Bialgebraic Foundations for the Operational Semantics of String Diagrams

Filippo Bonchi[1], Robin Piedeleu[1], Paweł Sobociński[1,], Fabio Zanasi[1]

[a] Università di Pisa, Italy
[b] University College London, United Kingdom
[c] Tallinn University of Technology, Estonia

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.

Keywords: String Diagram, Structural Operational Semantics, Bialgebraic semantics

1. Introduction

Starting from the seminal works of Hoare and Milner, there was an explosion [**? ? ? ? ?**] 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

[*]Supported by the ESF funded Estonian IT Academy research measure (project 2014-2020.4.05.19-0001)

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 [? ? ?]; succinctly stated, syntactic operations with semantics defined using such formats are homomorphic wrt the semantic space of behaviours.

Another thread of concurrency theory research [? ? ?] 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 [?] and CSP [?] often resort to pictures that give intuition about topological aspects of syntactic specifications. However, compositionality has not—historically—been a principal focus of research.

In this paper we propose a framework that allows us to eat our cake and have it too. We use *string diagrams* [?] 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[? ?], signal flow graphs [? ? ?], Petri nets [?], and electrical circuits [? ?], amongst several other applications.

Our main contribution is the adaptation of Turi and Plotkin's bialgebraic semantics (*abstract GSOS*) [? ?] 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) [?], tile logic [?], the wire calculus [?] and recent work on modelling signal flow graphs and Petri nets [? ?]. 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 [?], 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.

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

The crucial observation is that (??) 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. $\llbracket s + t\rrbracket = \llbracket s\rrbracket + \llbracket t\rrbracket$, 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 (??), 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.

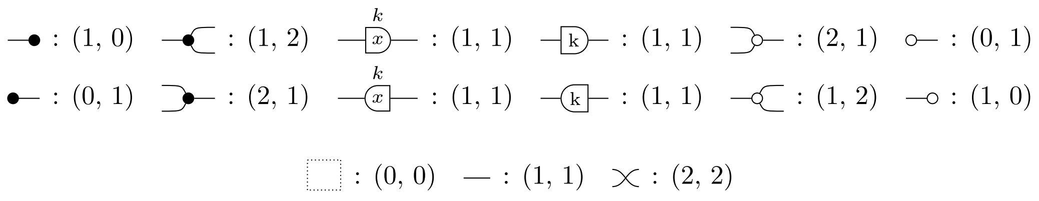
Adapting the bialgebraic approach to the 2-dimensional syntax of props requires some technical work since, e.g. the composition operation of monoidal categories is a dependent operation. For this reason we need to adapt the usual notion of a syntax endofunctor on the category of sets; instead we work in a category Sig whose objects are spans $\mathbb{N} \leftarrow \Sigma \rightarrow \mathbb{N}$, 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 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 [?], and one which is a compositional account of Petri nets [?].

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 [?] and Milner's CCS [?].

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

This work extends the conference paper [?] with a greatly expanded section on preliminaries (Section ??), ??, and all the missing proofs. Moreover, Section ??, Remark ?? and Section ?? are new contributions.



$$\frac{c : (k_1, k_2) \quad d : (k_2, k_3)}{c \,;\, d : (k_1, k_3)} \qquad \frac{c : (k_1, l_1) \quad d : (k_2, l_2)}{c \oplus d : (k_1{+}k_2, l_1{+}l_2)}$$

Figure 1: Sorting discipline for Circ_R

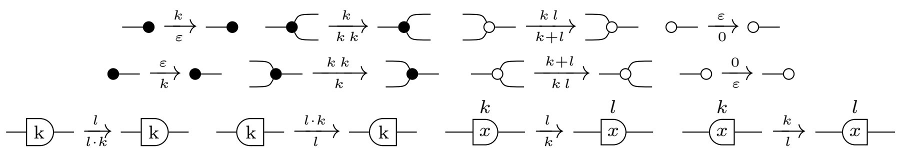
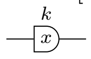
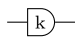
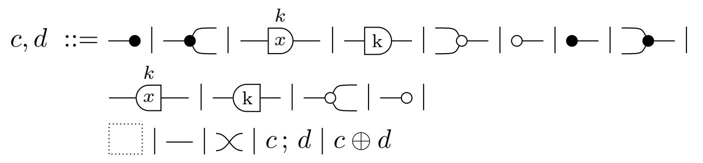


Figure 2: Structural Operational Semantics for the generators of Circ_R. Intuitively, from left to right, these are elementary connectors modelling discard, copy, one-place register, multiplication by a scalar, addition, and the constant zero.

2. Motivating Example

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



(2)

(3)

(4)

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 **??**. An easy induction confirms uniqueness of sorting.

The operational meaning of terms is defined recursively by the structural rules in Figs. **??** and **??** 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 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.

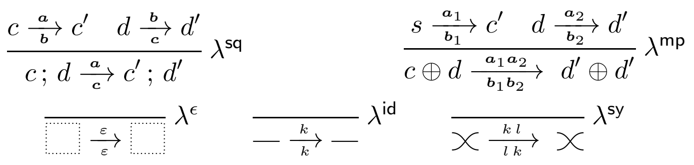
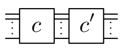
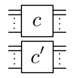
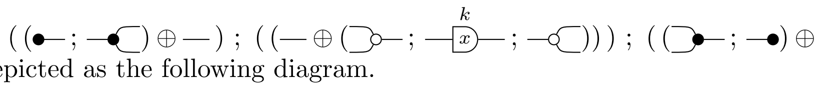
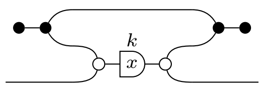


Figure 3: Structural operational semantics for the operations of Circ_R.

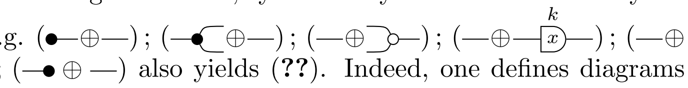
Our chief focus in this paper is the study of semantics specifications of the kind given in Figs. **??** and **??**. So far, the technical difference with typical GSOS examples [**?**] is the presence of a sorting discipline. A more significant difference, which we will now highlight, is that sorted 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 (**??**)-(**??**) 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.

$p_k ::=$  (5)

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 (**??**) 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. **??** and **??**, $p_k \xrightarrow[o]{i} p_{k'}$ 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 (**??**). 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

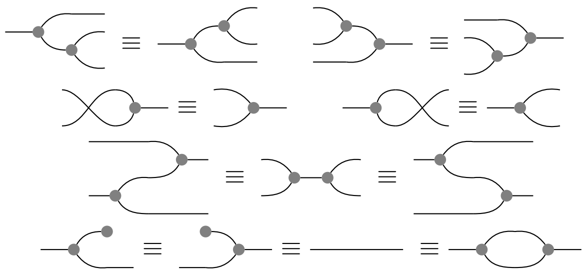


Figure 4: Axioms of special Frobenius bimonoids

categories (SMCs):

$$(f\,;g)\,;h \;\equiv\; f\,;(g\,;h) \qquad (f\,;id_m) \;\equiv\; f \qquad (id_n;f) \;\equiv\; f \tag{6}$$
$$(f\oplus g)\oplus h \;\equiv\; f\oplus(g\oplus h) \qquad (\epsilon\oplus f) \;\equiv\; f \quad (f\oplus\epsilon) \;\equiv\; f \tag{7}$$
$$(f\,;g)\oplus(h\,;i) \;\equiv\; (f\oplus h)\,;(g\oplus i) \tag{8}$$
$$\sigma_{1,1}\,;\sigma_{1,1} \;\equiv\; id_2 \qquad (\sigma_{1,n};(f\oplus id_1)) \equiv (id_1\oplus f);\sigma_{1,m} \tag{9}$$
$$(\sigma_{n,1};(id_1\oplus g)) \equiv (g\oplus id_1);\sigma_{m,1} \tag{10}$$

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 [**?**].

