\dagger}_{\mathtt{SM}}$ by the equations (5)-(7) of SMCs. By Proposition 3, this is well-defined if $\lambda^{\dagger}$ preserves $\mathbb{E}_{\mathsf{SM}}$. We show compatibility with associativity of sequential composition—the other equations can be verified similarly. This amounts to checking that if $\lambda^{\dagger}$ allows the derivation for $s_1 \,;\, (s_2 \,;\, s_3)$ as below left, then there exists a derivation for $(s_1 \,;\, s_2) \,;\, s_3$ as on the right, and vice-versa.

$$\frac{s_1 \xrightarrow[v]{u} s_1 \qquad \dfrac{s_2 \xrightarrow[w]{v} s_2 \quad s_3 \xrightarrow[x]{w} s_3}{s_2 \,;\, s_3 \xrightarrow[x]{v} s_2 \,;\, s_3}}{s_1 \,;\, (s_2 \,;\, s_3) \xrightarrow[x]{u} s_1 \,;\, (s_2 \,;\, s_3)} \qquad\qquad \frac{\dfrac{s_1 \xrightarrow[v]{u} s_1 \quad s_2 \xrightarrow[w]{v} s_2}{s_1 \,;\, s_2 \xrightarrow[w]{u} s_1 \,;\, s_2} \qquad s_3 \xrightarrow[x]{w} s_3}{(s_1 \,;\, s_2) \,;\, s_3 \xrightarrow[x]{u} (s_1 \,;\, s_2) \,;\, s_3}$$

By Proposition 3, we can therefore upgrade $\lambda^{\dagger}$ to a distributive law $\lambda^{\dagger}{}_{/\mathsf{SM}}\colon \mathcal{S}_{\mathtt{PROP}}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_{\mathtt{PROP}}$.

We are now ready to construct the compositional semantics as a morphism into the final coalgebra. One starts with a coalgebra $\beta\colon \Sigma \to \mathcal{F}(\mathcal{S}_{\mathtt{PROP}}(\Sigma))$ that describes the behaviour of Σ-operations, assigning to each a set of transitions, as in (10). The difference with (10) is that, because $\mathcal{F}$ is applied to $\mathcal{S}_{\mathtt{PROP}}(\Sigma)$ instead of just Σ, the right-hand side of each transition contains not just a Σ-operation, but a *string diagram*: a Σ-term modulo the laws of SMCs.

As recalled in § 3, using the distributive law $\lambda^{\dagger}{}_{/\mathsf{SM}}$ we can lift $\beta\colon \Sigma \to \mathcal{F}(\mathcal{S}_{\mathtt{PROP}}(\Sigma))$ to a $\lambda^{\dagger}{}_{/\mathsf{SM}}$-bialgebra, $\beta^{\sharp} : \mathcal{S}_{\mathtt{PROP}}(\Sigma) \to \mathcal{F}(\mathcal{S}_{\mathtt{PROP}}(\Sigma))$. Since this is a $\mathcal{F}$-coalgebra, the final $\mathcal{F}$-coalgebra Ω (the existence of which is shown in [7]) yields a semantics $[\![\cdot]\!]_{\beta}$ as below. The operational semantics of a string diagram c is $\beta^{\sharp}(c)$, obtained from *(i)* transitions for Σ-operations given by β and *(ii)* the derivation rules (Fig. 3) of $\lambda^{\dagger}{}_{/\mathsf{SM}}$. Instead, $[\![c]\!]_{\beta}$ is the observable behaviour: intuitively, its transition systems modulo bisimilarity.

$$\begin{tikzcd}
\mathcal{S}_{\mathtt{PROP}}(\Sigma) \arrow[r, dashed, "{[\![\cdot]\!]_{\beta}}"] \arrow[d, "\beta^{\sharp}"] & \Omega \arrow[d] \\
\mathcal{F}(\mathcal{S}_{\mathtt{PROP}}(\Sigma)) \arrow[r, "{\mathcal{F}([\![\cdot]\!]_{\beta})}"'] & \mathcal{F}(\Omega)
\end{tikzcd}$$

The bialgebraic semantics framework ensures that $\mathcal{S}_{\mathtt{PROP}}(\Sigma)$ and Ω are $\mathcal{S}_{\mathtt{PROP}}$-algebras, which by Proposition 5 are props. This means that the final coalgebra Ω is a prop and that $[\![\cdot]\!]_{\beta}$ is a prop morphism, preserving identities, symmetries and guaranteeing compositionality: $[\![s \,;\, t]\!]_{\beta} = [\![s]\!]_{\beta} \,;\, [\![t]\!]_{\beta}$ and $[\![s \oplus t]\!]_{\beta} = [\![s]\!]_{\beta} \oplus [\![t]\!]_{\beta}$.