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, Circ_R models linear dynamical systems [**? ? ?**] and both the black and the white structures form special Frobenius bimonoids, meaning the axioms of Fig. **??** hold, replacing the gray circles by either black or white. When R is the semiring of natural numbers, Circ_R models Petri nets [**?**] and only the black structure satisfies these equations. In § **??**, we shall see that the black Frobenius structure gives rise to the synchronisation mechanism used by Hoare in CSP [**?**], while the white Frobenius structure to that used by Milner in CCS [**?**].

3. Background: Bialgebras and GSOS Specifications

(Co)algebras for endofunctors. Given a category $\mathcal{C}$ and a functor $\mathcal{F}\colon\mathcal{C}\to\mathcal{C}$, an $\mathcal{F}$*-algebra* is a pair (X,α) for an object X and an arrow $\alpha\colon\mathcal{F}X\to X$ in $\mathcal{C}$. An $\mathcal{F}$-algebra morphism $f\colon(X,\alpha)\to(X',\alpha')$ is an arrow $f\colon X\to X'$ such that $f\circ\alpha=\alpha'\circ\mathcal{F}f$. The category of $\mathcal{F}$-algebras and their morphisms is denoted by $Alg(\mathcal{F})$.

Coalgebras are defined dually: an $\mathcal{F}$*-coalgebra* is a pair (X,β) for an object X and an arrow $\alpha\colon X\to\mathcal{F}X$ in $\mathcal{C}$; an $\mathcal{F}$-coalgebra morphism $f\colon(X,\beta)\to(X',\beta')$

is an arrow $f\colon X \to X'$ such that $\alpha' \circ f = \mathcal{F}f \circ \alpha$. The category of $\mathcal{F}$-coalgebras and their morphisms is denoted by $CoAlg(\mathcal{F})$.

When $\mathcal{C}$ is Set, coalgebras can be thought as state-based systems: an $\mathcal{F}$-*coalgebra* (X, β) consists of a set of states X and a "transition" function $\beta\colon X \to \mathcal{F}X$. The functor $\mathcal{F}$ define the type of the transitions: for instance, coalgebras for the functor $\mathcal{F}X = 2 \times X^A$ are deterministic automata (see **??**). When a final object (Ω, ω) exists in $CoAlg(\mathcal{F})$, this can be thought as a universe of all possible $\mathcal{F}$-behaviours. The unique $\mathcal{F}$-coalgebra morphism $[\![\cdot]\!]\colon (X, \beta) \to (\Omega, \omega)$ is a function mapping every state of X into its behaviour.

Monads. An endofunctor $\mathcal{T}$ forms a *monad* when it is equipped with natural transformations $\eta\colon Id \to \mathcal{T}$ and $\mu\colon \mathcal{T}\mathcal{T} \to \mathcal{T}$ such that $\mu \circ \eta T = Id_{\mathcal{T}} = \mu \circ T\eta$ and $\mu \circ T\mu = \mu \circ \mu T$. An *Eilenberg-Moore algebra for a monad* $(\mathcal{T}, \eta, \mu)$ is a $\mathcal{T}$-algebra $\alpha\colon \mathcal{T}X \to X$ such that $\alpha \circ \eta_X = id_X$ and $g \circ \mu_X = g \circ \mathcal{T}g$. Morphisms are simply morphisms of $\mathcal{T}$-algebras. The category of Eilenberg-Moore algebras for $\mathcal{T}$ and their morphisms is denoted by $EM(\mathcal{T})$. For the sake of brevity, when $\mathcal{T}$ is a monad, $\mathcal{T}$-algebra will not mean algebra for the endofunctor $\mathcal{T}$, but Eilenberg-Moore algebra for the monad $\mathcal{T}$.

A *monad morphism* from a monad $(\mathcal{T}, \eta, \mu)$ to a monad $(\mathcal{T}', \eta', \mu')$ on the same category $\mathcal{C}$ is a natural transformation $\kappa\colon \mathcal{T} \to \mathcal{T}'$ such that $\kappa \circ \eta = \eta'$ and $\kappa \circ \mu = \mu' \circ \kappa\kappa$.

A recurrent monad in our work is the powerset monad $\mathcal{P}_\kappa$ bounded by a cardinal κ. It maps a set X to the set $\mathcal{P}_\kappa X = \{U \mid U \subseteq X,\ U \text{ has cardinality at most } \kappa\}$ and a function $f\colon X \to Y$ to $\mathcal{P}_\kappa f\colon \mathcal{P}_\kappa X \to \mathcal{P}_\kappa Y$, $\mathcal{P}_\kappa f(U) = \{f(u) \mid u \in U\}$. The unit η of $\mathcal{P}_\kappa$ is given by singleton, i.e., $\eta(x) = \{x\}$ and the multiplication μ is given by union, i.e., $\mu(S) = \bigcup_{U \in S} U$ for $S \in \mathcal{P}_\kappa \mathcal{P}_\kappa X$.

For instance, with $\kappa = \omega$, $\mathcal{P}_\omega$ is the *finite* powerset monad, mapping a set to its finite subsets.

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 $\mathcal{F}^\dagger X + \mathcal{F}(\mathcal{F}^\dagger X) \xrightarrow{[id, \kappa_X]} \mathcal{F}^\dagger X$.

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.

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

We refer the reader to **??** for a fully developed example of a GSOS specification, in the context of the semantics of nondeterministic automata.

Coproduct of GSOS specifications. Suppose to have two functors $\mathcal{S}_1, \mathcal{S}_2 \colon \mathcal{C} \to \mathcal{C}$ modelling two syntaxes and a functor $\mathcal{F} \colon \mathcal{C} \to \mathcal{C}$ modelling the coalgebraic behaviour, such that there are two GSOS specifications

$$\lambda_1 \colon \mathcal{S}_1\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_1^\dagger \ \text{ and } \ \lambda_2 \colon \mathcal{S}_2\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_2^\dagger.$$

Then we can construct a GSOS specification

$$\lambda_1 \cdot \lambda_2 \colon (\mathcal{S}_1 + \mathcal{S}_2)\mathcal{F} \Rightarrow \mathcal{F}(\mathcal{S}_1 + \mathcal{S}_2)^\dagger$$

as follows

$$\begin{tikzcd}
 & & \mathcal{S}_1\mathcal{F} \arrow[r, "\lambda_1"] \arrow[d, "\gamma_1"'] & \mathcal{F}\mathcal{S}_1^\dagger \arrow[d, "\mathcal{F}\iota_1"] \\
(\mathcal{S}_1 + \mathcal{S}_2)\mathcal{F} & \cong & \mathcal{S}_1\mathcal{F} + \mathcal{S}_2\mathcal{F} \arrow[r, dashed] & \mathcal{F}(\mathcal{S}_1 + \mathcal{S}_2)^\dagger \\
 & & \mathcal{S}_2\mathcal{F} \arrow[r, "\lambda_2"'] \arrow[u, "\gamma_1"] & \mathcal{F}\mathcal{S}_2^\dagger \arrow[u, "\mathcal{F}\iota_1"']
\end{tikzcd}$$

In the above diagram, the dashed map is given by universal property of the coproduct $\mathcal{S}_1\mathcal{F} + \mathcal{S}_2\mathcal{F}$. The definitions of γ_1 and γ_2 are as follows. For $X \in \mathcal{C}$, $\gamma_1(X)$ is the unique $X + \mathcal{S}_1$-algebra map from the initial $X + \mathcal{S}_1$-algebra $\mathcal{S}_1^\dagger(X)$ to $(\mathcal{S}_1 + \mathcal{S}_2)^\dagger(X)$. Indeed, since $(\mathcal{S}_1 + \mathcal{S}_2)^\dagger(X)$ is the initial $X + \mathcal{S}_1 + \mathcal{S}_2$-algebra, it is in particular a $X + \mathcal{S}_1$-algebra by precomposition with the coproduct map $X + \mathcal{S}_1 \to X + \mathcal{S}_1 + \mathcal{S}_2$. The definition of $\gamma_2(X)$ is analogous, using the fact that $(\mathcal{S}_1 + \mathcal{S}_2)^\dagger(X)$ is also a $X + \mathcal{S}_2$-algebra.

Quotients of monads and distributive laws. Given the correspondence between finitary monads and algebraic theories [?], it natural to consider *quotients* of monads by additional equations. Following [? ? ?], 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 [? , 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.* [**?**]). *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:

$$\begin{tikzcd}
\mathcal{A}\mathcal{F}A \arrow[r, shift left, "l_{\mathcal{F}A}"] \arrow[r, shift right, "r_{\mathcal{F}A}"'] & \mathcal{T}\mathcal{F}A \arrow[r, "\lambda_A"] & \mathcal{F}\mathcal{T}A \arrow[r, "\mathcal{F}q^{\mathbb{E}}_A"] & \mathcal{F}\mathcal{T}_{/\mathbb{E}}A
\end{tikzcd}$$

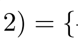
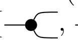
Proposition 3 (*cf.* [**?**]). *If $\lambda \colon \mathcal{T}\mathcal{F} \to \mathcal{F}\mathcal{T}$ preserves equations $\mathbb{E}$ then there exists a (unique) distributive law $\lambda_{/\mathbb{E}} \colon \mathcal{T}_{/\mathbb{E}}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{T}_{/\mathbb{E}}$ such 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 $\mathbb{N} \leftarrow \Sigma \to \mathbb{N}$ 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* § **??**. *Line* (**??**)-(**??**) *of its syntax together with the first two lines of the sorting discipline in Fig.* **??** *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 $\mathbb{N} \leftarrow \Sigma \to \mathbb{N}$. In more detail:

- Id: $\mathsf{Sig} \to \mathsf{Sig}$ is the identity functor.
- Σ: $\mathsf{Sig} \to \mathsf{Sig}$ 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 $\mathbb{N}$: $\mathsf{Sig} \to \mathsf{Sig}$ the constant functor to $\mathbb{N} \xleftarrow{id} \mathbb{N} \xrightarrow{id} \mathbb{N}$.
- $(\cdot)\,;(\cdot)$: $\mathsf{Sig}^2 \to \mathsf{Sig}$ 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, $(\cdot)\,;(\cdot)$: $\mathsf{Sig}^2 \to \mathsf{Sig}$ is associative with unit $\mathbb{N}$: $\mathsf{Sig} \to \mathsf{Sig}$.

- $(\cdot) \oplus (\cdot)$: $\mathsf{Sig}^2 \to \mathsf{Sig}$ 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 $+$: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is usual $\mathbb{N}$-addition. Again $(\cdot) \oplus (\cdot)$: $\mathsf{Sig}^2 \to \mathsf{Sig}$ 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 $\mathcal{G}$: $\mathsf{Set} \to \mathsf{Set}$ to a functor $\bar{\mathcal{G}}$: $\mathsf{Sig} \to \mathsf{Sig}$ defined by post-composition with $\mathcal{G}$, that is $\bar{\mathcal{G}}(\Sigma) = \mathcal{G} \circ \Sigma$ for all Σ: $\mathbb{N} \times \mathbb{N} \to \mathsf{Set}$. In particular, we shall often use the functor $\overline{\mathcal{P}_\kappa}$ obtained by post-composition with the κ-bounded powerset functor $\mathcal{P}_\kappa$: $\mathsf{Set} \to \mathsf{Set}$.[1]

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

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

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.

$$\mathcal{S}_{\mathsf{SM}} := (Id\,;\,Id) + \iota + (Id \oplus Id) + \epsilon + \sigma\colon \mathsf{Sig} \to \mathsf{Sig}. \tag{11}$$

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 $\mathit{identity}\colon \iota \to \Sigma$, $\mathit{composition}\colon \Sigma\,;\,\Sigma \to \Sigma$, $\mathit{parallel}\colon \Sigma \oplus \Sigma \to \Sigma$, $\mathit{unit}\colon \epsilon \to \Sigma$, and $\mathit{swap}\colon \sigma \to \Sigma$. This 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 (**??**)-(**??**) 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 § **??**. The functor $\mathcal{A}\colon \mathsf{Sig} \to \mathsf{Sig}$, defined below, has summands following the order (**??**)-(**??**):

$$\begin{aligned}(Id \oplus Id \oplus Id) + Id + Id + \sigma + ((Id\,;\,Id) \oplus (Id\,;\,Id))\\ +(Id\,;\,Id\,;\,Id) + Id + Id + Id^{+1} + Id^{+1}\end{aligned} \tag{12}$$

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 (**??**)) 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 (**??**) (associativity).

- the seventh summand Id in (**??**) 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 (**??**).

- an element in Σ^{+1} (last summand of (**??**)) 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 $\mathcal{S}^{\dagger}_{\mathtt{SM}}(\Sigma)$ of type $(n+1, m+1)$, thus giving the final equation in (**??**).

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

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

Example 6. *The monad $\mathcal{S}_{\mathtt{PROP}}$ takes Σ to the prop freely generated by Σ. Taking Σ as in Example* **??**, *one obtains $\mathcal{S}_{\mathtt{PROP}}(\Sigma)$ with arrows $n \to m$ string diagrams of* Circ_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' [**?**]. 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.* **??***).*

CW props with prop morphisms preserving the Frobenius bimonoid form a subcategory **CW** of **PROP**. We can now extend the prop monad of § **??** 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 $\mathcal{S}_{\mathtt{FR}} := \mathcal{S}_{\mathtt{PROP}} +$ —●< $+$ —● $+$ >●— $+$ ●—.

We now need to quotient $\mathcal{S}_{\mathtt{FR}}$ by the defining equations of special Frobenius bimonoids (Fig. **??**). 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:

$$x \xrightarrow[a\,b]{b} y \qquad x \xrightarrow[a\,a]{a} x \qquad z \xrightarrow[a]{b} z \tag{13}$$

Example 8. *In our main example, Fig. ?? defines a coalgebra $\beta\colon \Sigma \to \overline{\mathcal{P}_\kappa}(L\,;\,\Sigma\,;\,L)$ where Σ is the signature from Example ?? 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. Existence of Final Coalgebras

First, we prove the existence of the final coalgebra associated to each behaviour functor.

Proposition 9. *There exists a final coalgebra $\Omega \to \mathcal{F}(\Omega)$.*

Proof. Being isomorphic to a presheaf category, Sig is accessible. Then one may construct Ω via a so-called terminal sequence [?], provided that (i) $\mathcal{F}$ preserves monomorphisms and (ii) is accessible. Because functor composition preserve these properties, it suffices to check them componentwise on $\mathcal{F} = \overline{\mathcal{P}_\kappa}(L\,;\,Id\,;\,L)$. First, $(-\,;\,-)$ is defined by pullbacks, which preserve monomorphisms and (filtered) colimits; it follows that $(-\,;\,-)$ satisfies (i) and (ii). Next, the functor $\overline{\mathcal{P}_\kappa}$ satisfies (i) for the same reason as the arbitrary powerset functor, and it satisfies (ii) because it is a κ-bounded functor. Finally, satisfaction of (i) and (ii) is completely obvious for Id and the constant functor L. □

Note that the proof of Proposition ?? is constructive: following [?], for a fixed $(n,m) \in \mathbb{N} \times \mathbb{N}$, we can visualise the elements of Ω as κ-branching trees with edges labelled with pairs of labels $(l_1, l_2) \in L \times L$ with $|l_1| = n, |l_2| = m$. The unique $\mathcal{F}$-coalgebra map $[\![\cdot]\!]_\beta\colon \mathcal{S}_{\mathsf{PROP}}(\Sigma) \to \Omega$ sends a Σ-term t to the tree whose branching describes all the executions from t according to the transition rules defined by β.