▶ **Example 9.** Coming back to our running example, in Example 8 we showed that rules in Fig. 2 induce a coalgebra of type $\Sigma \to \mathcal{F}(\Sigma)$. Since each operation in Σ is itself a string diagram (formally, via the unit $\eta_{\Sigma}\colon \Sigma \to \mathcal{S}_{\mathtt{PROP}}(\Sigma)$), the same rules induce a coalgebra $\beta : \Sigma \to \mathcal{F}\mathcal{S}_{\mathtt{PROP}}(\Sigma)$, which has the type required for the above construction. The resulting coalgebra $\beta^{\sharp}\colon \mathcal{S}_{\mathtt{PROP}}(\Sigma) \to \mathcal{F}\mathcal{S}_{\mathtt{PROP}}(\Sigma)$ assigns to each diagram of $\mathcal{S}_{\mathtt{PROP}}(\Sigma)$ the set of transitions specified by the combined operational semantics of Figs. 2 and 3. The preceding discussion implies that, when e.g. $\mathsf{R} = \mathbb{N}$, bisimilarity for the Petri nets of [6] is a congruence.

5.2 Bialgebraic Semantics for Carboni-Walters Props

In this section we shall see two different ways of extending the GSOS specification of § 5.1 for CW props (see § 4.4). They correspond to the operational semantics of the black and white (co)monoids as given in Fig. 2. In the next section, we will see that these two different extensions give rise to two classic forms of synchronisation: à la Hoare and à la Milner.

Black distributive law. The first interprets the operations of the Frobenius structure as label synchronisation: from the black node derivations on the left of Fig. 2 we get

GSOS specifications given by natural transformations $\Rightarrow \mathcal{F}(\,^{\dagger})$, $\Rightarrow \mathcal{F}(\,^{\dagger})$, $\Rightarrow \mathcal{F}(\,^{\dagger})$, and $\Rightarrow \mathcal{F}(\,^{\dagger})$. Recall that, here, we use the diagrams to denote their associated functors $\mathsf{Sig} \to \mathsf{Sig}$. By taking the coproduct of these and λ, the GSOS specification for props from § 5.1, we obtain a specification $\lambda_{\bullet}$ for $\mathcal{S}_{\mathtt{FR}}$. It is straightforward to verify that $\lambda_{\bullet}^{\dagger} : \mathcal{S}_{\mathtt{FR}}^{\dagger}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_{\mathtt{FR}}^{\dagger}$ preserves the equations of special Frobenius bimonoids (Fig. 4), yielding a distributive law $\lambda_{\bullet/\mathsf{cw}}^{\dagger} : \mathcal{S}_{\mathtt{CW}}^{\dagger}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_{\mathtt{CW}}^{\dagger}$. As before, with $\lambda_{\bullet/\mathsf{cw}}^{\dagger}$ we obtain a bialgebra $\beta_{\bullet}^{\sharp} \colon \mathcal{S}_{\mathtt{CW}}(\Sigma) \to \mathcal{F}\mathcal{S}_{\mathtt{CW}}(\Sigma)$ from any coalgebra $\beta \colon \Sigma \to \mathcal{F}\mathcal{S}_{\mathtt{CW}}(\Sigma)$.

White distributive law. When the set of labels A is an *Abelian group*, it is possible to give a different GSOS specifications for the Frobenius structure, capturing the group operation of A: from the white node derivations on the right of Fig. 2 we get GSOS specifications $\Rightarrow \mathcal{F}(\,^{\dagger})$, $\Rightarrow \mathcal{F}(\,^{\dagger})$, $\Rightarrow \mathcal{F}(\,^{\dagger})$, and $\Rightarrow \mathcal{F}(\,^{\dagger})$. Using a now familiar procedure we obtain a GSOS specifications $\lambda_{\circ}$ for $\mathcal{S}_{\mathtt{FR}}$. The group structure on A guarantees [39] that $\lambda_{\circ}^{\dagger} : \mathcal{S}_{\mathtt{FR}}^{\dagger}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_{\mathtt{FR}}^{\dagger}$ preserves the equations of special Frobenius bimonoids (Fig. 4). Therefore we get a distributive law $\lambda_{\circ/\mathsf{cw}}^{\dagger} : \mathcal{S}_{\mathtt{CW}}^{\dagger}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_{\mathtt{CW}}^{\dagger}$.