5.2. 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 § **??** that $\mathcal{S}_{\mathtt{PROP}}$ is a quotient of $\mathcal{S}_{\mathtt{SM}}^{\dagger}$. We start by letting $\mathcal{F} = \overline{\mathcal{P}_\kappa}(L\,;\,Id\,;\,L)$ interact with the individual summands of $\mathcal{S}_{\mathtt{SM}}$ (see (**??**)), corresponding to the operations of props. This amounts to defining GSOS specifications:

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

Definitions of these maps are succinctly given via derivation rules, in Fig. **??**.

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

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

The last step is to upgrade $\lambda^{\dagger}$ to a distributive law $\lambda^{\dagger}{}_{/\mathsf{SMC}}$ over the quotient $\mathcal{S}_{\mathtt{PROP}}$ of $\mathcal{S}_{\mathtt{SM}}^{\dagger}$ by the equations (**??**)-(**??**) of SMCs. By Proposition **??**, 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.

$$\dfrac{s_1 \xrightarrow[v]{u} s_1' \qquad \dfrac{s_2 \xrightarrow[w]{v} s_2' \quad s_3 \xrightarrow[x]{w} s_3'}{s_2 \,;\, s_3 \xrightarrow[x]{v} s_2' \,;\, s_3'}}{s_1 \,;\, (s_2 \,;\, s_3) \xrightarrow[x]{u} s_1' \,;\, (s_2' \,;\, s_3')} \qquad \dfrac{\dfrac{s_1 \xrightarrow[v]{u} s_1' \quad s_2 \xrightarrow[w]{v} s_2'}{s_1 \,;\, s_2 \xrightarrow[w]{u} s_1' \,;\, s_2'} \qquad s_3 \xrightarrow[x]{w} s_3'}{(s_1 \,;\, s_2) \,;\, s_3 \xrightarrow[x]{u} (s_1' \,;\, s_2') \,;\, s_3'} \tag{14}$$

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

Remark 10. *The above distributive law is not unique: one can devise alternative distributive laws of type $\mathcal{S}_{\mathsf{PROP}}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_{\mathsf{PROP}}$. Indeed, any structure of prop defined over pairs of labels (as morphisms) would work. We sketch below how to define another distributive law $\mathcal{S}_{\mathsf{CAT}}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_{\mathsf{CAT}}$, as this is sufficient to make the point.*

First, we can always assume that we have some monoid structure $(\cdot, e)$ for any non-empty finite set of labels A, say by ordering the labels and computing modulo $|A|$. We now show how we can extend this to a monoid (and thus, a category with one object) on elements of A^*. We treat words over A as functions $w : \mathbb{N} \to A \cup \{\bot\}$ for which there exists some integer $|w|$ (called the length of the word w) such that $w(i) \neq \bot$ for $i \leq |w|$ and $w(i) = \bot$ for $i > |w|$. Then we can extend the binary operation $\cdot$ naturally to $A \cup \{\bot\}$ by $a \cdot \bot = \bot \cdot a = a$. Finally, we extend this further to an associative binary operation on words by computing $\cdot$ pointwise and truncating to the length of the first word:

$$(v \cdot w)(i) = \begin{cases} v(i) \cdot w(i) & \textit{if } i \leq |v|, \\ \bot & \textit{otherwise.} \end{cases}$$

By definition, notice that $|v \cdot w| = |v|$.

Now, let the sequential composition component of the distributive law be given by

$$\dfrac{s_1 \xrightarrow[w_2]{v_1} s_1' \quad s_2 \xrightarrow[w_2]{v_2} s_2'}{s_1 \,;\, s_2 \xrightarrow[w_2 \cdot w_1]{v_1 \cdot v_2} s_1' \,;\, s_2'}$$

*Note the inversion for $w_2 \cdot w_1$, which is needed because we require $|w_2 \cdot w_1| = |w_2|$ for this to be a distributive law. This definition guarantees that we can replicate the two derivations from (**??**), where the labels of the last transitions are the same.*

$$\dfrac{s_1 \xrightarrow[w_1]{v_1} s_1' \qquad \dfrac{s_2 \xrightarrow[w_2]{v_2} s_2' \quad s_3 \xrightarrow[w_3]{v_3} s_3'}{s_2 \,;\, s_3 \xrightarrow[w_3 \cdot w_2]{v_2 \cdot v_3} s_2' \,;\, s_3'}}{s_1 \,;\, (s_2 \,;\, s_3) \xrightarrow[w_3 \cdot w_2 \cdot w_1]{v_1 \cdot v_2 \cdot v_3} s_1' \,;\, (s_2' \,;\, s_3')} \qquad \dfrac{\dfrac{s_1 \xrightarrow[w_2]{v_1} s_1' \quad s_2 \xrightarrow[w_2]{v_2} s_2'}{s_1 \,;\, s_2 \xrightarrow[w_2 \cdot w_1]{v_1 \cdot v_2} s_1' \,;\, s_2'} \qquad s_3 \xrightarrow[w_3]{v_3} s_3'}{(s_1 \,;\, s_2) \,;\, s_3 \xrightarrow[w_3 \cdot w_2 \cdot w_1]{v_1 \cdot v_2 \cdot v_3} (s_1' \,;\, s_2') \,;\, s_3'}$$

The identity component of the distributive law is given by

$$\dfrac{}{id_n \xrightarrow[e^n]{e^n} id_n}$$

where the e^n label is the word of length n containing only e, the unit of the monoid structure on A, e.g., $id_3 \xrightarrow[eee]{eee} id_3$ for the identity on 3 wires. As for associativity above, we can prove (right) unitality:

$$\frac{s \xrightarrow[w]{v} s' \quad id_n \xrightarrow[e^n]{e^n} id_n}{s \,;\, id_n \xrightarrow[e^n \cdot w]{v \cdot e^n} s' \,;\, id_n}$$

This defines a suitable distributive law $\mathcal{S}_{CAT}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_{CAT}$ because, by definition, $v \cdot e^n = v$ and $e^n \cdot w = w$ since $|w| = n$.

We are now ready to construct the compositional semantics as a morphism into the final coalgebra. One starts with a coalgebra $\beta \colon \Sigma \to \mathcal{F}(\mathcal{S}_{\mathsf{PROP}}(\Sigma))$ that describes the behaviour of Σ-operations, assigning to each a set of transitions, as in (**??**). The difference with (**??**) is that, because $\mathcal{F}$ is applied to $\mathcal{S}_{\mathsf{PROP}}(\Sigma)$ 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 § **??**, using the distributive law $\lambda^{\dagger}{}_{/\mathsf{SM}}$ we can lift $\beta \colon \Sigma \to \mathcal{F}(\mathcal{S}_{\mathsf{PROP}}(\Sigma))$ to a $\lambda^{\dagger}{}_{/\mathsf{SM}}$-bialgebra, $\beta^{\sharp} : \mathcal{S}_{\mathsf{PROP}}(\Sigma) \to \mathcal{F}(\mathcal{S}_{\mathsf{PROP}}(\Sigma))$. Since this is a $\mathcal{F}$-coalgebra, the final $\mathcal{F}$-coalgebra Ω (the existence of which is shown in [**?**]) 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. **??**) of $\lambda^{\dagger}{}_{/\mathsf{SM}}$. Instead, $[\![c]\!]_{\beta}$ is the observable behaviour: intuitively, its transition systems modulo bisimilarity.