▶ Remark 10. Given the results in this section, one could ask if bialgebraic semantics works for *any* categorical structure. A notable case in which it fails is that of Lawvere theories [29]. These can be seen as props with a *natural* comonoid structure on each object [12]. One may define a monad for Lawvere theories following the same recipe as above. However, it turns out that this monad is incompatible with the GSOS specification for the comonoid given in Fig. 2. To see the problem consider a term d that can perform two transitions nondeterministically: $d \xrightarrow[a]{\epsilon} d$ and $d \xrightarrow[b]{\epsilon} d$. The naturality of the comonoid forces $d\,;\, \approx d \oplus d$ but $d\,;\,$ can only perform the $a\,a$ and $b\,b$ transitions while $d \oplus d$ can also perform $a\,b$ or $b\,a$. Thus the specification would not be compositional. For more details, we refer the reader to the appendix of [7].

6 Black and White Frobenius as Hoare and Milner Synchronisation

The role of this section is twofold: on the one hand we demonstrate how classical process calculus syntax benefits from a string diagrammatic treatment; on the other we draw attention towards a surprising observation, namely that the black and white Frobenius structures discussed previously provide the synchronisation mechanism of, respectively, CSP and CCS.

6.1 Syntax

We consider a minimal process calculus for simplicity. Assume a countable set $\mathcal{N}$ of *names*, a_1, a_2, ...and a set $\mathcal{V}$ of *process variables*, $\mathtt{f}$, $\mathtt{g}$, ..., equipped with a function $ar \colon \mathcal{V} \to \mathbb{N}$ that assigns the set of names that the process may use: $ar(\mathtt{f}) = n$ means that the process $\mathtt{f}$ uses only names $\{a_1, \ldots, a_n\}$. This is Hoare's [28] notion of *alphabet* for process variables.

Roughly speaking, in a string diagram, dangling wires perform the job of variables. To ease the translation of terms to diagrams, we include permutations of names in the syntax, hereafter denoted by σ. For a permutation $\sigma \colon \mathcal{N} \to \mathcal{N}$, its support is the set $supp(\sigma) = \{a_i \mid a_i \neq \sigma(a_i)\}$; σ is *finitely supported* if $supp(\sigma)$ is finite. For each finitely supported permutation σ its *degree* is defined as the greatest $i \in \mathbb{N}$ such that $a_i \in supp(\sigma)$.

The set of processes is defined recursively as follows

$$P := P|P, \;\; \nu a_i(P), \;\; \mathtt{f}, \;\; P\sigma$$

where $a_i \in \mathcal{N}$, $\mathtt{f} \in \mathcal{V}$ and σ is a finitely supported permutation of names. The symbol $|$ stands for the parallel composition of processes. The symbol νa_i stands for the restriction, or hiding, of the name a_i. Observe that there are no primitives for prefixes, non-deterministic choice or recursion: these will appear in the declaration of process variables which we will describe in § 6.2. The idea here is to separate the behaviour, specified in the declaration of process variables, and the communication topology of the network, given by the syntax above. The notion of alphabet can be defined for all processes as follows:

$$al(P|Q) = al(P) \cup al(Q) \quad al(\nu a_i(P)) = al(P) \setminus \{a_i\} \quad al(\mathtt{f}) = \{a_1, \dots, a_{ar(\mathtt{f})}\} \quad al(P\sigma) = \sigma[al(P)]$$

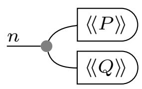
From one-dimensional to two-dimensional syntax. We use a typing discipline to guide the translation of terms to string diagrams:

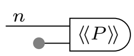
$$\frac{n \vdash P \quad n \vdash Q}{n \vdash P|Q} \qquad \frac{n+1 \vdash P}{n \vdash \nu a_{n+1}(P)} \qquad \frac{ar(\mathtt{f}) = n}{n \vdash \mathtt{f}} \qquad \frac{n \vdash P \quad degree(\sigma) \leq n}{n \vdash P\sigma} \qquad \frac{n \vdash P}{n+1 \vdash P} \tag{11}$$

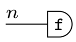
The meaning of the types is explained by the following lemma, easily proven by induction.

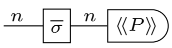
▶ **Lemma 11.** *If $n \vdash P$ then $al(P) \subseteq \{a_1, \dots a_n\}$.*

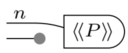
We will translate processes to the CW prop freely generated from $\Sigma = \{\mathtt{f} \colon (n, 0) \mid \mathtt{f} \in \mathcal{V} \text{ and } ar(\mathtt{f}) = n\}$; in particular a typed process $n \vdash P$ results in a string diagram of $\mathcal{S}_{\mathtt{CW}}(\Sigma)(n, 0)$. The translation $\langle\langle \cdot \rangle\rangle$ is defined recursively on typed terms as follows:

$\langle\langle n \vdash P|Q \rangle\rangle =$ 

$\langle\langle n \vdash \nu a_{n+1}(P) \rangle\rangle =$ 

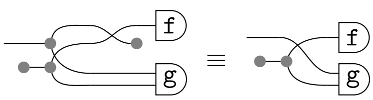
$\langle\langle n \vdash \mathtt{f} \rangle\rangle =$ 

$\langle\langle n \vdash P\sigma \rangle\rangle =$ 

$\langle\langle n+1 \vdash P \rangle\rangle =$ 

where for σ with $degree(\sigma) < n$, $\overline{\sigma} \colon n \to n$ is the obvious corresponding arrow in $\mathcal{S}_{\mathtt{CW}}(\Sigma)$.

▶ **Example 12.** Let $\mathcal{V} = \{\mathtt{f}, \mathtt{g}\}$ with $ar(\mathtt{f}) = 1$ and $ar(\mathtt{g}) = 2$. Let $[a_2/a_1] \colon \mathcal{N} \to \mathcal{N}$ be the permutation swapping a_1 and a_2. One can easily check that $1 \vdash \nu a_2(\mathtt{f}[a_2/a_1] \,|\, \mathtt{g})$. Then $\langle\langle 1 \vdash \nu a_2(\mathtt{f}[a_2/a_1] \,|\, \mathtt{g}) \rangle\rangle$ is as on the right.



6.2 Semantics

In order to give semantics to the calculus, we assume a set $\mathcal{A}$ of actions, α, β, ... Since, we will consider different sets of actions (for Hoare and Milner synchronisation), we assume them to be functions of type $\mathcal{N} \to M$ for some monoid $(M, +, 0)$. The support of an action α is the set $\{a_i \mid \alpha(a_i) \neq 0\}$. The alphabet of α, written $al(\alpha)$ is identified with its support.

For Hoare synchronisation, the monoid M is $(2, \cup, 0)$, while for Milner it is $(\mathbb{Z}, +, 0)$. In both cases, we will write a_i for the function mapping the name a_i to 1 and all the others to 0. For Milner synchronisation, write $\overline{a_i}$ for the function mapping a_i to -1.

To give semantics to processes, we need a *process declaration* for each $\mathtt{f} \in \mathcal{V}$. That is, an expression $\mathtt{f} := \sum_{i \in I} \alpha_i.P_i$, for some finite set I, $\alpha_i \in \mathcal{A}$ and processes P_i such that

$$\{a_1, \dots a_{ar(\mathtt{f})}\} \subseteq \bigcup_{i \in I} al(\alpha_i) \cup \bigcup_{i \in I} al(P_i) \tag{12}$$

The basic behaviour of process declarations is captured by the three rules below.

$$\frac{}{\mathtt{f} \xrightarrow{0} \mathtt{f}} \qquad \frac{\mathtt{f} := \sum_{i \in I} \alpha_i.P_i}{\mathtt{f} \xrightarrow{\alpha_i} P_i} \qquad \frac{P \xrightarrow{\alpha} P'}{P\sigma \xrightarrow{\alpha \circ \sigma} P'\sigma} \tag{13}$$

▶ **Example 13.** Recall $\mathtt{f}$ and $\mathtt{g}$ from Example 12. Assume declarations $\mathtt{f}{:=}a_1.\nu a_2(\mathtt{f}[a_2/a_1]\,|\,\mathtt{g})$ and $\mathtt{g}{:=}a_1.\mathtt{g}+a_2.\mathtt{g}$. Observe that they respect (12). We have that $\mathtt{g} \xrightarrow{a_1} \mathtt{g}$ and $\mathtt{g} \xrightarrow{a_2} \mathtt{g}$ while $\mathtt{f} \xrightarrow{a_1} \nu a_2(\mathtt{f}[a_2/a_1]\,|\,\mathtt{g})$. Similarly $\mathtt{f}[a_2/a_1] \xrightarrow{a_2} (\nu a_2(\mathtt{f}[a_2/a_1]\,|\,\mathtt{g}))[a_2/a_1]$.