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

The bialgebraic semantics framework ensures that $\mathcal{S}_{\mathsf{PROP}}(\Sigma)$ and Ω are $\mathcal{S}_{\mathsf{PROP}}$-algebras, which by Proposition **??** 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 11. *Coming back to our running example, in Example **??** we showed that rules in Fig. **??** induce a coalgebra of type $\Sigma \to \mathcal{F}(\Sigma)$. Since each operation in Σ is itself a string diagram (formally, via the unit $\eta_{\Sigma} \colon \Sigma \to \mathcal{S}_{PROP}(\Sigma)$), the same rules induce a coalgebra $\beta : \Sigma \to \mathcal{F}\mathcal{S}_{PROP}(\Sigma)$, which has the type required for the above construction. The resulting coalgebra $\beta^{\sharp} \colon \mathcal{S}_{PROP}(\Sigma) \to \mathcal{F}\mathcal{S}_{PROP}(\Sigma)$ assigns to each diagram of $\mathcal{S}_{PROP}(\Sigma)$ the set of transitions specified by the combined operational semantics of Figs. **??** and **??**. The preceding discussion implies that, when e.g.* $\mathsf{R} = \mathbb{N}$*, bisimilarity for the Petri nets of [**?**] is a congruence.*

5.3. Bialgebraic Semantics for Carboni-Walters Props

In this section we shall see two different ways of extending the GSOS specification of § **??** for CW props (see § **??**). They correspond to the operational semantics of the black and white (co)monoids as given in Fig. **??**. 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.

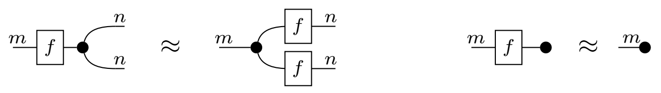
Black distributive law. The first interprets the operations of the Frobenius structure as label synchronisation: from the black node derivations on the left of Fig. **??** we get GSOS specifications given by natural transformations —●< $\Rightarrow \mathcal{F}($—●<$^{\dagger})$, —● $\Rightarrow \mathcal{F}($—●$^{\dagger})$, >●— $\Rightarrow \mathcal{F}($>●—$^{\dagger})$, 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 § **??**, 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. **??**), yielding a distributive law $\lambda_{\bullet/\mathtt{CW}}^{\dagger} : \mathcal{S}_{\mathtt{CW}}^{\dagger}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_{\mathtt{CW}}^{\dagger}$. As before, with $\lambda_{\bullet/\mathtt{CW}}^{\dagger}$ 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. **??** 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 [**?**] 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. **??**). Therefore we get a distributive law $\lambda_{\circ/\mathtt{CW}}^{\dagger} : \mathcal{S}_{\mathtt{CW}}^{\dagger}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_{\mathtt{CW}}^{\dagger}$.

5.4. Failure of Compositionality for Lawvere Theories

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 [**?**]. These can be seen as props with a *natural* comonoid structure on each object [**?**]. 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. **??**. The problem comes from the incompatibility of this structure with the nondeterminism of the behaviour functor. We explain this in more detail below.

The signature of a Lawvere theory is that of a prop with the additional comonoids. As for the comonoid part of Frobenius bimonoids (Fig.**??**), we only need to introduce this comonoid on the generating object 1 since all the others can be obtained by parallel and sequential composition of these basic morphisms and the symmetries. Let —●< $: \mathsf{Sig} \to \mathsf{Sig}$ and —● $: \mathsf{Sig} \to \mathsf{Sig}$ be the constant functors defined exactly as in § **??**. $\mathcal{S}_{\mathtt{L}}$ is is then obtained as the quotient of $\mathcal{S}_{\mathtt{PROP}} +$ —●< $+$ —● by the defining equations of cocommutative comonoids (first line of Fig. **??**), and the requirement that the comonoids be natural in the sense that every morphism $f : m \to n$ should be a comonoid homomorphism:

 (15)

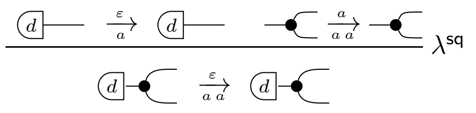
The intuitive interpretation is that every morphism can be copied and deleted using the comonoid structure.

To give a GSOS specification for $\mathcal{S}_\mathsf{L}$ it is natural to extend the specification for props with specifications for the comonoids. These are entirely analogous to the black interpretation of the comonoid part of CW props as copying and deleting operations as given in Fig. **??**. However, this specification turns out to be incompatible with the required naturality of comonoids. To see this, consider the following counter-example.

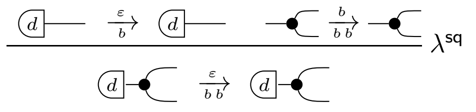
Let Σ be the signature with a single symbol $d : 0 \to 1$. We specify its behaviour as the coalgebra $\beta : \Sigma \to \mathcal{F}\mathcal{S}_\mathsf{L}\Sigma$ given by the two transitions

$$d \xrightarrow[a]{\varepsilon} d \qquad d \xrightarrow[b]{\varepsilon} d$$

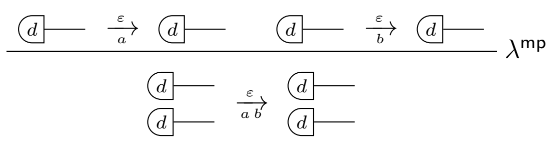
Note the use of nondeterminism here: it is this form of nondeterminism combined with the naturality requirement of (**??**) that will lead to a contradiction. The derivation rules of § **??** give us only two possible transitions for $d\,;$:



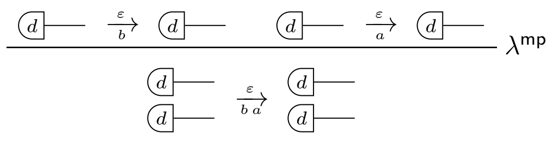
and



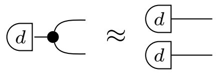
whereas $f \oplus f$ can also perform the following two transitions:



and



This is in contradiction with the leftmost equation of (**??**) which requires that



. Thus the specification would not be compositional.

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 $\{a_1, \dots, a_n\}$. This is Hoare's [?] 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 $\mathit{supp}(\sigma) = \{a_i \mid a_i \neq \sigma(a_i)\}$; σ is *finitely supported* if $\mathit{supp}(\sigma)$ is finite. For each finitely supported permutation σ its *degree* is defined as the greatest $i \in \mathbb{N}$ such that $a_i \in \mathit{supp}(\sigma)$.

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 § **??**. 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:

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

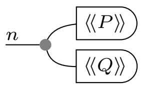
From one-dimensional to two-dimensional syntax. We use a typing discipline to guide the translation of terms to string diagrams:

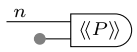
$$\frac{n \vdash P \quad n \vdash Q}{n \vdash P|Q} \qquad \frac{n+1 \vdash P}{n \vdash \nu a_{n+1}(P)} \qquad \frac{ar(\mathtt{f}) = n}{n \vdash \mathtt{f}}$$
$$\frac{n \vdash P \quad degree(\sigma) \leq n}{n \vdash P\sigma} \qquad \frac{n \vdash P}{n+1 \vdash P} \tag{16}$$

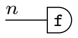
The meaning of the types is explained by the following lemma, easily proven by induction.

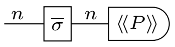
Lemma 12. *If $n \vdash P$ then $al(P) \subseteq \{a_1, \dots a_n\}$.*

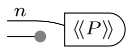
We will translate processes to the CW prop freely generated from $\Sigma = \{\mathtt{f} \colon (n, 0) \mid \mathtt{f} \in \mathcal{V} \text{ and } ar(\mathtt{f}) = n\}$; in particular a typed process $n \vdash P$ results in a string diagram of $\mathcal{S}_{\mathtt{CW}}(\Sigma)(n, 0)$. The translation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

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

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

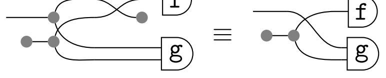
$\langle\!\langle n \vdash \mathtt{f} \rangle\!\rangle =$ 

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

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

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 13. *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 below.*



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

$$\{a_1, \dots a_{ar(\mathtt{f})}\} \subseteq \bigcup_{i\in I} al(\alpha_i) \cup \bigcup_{i\in I} al(P_i) \tag{17}$$

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

$$\frac{}{\mathtt{f} \xrightarrow{0} \mathtt{f}} \qquad \frac{\mathtt{f}{:=} \sum_{i\in I} \alpha_i.P_i}{\mathtt{f} \xrightarrow{\alpha_i} P_i} \qquad \frac{P \xrightarrow{\alpha} P'}{P\sigma \xrightarrow{\alpha\circ\sigma} P'\sigma} \tag{18}$$