To define the semantics of parallel and restriction, we need to distinguish between the Hoare and Milner synchronisation patterns.

Hoare synchronisation. Here actions are functions $\alpha\colon \mathcal{N} \to 2$, which can equivalently be thought of as subsets of $\mathcal{N}$. The synchronisation mechanism presented below is analogous to the one used in CSP [28]. The main difference is the level of concurrency: the classical semantics [28] is purely interleaving, while for us it is a step semantics. Essentially, in $P|Q$, the processes P and Q may evolve independently on the non-shared names, i.e. the evolution of two or more processes may happen at the same time. It is for this reason that our actions are sets of names. The operational semantics of parallel and restriction is given by rules

$$\frac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\beta} Q' \quad \alpha \cap al(Q) = \beta \cap al(P)}{P|Q \xrightarrow{\alpha\cup\beta} P'|Q'} \qquad \frac{P \xrightarrow{\alpha} P'}{\nu a_i(P) \xrightarrow{\alpha\backslash\{a_i\}} \nu a_i(P')} \tag{14}$$

We write $\xrightarrow{\alpha}_H$ for the transition systems generated by the rules (13), (14). By a simple inductive argument, using (12) as base case, we see that for all processes P, if $P \xrightarrow{\alpha} P'$ then $\alpha \subseteq al(P)$. The rule for parallel, therefore, ensures that P and Q synchronise over all of their shared names. The rule for restriction hides a_i from the environment. For instance, if $\alpha = \{a_i\}$, then $\nu a_i(P) \xrightarrow{\emptyset} \nu a_i(P')$. If $\alpha = \{a_j\}$ with $a_j \neq a_i$, then $\nu a_i(P) \xrightarrow{\{a_j\}} \nu a_i(P')$.

▶ **Example 14.** Recall $\mathtt{f}$ and $\mathtt{g}$ from Example 13. We have that $\mathtt{f} \xrightarrow{a_1}_H \nu a_2(\mathtt{f}[a_2/a_1]\,|\,\mathtt{g})$. From $\nu a_2(\mathtt{f}[a_2/a_1]\,|\,\mathtt{g})$, there are two possibilities: either $\mathtt{f}[a_2/a_1]$ and $\mathtt{g}$ synchronise on a_2, and in this case we have $\nu a_2(\mathtt{f}[a_2/a_1]\,|\,\mathtt{g}) \xrightarrow{\emptyset}$, or $\mathtt{g}$ proceeds without synchronising on a_1, therefore $\nu a_2(\mathtt{f}[a_2/a_1]\,|\,\mathtt{g}) \xrightarrow{\{a_1\}}_H$ since a_1 belongs to $al(\mathtt{g})$ and not to $al(\mathtt{f}[a_2/a_1])$.

Milner synchronisation. We take $\mathcal{A} = \mathbb{Z}^{\mathcal{N}}$. Sum of functions, denoted by $+$, is defined pointwise and we write 0 for its unit, the constant 0 function.

$$\frac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\beta} Q'}{P|Q \xrightarrow{\alpha+\beta} P'|Q'} \qquad \frac{P \xrightarrow{\alpha} P'}{\nu a_i(P) \xrightarrow{\alpha} \nu a_i(P')}\ \alpha(a_i) = 0 \tag{15}$$

We write $\xrightarrow{\alpha}_M$ for the transition system generated by the rules (13), (15).

Functions in $\mathbb{Z}^{\mathcal{N}}$ to represent concurrent occurrences of CCS send and receive actions. A single CCS action a is the function mapping a to 1 and all other names to 0. Similarly, the action $\bar{a}$ maps a to -1 and the other names to 0. The silent action τ is the function 0. With this in mind, it is easy to see that, similarly to CCS, the rightmost rule forbids $\nu a_i(P) \xrightarrow{\alpha} \nu a_i(P')$ whenever $\alpha = a_i$ or $\alpha = \bar{a}_i$. CCS-like synchronisation is obtained by the leftmost rule: when $\alpha = a_i$ and $\beta = \bar{a}_i$, one has that $P|Q \xrightarrow{0} P'|Q'$.

A simple inductive argument confirms that $P \xrightarrow{0} P$ for any process P. Then, by the leftmost rule again, one has that whenever $Q \xrightarrow{\beta} Q'$, then $P|Q \xrightarrow{\beta} P|Q'$. Note, however, that as in § 6.2, while our synchronisation mechanism is essentially Milner's CSS handshake, our semantics is not interleaving and allows for step concurrency. It is worth remarking that the operational rules in (15) have already been studied by Milner in its work on SCCS [37].

Semantic correspondence. For an action $\alpha\colon \mathcal{N} \to M$ with $al(\alpha) \subseteq \{a_1, \ldots a_n\}$, we write $n \vdash \alpha$ for the restriction $\{a_1, \ldots, a_n\} \to M$. Define coalgebras $\beta_b, \beta_w \colon \Sigma \to \overline{\mathcal{P}_\kappa}(L\,;\,\mathcal{S}_{\mathtt{CW}}(\Sigma)\,;\,L)$

for each $\mathtt{f} \in \Sigma_{n,0}$ where $\mathtt{f} := \sum_{i \in I} \alpha_i.P_i$ as

$$\beta_b(\mathtt{f}) = \beta_w(\mathtt{f}) = \big\{\big(n \vdash \alpha_i, \langle\!\langle P_i \rangle\!\rangle, \bullet\big) \mid i \in I\big\} \cup \big\{\big(n \vdash 0, \mathtt{f}, \bullet\big)\big\}\,.$$

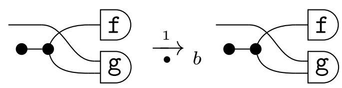
For both β_b and β_w, L is the span $\mathbb{N} \xleftarrow{|\cdot|} A^* \xrightarrow{|\cdot|} \mathbb{N}$, but $A = 2$ for β_b and $A = \mathbb{Z}$ for β_w.

Via the distributive law (§ 5.2) for the black Frobenius, we obtain the coalgebra $\beta_b^\sharp \colon \mathcal{S}_{\mathtt{CW}}(\Sigma) \to \overline{\mathcal{P}_\kappa}(L\,;\mathcal{S}_{\mathtt{CW}}(\Sigma)\,;L)$. Via the white Frobenius, we obtain $\beta_w^\sharp \colon \mathcal{S}_{\mathtt{CW}}(\Sigma) \to \overline{\mathcal{P}_\kappa}(L\,;\mathcal{S}_{\mathtt{CW}}(\Sigma)\,;L)$. We write $c \xrightarrow[\alpha]{\beta}{}_b\, d$ for $(\alpha,\beta,d) \in \beta_b^\sharp(c)$ and $c \xrightarrow[\alpha]{\beta}{}_w\, d$ for $(\alpha,\beta,d) \in \beta_w^\sharp(c)$. The correspondence can now be stated formally.

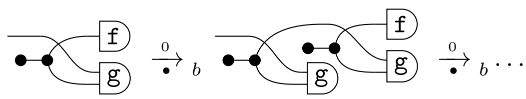
▶ **Theorem 15.** *Let $n \vdash P$ and $n \vdash \alpha$ such that $al(\alpha) \subseteq al(P)$.*

- **Hoare is black.** *If $P \xrightarrow{\alpha}_H P'$ then $\overset{n}{—}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha}{}_b \overset{n}{—}\boxed{\langle\!\langle P' \rangle\!\rangle}$. Vice versa, if $\overset{n}{—}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha}{}_t \overset{n}{—}\boxed{d}$ then there is $n \vdash P'$ s.t. $P \xrightarrow{\alpha}_H P'$ and $\overset{n}{—}\boxed{\langle\!\langle P' \rangle\!\rangle} = \overset{n}{—}\boxed{d}$.*
- **Milner is white.** *If $P \xrightarrow{\alpha}_M P'$ then $\overset{n}{—}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha}{}_w \overset{n}{—}\boxed{\langle\!\langle P' \rangle\!\rangle}$. Vice versa, if $\overset{n}{—}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha}{}_w \overset{n}{—}\boxed{d}$ then there is $n \vdash P'$ s.t. $P \xrightarrow{\alpha}_M P'$ and $\overset{n}{—}\boxed{\langle\!\langle P' \rangle\!\rangle} = \overset{n}{—}\boxed{d}$.*