Example 14. *Recall* $\mathtt{f}$ *and* $\mathtt{g}$ *from Example* **??**. *Assume the following declarations:*

$$\mathtt{f}{:=}a_1.\nu a_2(\mathtt{f}[a_2/a_1] \,|\, \mathtt{g}) \quad \textit{and} \quad \mathtt{g}{:=}a_1.\mathtt{g} + a_2.\mathtt{g}.$$

Observe that they respect (**??**). *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 [**?**]. The main difference is the level of concurrency: the classical semantics [**?**] 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

$$\frac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\beta} Q' \quad \alpha \cap al(Q) = \beta \cap al(P)}{P|Q \xrightarrow{\alpha\cup\beta} P'|Q'} \qquad \frac{P \xrightarrow{\alpha} P'}{\nu a_i(P) \xrightarrow{\alpha\setminus\{a_i\}} \nu a_i(P')} \tag{19}$$

We write $\xrightarrow{\alpha}_H$ for the transition systems generated by the rules (**??**), (**??**). By a simple inductive argument, using (**??**) 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 15. *Recall* $\mathtt{f}$ *and* $\mathtt{g}$ *from Example* **??**. *We have that* $\mathtt{f} \xrightarrow{a_1}_H \nu a_2(\mathtt{f}[a_2/a_1] \mid \mathtt{g})$. *From* $\nu a_2(\mathtt{f}[a_2/a_1] \mid \mathtt{g})$, *there are two possibilities: either* $\mathtt{f}[a_2/a_1]$ *and* $\mathtt{g}$ *synchronise on* a_2, *and in this case we have* $\nu a_2(\mathtt{f}[a_2/a_1] \mid \mathtt{g}) \xrightarrow{\emptyset}$, *or* $\mathtt{g}$ *proceeds without synchronising on* a_1, *therefore* $\nu a_2(\mathtt{f}[a_2/a_1] \mid \mathtt{g}) \xrightarrow{\{a_1\}}_H$ *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.

$$\frac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\beta} Q'}{P|Q \xrightarrow{\alpha+\beta} P'|Q'} \qquad \frac{P \xrightarrow{\alpha} P'}{\nu a_i(P) \xrightarrow{\alpha} \nu a_i(P')}\ \alpha(a_i) = 0 \tag{20}$$

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

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 § **??**, 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 (**??**) have already been studied by Milner in its work on SCCS [**?**].

Semantic correspondence. For an action $\alpha \colon \mathcal{N} \to M$ with $al(\alpha) \subseteq \{a_1, \dots a_n\}$, we write $n \vdash \alpha$ for the restriction $\{a_1, \dots, a_n\} \to M$. 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

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

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.

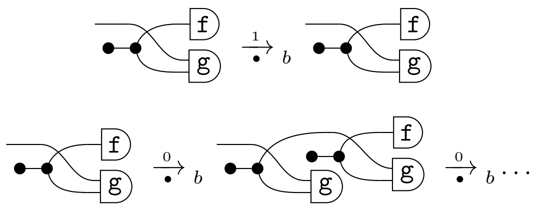
Via the distributive law (§ **??**) for the black Frobenius, we obtain the coalgebra $\beta_b^\sharp \colon \mathcal{S}_{\mathtt{CW}}(\Sigma) \to \overline{\mathcal{P}_\kappa}(L\,;\mathcal{S}_{\mathtt{CW}}(\Sigma)\,; L)$. Via the white Frobenius, we obtain $\beta_w^\sharp \colon \mathcal{S}_{\mathtt{CW}}(\Sigma) \to \overline{\mathcal{P}_\kappa}(L\,;\mathcal{S}_{\mathtt{CW}}(\Sigma)\,; L)$. 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 16. *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* $\overset{n}{\text{—}}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha}_w \overset{n}{\text{—}}\boxed{\langle\!\langle P' \rangle\!\rangle}$ *. Vice versa, if* $\overset{n}{\text{—}}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha}_w \overset{n}{\text{—}}\boxed{d}$ *then there is* $n \vdash P'$ *s.t.* $P \xrightarrow{\alpha}_M P'$ *and* $\overset{n}{\text{—}}\boxed{\langle\!\langle P' \rangle\!\rangle} = \overset{n}{\text{—}}\boxed{d}$ *.*

Example 17. *We illustrate the semantic correspondence by returning to Example* **??***. Diagrammatically, it yields the following transitions:*



6.3. Proof of Theorem **??**

We split Theorem **??** into two propositions: Proposition **??**, containing the first part of each statement and Proposition **??**, the converse correspondence. We prove each of these separately below.

Proposition 18. *Let* $n \vdash P$*.*

1. *If* $P \xrightarrow{\alpha}_H P'$ *then* $\overset{n}{\text{—}}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha}_b \overset{n}{\text{—}}\boxed{\langle\!\langle P' \rangle\!\rangle}$

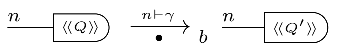
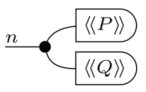
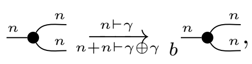
2. *If* $P \xrightarrow{\alpha}_M P'$ *then* $\overset{n}{\text{—}}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha}_w \overset{n}{\text{—}}\boxed{\langle\!\langle P' \rangle\!\rangle}$

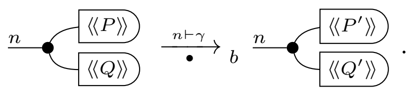
Proof. We prove 1; the proof for 2 is analogous. Hereafter, given $n \vdash \alpha$, we write $n+n \vdash \alpha \oplus \alpha$ for the function mapping a_i to $\alpha(a_i)$ if $i \leq n$ and to $\alpha(a_{i-n})$ if $i > n$.

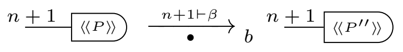
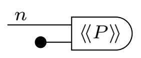
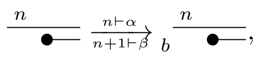
Suppose that $P \xrightarrow{\alpha}_H P'$. We prove by induction on the rules of (**??**) that

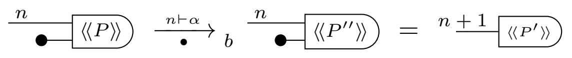
$$\overset{n}{\text{—}}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha}_b \overset{n}{\text{—}}\boxed{\langle\!\langle P' \rangle\!\rangle} .$$

- For the rule
$$\frac{n \vdash P \quad n \vdash Q}{n \vdash P|Q}$$
we observe that by the leftmost rule in (**??**), $(P|Q) \xrightarrow{\gamma}_H R$ iff $R = P'|Q'$, $P \xrightarrow{\alpha}_H P'$ and $Q \xrightarrow{\beta}_H Q'$ with $\gamma = \alpha \cup \beta$ and $\alpha \cap al(Q) = \beta \cap al(P)$. By the induction hypothesis we have that $\overset{n}{\text{—}}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha}_b \overset{n}{\text{—}}\boxed{\langle\!\langle P' \rangle\!\rangle}$ and $\overset{n}{\text{—}}\boxed{\langle\!\langle Q \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\beta}_b \overset{n}{\text{—}}\boxed{\langle\!\langle Q' \rangle\!\rangle}$. Since $\alpha \cap al(Q) = \beta \cap al(P)$ then $(\beta \setminus \alpha) \cap al(P) = \emptyset$: by Lemma **??**, we have that $\overset{n}{\text{—}}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\gamma}_b \overset{n}{\text{—}}\boxed{\langle\!\langle P' \rangle\!\rangle}$. By

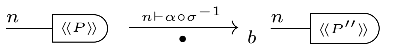
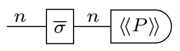
a symmetric argument, . Now, $\langle\!\langle n \vdash P|Q \rangle\!\rangle$ is equal by definition to . Since , then

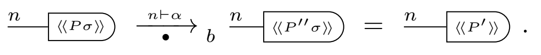


- For the rule
$$\frac{n+1 \vdash P}{n \vdash \nu a_{n+1}(P)}$$
we observe that by the rightmost rule in (**??**), $\nu a_{n+1}(P) \xrightarrow{\alpha}_H P'$ iff $P' = \nu a_{n+1}(P'')$ and $P \xrightarrow{\beta}_H P''$ with $\beta \setminus \{a_{n+1}\} = \alpha$. By the induction hypothesis . Now $\langle\!\langle n \vdash \nu a_{n+1}(P) \rangle\!\rangle$ is equal by definition to . Since , then

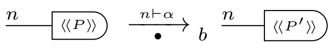


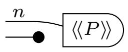
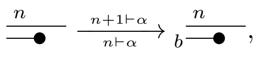
- For the rule
$$\frac{ar(\mathtt{f}) = n}{n \vdash \mathtt{f}}$$
the result is immediate by the definition of β_b and the two leftmost rules in (**??**).

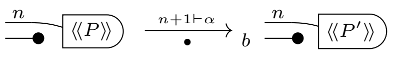
- For the rule
$$\frac{n \vdash P \quad degree(\sigma) \leq n}{n \vdash P\sigma}$$
we observe that by the rightmost rule in (**??**), $P\sigma \xrightarrow{\alpha}_H P'$ iff $P' = P''\sigma$ and $P' \xrightarrow{\alpha\circ\sigma^{-1}}_H P''$. By the induction hypothesis, we have that . Observe that $n \vdash \alpha \circ \sigma^{-1}$, since $n \vdash \alpha$ and both σ and σ^{-1} have degree n. Now, $\langle\!\langle n \vdash P\sigma \rangle\!\rangle$ is equal by definition to . Since $\overline{\sigma} \xrightarrow[n\vdash\alpha\circ\sigma^{-1}]{n\vdash\alpha}{}_b \overline{\sigma}$, then



- For the rule
$$\frac{n \vdash P}{n+1 \vdash P}$$

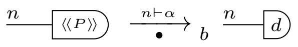
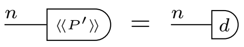
observe that for all processes P, if $P \xrightarrow{\alpha}_H P'$, then $al(\alpha) \subseteq al(P)$. Since $n \vdash P$, $al(P) \subseteq \{a_1, \dots, a_n\}$, then $al(\alpha) \subseteq \{a_1, \dots a_n\}$. So $n \vdash \alpha$. By the induction hypothesis  . Now, $\langle\langle n+1 \vdash P\rangle\rangle$ is

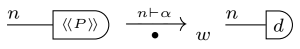
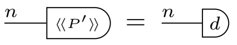
equal by definition to  . Since , then

 .

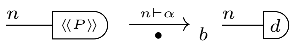
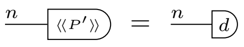
$\square$

Proposition 19. *Let $n \vdash P$, $n \vdash \alpha$ such that $al(\alpha) \subseteq al(P)$*

1. *If*  *, then there exists $n \vdash P'$ such that $P \xrightarrow{\alpha}_H P'$ and*  .

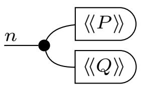
2. *If*  *then there is $n \vdash P'$ such that $P \xrightarrow{\alpha}_M P'$ and*  .

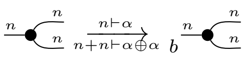
Proof. We prove 1, the proof for 2 is analogous. As before, given $n \vdash \alpha$, we write $n+n \vdash \alpha \oplus \alpha$ for the function mapping a_i to $\alpha(a_i)$ if $i \leq n$ and to $\alpha(a_{i-n})$ if $i > n$.

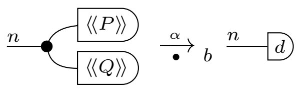
Suppose that  . We prove by induction on the rules of (**??**) that there exists $n \vdash P'$ such that $P \xrightarrow{\alpha}_H P'$ and  .

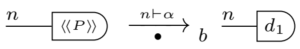
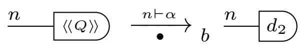
- For the rule

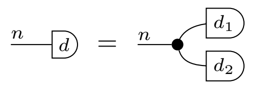
$$\frac{n \vdash P \quad n \vdash Q}{n \vdash P|Q}$$

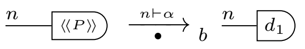
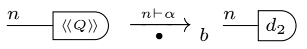
  we observe that by definition $\langle\langle n \vdash P|Q\rangle\rangle =$  . Since

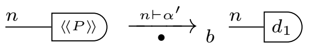
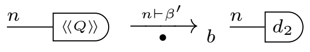


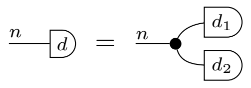
  then  iff

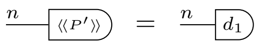
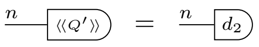
 and 

with  . Now take $\beta' = \alpha \cap al(Q)$ and $\alpha' = \alpha \cap al(P)$. We have that $\beta' \subseteq al(Q)$ and $\alpha' \subseteq al(P)$. Moreover, since $\alpha \subseteq al(P) \cup al(Q)$ by hypothesis, it holds that $\alpha' \cup \beta' = \alpha$ and that $\alpha' \cap al(Q) = \beta' \cap al(P)$.

By Lemma **??** and  and  ,

we obtain that  and 

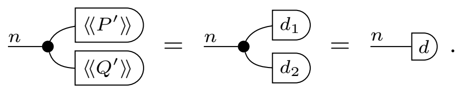
with  . We can now use induction hypothesis to get

a P' and Q' such that  ,  ,

$P \xrightarrow{\alpha'} P'$ and $Q \xrightarrow{\beta'} Q'$. By the leftmost rule in (**??**), we have that

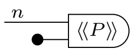
$$P|Q \xrightarrow{\alpha} P'|Q'.$$

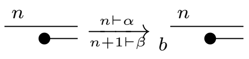
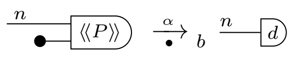
Finally, by definition of $\langle\!\langle\cdot\rangle\!\rangle$,

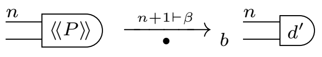
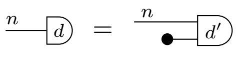
$$\langle\!\langle n \vdash P'|Q' \rangle\!\rangle =$$ 

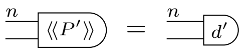
- For the rule

$$\frac{n+1 \vdash P}{n \vdash \nu a_{n+1}(P)}$$

we observe that by definition $\langle\!\langle n \vdash \nu a_{n+1} P \rangle\!\rangle =$  . Since

 with $\alpha = \beta \backslash \{a_{n+1}\}$, then 

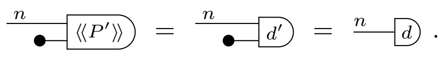
iff  with  . We can now

use the induction hypothesis to get a P' such that 

and $P \xrightarrow{\alpha} P'$. By the fact that $\alpha = \beta \setminus \{a_{n+1}\}$ and the rightmost rule in (**??**), we have that

$$\nu a_{n+1}(P) \xrightarrow{\alpha} \nu a_{n+1}(P').$$

By definition of $\langle\!\langle\cdot\rangle\!\rangle$,

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

- For the rule

$$\frac{ar(\mathtt{f}) = n}{n \vdash \mathtt{f}}$$

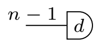
the result is immediate by the definition of β_b and the two leftmost rules in (**??**).