▶ **Example 16.** We illustrate the semantic correspondence by returning to Example 13. Diagrammatically, it yields the following transitions:



and



7 Related and Future Work

The terminology *Hoare and Milner synchronisation* is used in Synchronised Hyperedge Replacement (SHR) [20, 33]. Our work is closely related to SHR: indeed, the prop $\mathcal{S}_{\mathtt{CW}}(\Sigma)$ has arrows open hypergraphs, where hyperedges are labeled with elements of Σ [5]. To define a coalgebra $\beta \colon \Sigma \to \mathcal{F}\mathcal{S}_{\mathtt{CW}}(\Sigma)$ is to specify a transition system for each label in Σ. Then, constructing the coalgebra $\beta^\sharp \colon \mathcal{S}_{\mathtt{CW}}(\Sigma) \to \mathcal{F}\mathcal{S}_{\mathtt{CW}}(\Sigma)$ from a distributive law amounts to giving a transition system to all hypergraphs according to some synchronisation policy (e.g. à la Hoare or à la Milner). SHR systems equipped with Hoare and Milner synchronisation are therefore instances of our approach. A major difference is our focus on the algebraic aspects: e.g. since string diagrams can be regarded as syntax as well as combinatorial entities, their syntactic nature allows for the bialgebraic approach, and simple inductive proofs. The operational rules in Figure 3 are also those of tile systems [23]. However, in the context of tiles, transitions are arrows of the vertical *category*: this forces every state to perform at least one identity transition. For example, it is not possible to consider empty sets of transitions, which can be a useful feature in the string diagrammatic approach, see [8].

Amongst the many other related models, it is worth mentioning bigraphs [38]. While also graphical, bigraphs can be nested hierarchically, a capability that we have not considered. Moreover, the behaviour functor $\mathcal{F}$ in § 5 forces the labels and the arriving states to have the same sort as the starting states. Therefore, fundamental mobility mechanisms such as scope-extrusion cannot immediately be addressed within our framework. We are confident, however, that the solid algebraic foundation we have laid here for the operational semantics of two-dimensional syntax will be needed to shed light on such concepts as hierarchical composition and mobility. Some ideas may come from [14].

References

1. Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols. In *LICS 2004*, pages 415–425, 2004.
2. John Baez and Jason Erbele. Categories in control. *Theory Appl. Categ.*, 30:836–881, 2015.
3. John Baez and Brendan Fong. A compositional framework for passive linear circuits. *J. Complex Netw.*, 54:5, 2015.
4. Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.
5. Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Paweł Sobociński, and Fabio Zanasi. Rewriting modulo symmetric monoidal structure. In *LICS 2016*, pages 1–10, 2016.
6. Filippo Bonchi, Joshua Holland, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. Diagrammatic algebra: from linear to concurrent systems. In *POPL 2019*, 2019.
7. Filippo Bonchi, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. Bialgebraic semantics for string diagrams, 2019. `arXiv:1906.01519`.
8. Filippo Bonchi, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. Graphical affine algebra. In *To appear in LICS 2019*, 2019.
9. Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. A categorical semantics of signal flow graphs. In *CONCUR 2014*, pages 435–450, 2014.
10. Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. Full abstraction for signal flow graphs. In *POPL 2015*, pages 515–526, 2015.
11. Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. The calculus of signal flow diagrams I: linear relations on streams. *Inf. Comput.*, 252:2–29, 2017.
12. Filippo Bonchi, Pawel Sobociński, and Fabio Zanasi. Deconstructing Lawvere with distributive laws. *J. Log. Algebr. Meth. Program.*, 95:128–146, 2018.
13. Marcello M. Bonsangue, Helle Hvid Hansen, Alexander Kurz, and Jurriaan Rot. Presenting distributive laws. In *Algebra and Coalgebra in Computer Science - 5th International Conference, CALCO 2013, Warsaw, Poland, September 3-6, 2013. Proceedings*, pages 95–109, 2013.
14. Roberto Bruni, Ugo Montanari, Gordon D. Plotkin, and Daniele Terreni. On hierarchical graphs: reconciling bigraphs, gs-monoidal theories and gs-graphs. *Fundam. Inform.*, 134(3-4):287–317, 2014.
15. Maria Grazia Buscemi and Ugo Montanari. A first order coalgebraic model of pi-calculus early observational equivalence. In Lubos Brim, Petr Jancar, Mojmír Kretínský, and Antonín Kucera, editors, *CONCUR 2002 - Concurrency Theory, 13th International Conference, Brno, Czech Republic, August 20-23, 2002, Proceedings*, volume 2421 of *Lecture Notes in Computer Science*, pages 449–465. Springer, 2002. URL: `https://doi.org/10.1007/3-540-45694-5_30`, `doi:10.1007/3-540-45694-5\_30`.
16. Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theor. Comput. Sci.*, 240(1):177–213, 2000.
17. Giuseppe Castagna, Jan Vitek, and Francesco Zappa Nardelli. The seal calculus. *Inform. Comput.*, 201(1):1–54, 2005.
18. Bob Coecke and Ross Duncan. Interacting quantum observables. In *ICALP 2008*, pages 298–310, 2008.
19. Robert De Simone. Higher-level synchronising devices in Meije-SCCS. *Theor. Comput. Sci.*, 37:245–267, 1985.
20. Pierpaolo Degano and Ugo Montanari. A model for distributed systems based on graph rewriting. *J. ACM*, 34(2):411–449, 1987.
21. Brendan Fong, Paolo Rapisarda, and Paweł Sobociński. A categorical approach to open and interconnected dynamical systems. In *LICS 2016*, pages 1–10, 2016.
22. Brendan Fong and David I. Spivak. Hypergraph categories. *CoRR*, abs/1806.08304, 2018.
23. Fabio Gadducci and Ugo Montanari. The tile model. In *Proof, Language and Interaction: Essays in Honour of Robin Milner*, pages 133–166. MIT Press, 2000.

24 Dan R. Ghica. Diagrammatic reasoning for delay-insensitive asynchronous circuits. In *Abramsky Festschrift*, pages 52–68, 2013.

25 Jan Friso Groote. Transition system specifications with negative premises. *Theor. Comput. Sci.*, 118(2):263–299, 1993.

26 David Harel. Statecharts: A visual formalism for complex systems. *Sci. Comput. Program.*, 8(3):231–274, 1987.

27 C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, August 1978.

28 C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, Inc., 1985.

29 Martin Hyland and John Power. Lawvere theories and monads. In *Computation, Meaning, and Logic*, pages 437–458, 2007.

30 Kurt Jensen. *Coloured Petri nets: basic concepts, analysis methods and practical use*, volume 1. Springer Science & Business Media, 2013.

31 Piergiulio Katis, Nicoletta Sabadini, and Robert Frank Carslaw Walters. Span(Graph): an algebra of transition systems. In *AMAST 1997*, volume 1349 of *LNCS*, 1997.

32 Bartek Klin. Bialgebras for structural operational semantics: an introduction. *Theor. Comput. Sci.*, 412(38):5043–5069, 2011.

33 Ivan Lanese and Ugo Montanari. Hoare vs Milner: Comparing synchronizations in a graphical framework with mobility. *Electr. Notes Theor. Comput. Sci.*, 154(2):55–72, 2006.

34 Marina Lenisa, John Power, and Hiroshi Watanabe. Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads. *Electr. Notes Theor. Comput. Sci.*, 33:230 – 260, 2000.

35 Saunders Mac Lane. Categorical algebra. *B. Am. Math. Soc.*, 71:40–106, 1965.

36 Robin Milner. *A Calculus of Communicating Systems.* Springer-Verlag, 1982.

37 Robin Milner. Calculi for synchrony and asynchrony. *Theor. Comput. Sci.*, 25(3):267–310, 1983.

38 Robin Milner. *The space and motion of communicating agents.* Cambridge University Press, 2009.

39 Dusko Pavlovic. Quantum and classical structures in nondeterministic computation. In *QI 2009*, pages 143–157, 2009.

40 Wolfgang Reisig. *Petri nets: an introduction*, volume 4. Springer Science & Business Media, 2012.

41 Jurriaan Rot. *Enhanced Coinduction.* PhD thesis, University of Leiden, 2015.

42 Davide Sangiorgi and David Walker. *PI-Calculus: A Theory of Mobile Processes.* Cambridge University Press, 2001.

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

44 Paweł Sobociński. A non-interleaving process calculus for multi-party synchronisation. In *ICE 2009*, pages 87–98, 2009.

45 Daniele Turi and Gordon Plotkin. Towards a mathematical operational semantics. In *LICS 1997*, pages 280–291, 1997.