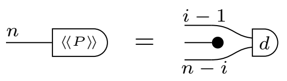
- For the rule
$$\frac{n \vdash P \quad degree(\sigma) \leq n}{n \vdash P\sigma}$$
we observe that by definition $\langle\!\langle n \vdash P\sigma \rangle\!\rangle = \overset{n}{—}\boxed{\overline{\sigma}}\overset{n}{—}\boxed{\langle\!\langle P \rangle\!\rangle}$. Since $\overline{\sigma} \xrightarrow[n\vdash\alpha\circ\sigma^{-1}]{n\vdash\alpha}{}_b\overline{\sigma}$, then $\overset{n}{—}\boxed{\overline{\sigma}}\overset{n}{—}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha}{}_b \overset{n}{—}\boxed{d}$ iff $\overset{n}{—}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha\circ\sigma^{-1}}{}_b \overset{n}{—}\boxed{d'}$ with $\overset{n}{—}\boxed{d} = \overset{n}{—}\boxed{\overline{\sigma}}\overset{n}{—}\boxed{d'}$. We can now use the induction hypothesis to get a P' such that $\overset{n}{—}\boxed{\langle\!\langle P' \rangle\!\rangle} = \overset{n}{—}\boxed{d'}$ and $P \xrightarrow{\alpha\circ\sigma^{-1}} P'$. By the rightmost rule in (**??**), we have that
$$P\sigma \xrightarrow{\alpha} P'\sigma$$
and, by the definition of $\langle\!\langle\cdot\rangle\!\rangle$,
$$\langle\!\langle n \vdash P'\sigma \rangle\!\rangle = \overset{n}{—}\boxed{\overline{\sigma}}\overset{n}{—}\boxed{\langle\!\langle P' \rangle\!\rangle} = \overset{n}{—}\boxed{\overline{\sigma}}\overset{n}{—}\boxed{d'} = \overset{n}{—}\boxed{d} .$$

- For the rule
$$\frac{n \vdash P}{n+1 \vdash P}$$
we observe that since $n \vdash P$, then $al(P) \subseteq \{a_1, \ldots, a_n\}$ and thus $a_{n+1} \notin \alpha$. By definition, $\langle\!\langle n+1 \vdash P \rangle\!\rangle = \overset{n}{—}\!\!\underset{—\bullet}{}\boxed{\langle\!\langle P \rangle\!\rangle}$. Since $\overset{n}{—}\!\!\underset{—\bullet}{} \xrightarrow[n\vdash\alpha]{n+1\vdash\alpha}{}_b \overset{n}{—}\!\!\underset{—\bullet}{}$, then $\overset{n}{—}\!\!\underset{—\bullet}{}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[n+1\vdash\alpha]{\bullet} \overset{n}{=}\boxed{d}$ iff $\overset{n}{—}\boxed{\langle\!\langle P \rangle\!\rangle} \xrightarrow[\bullet]{n\vdash\alpha}{}_b \overset{n}{—}\boxed{d'}$ with $\overset{n}{=}\boxed{d} = \overset{n}{—}\!\!\underset{—\bullet}{}\boxed{d'}$. We can now use induction hypothesis to get a P' such that $\overset{n}{—}\boxed{\langle\!\langle P' \rangle\!\rangle} = \overset{n}{—}\boxed{d'}$ and
$$P \xrightarrow{\alpha} P'.$$
By definition of $\langle\!\langle\cdot\rangle\!\rangle$,
$$\langle\!\langle n+1 \vdash P' \rangle\!\rangle = \overset{n}{—}\!\!\underset{—\bullet}{}\boxed{\langle\!\langle P' \rangle\!\rangle} = \overset{n}{—}\!\!\underset{—\bullet}{}\boxed{d'} = \overset{n}{—}\boxed{d} .$$

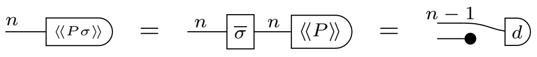
$\square$

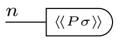
Lemma 20. *Let $n \vdash P$, $n \vdash \alpha$, $i \leq n$ and $a_i \notin al(P)$.*
$$\langle\!\langle n \vdash P \rangle\!\rangle \xrightarrow[\bullet]{n\vdash\alpha}{}_b \langle\!\langle n \vdash Q \rangle\!\rangle \quad \textit{iff} \quad \langle\!\langle n \vdash P \rangle\!\rangle \xrightarrow[\bullet]{n\vdash\alpha\cup\{a_i\}}{}_b \langle\!\langle n \vdash Q \rangle\!\rangle.$$

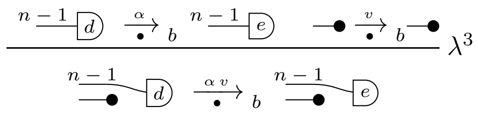
Proof. First, using Lemma **??**, we can find  such that

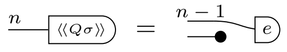
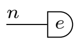
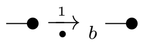
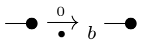


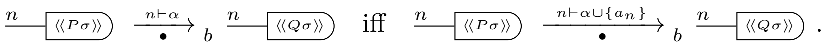
Then, we use a permutation to relocate the : let σ be the transposition $(i\ \ n)$. By definition of $\langle\!\langle - \rangle\!\rangle$ we have



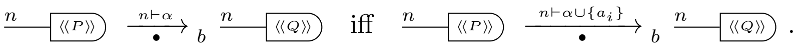
From the derivation rules for props (§ **??**), we see that any transition out of  must come from the derivation rule for $\oplus$, namely:



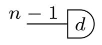
Hence, we must have  for some . Now, notice that we have  and  so that

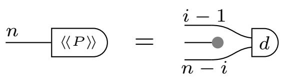


Applying the transposition $(i\ \ n)$ again we can conclude that



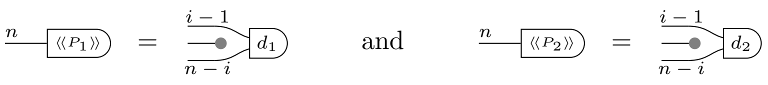
$\square$

Lemma 21. *Let $n \vdash P$, $i \leq n$ and $a_i \notin al(P)$. Then there exists*  *such that*

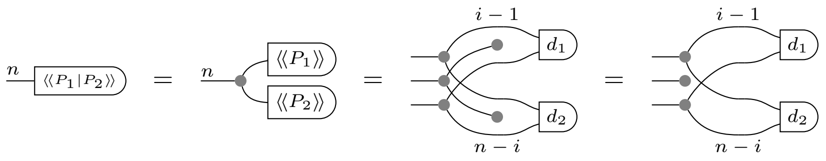


Proof. We prove this by structural induction on the rules of (**??**).

- For the rule
$$\frac{n \vdash P_1 \quad n \vdash P_2}{n \vdash P_1|P_2}$$
$al(P_1|P_2) = al(P_1) \cup al(P_2)$ and therefore $a_i \notin al(P)$ and $a_i \notin al(P_2)$. We can apply the induction hypothesis to obtain d_1 and d_2 such that



and we can deduce

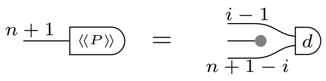


n
$\langle\!\langle P_1 \mid P_2\rangle\!\rangle$
$=$
n
$\langle\!\langle P_1\rangle\!\rangle$
$\langle\!\langle P_2\rangle\!\rangle$
$=$
$i-1$
d_1
d_2
$n-i$
$=$
$i-1$
d_1
d_2
$n-i$

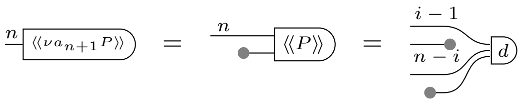
- For the rule

$$\frac{n+1 \vdash P}{n \vdash \nu a_{n+1}(P)}$$

we have $al(\nu a_{n+1}(P)) = al(P) \setminus \{a_{n+1}\}$ which implies that $a_i \notin al(P)$. We can therefore apply the induction hypothesis to find d such that



and we can deduce



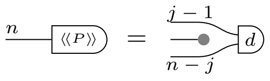
n
$\langle\!\langle \nu a_{n+1} P\rangle\!\rangle$
$=$
n
$\langle\!\langle P\rangle\!\rangle$
$=$
$i-1$
$n-i$
d

- P cannot be equal to a process variable `f` since, for $n \vdash$ `f` we have $al($`f`$) = ar($`f`$) = n$.

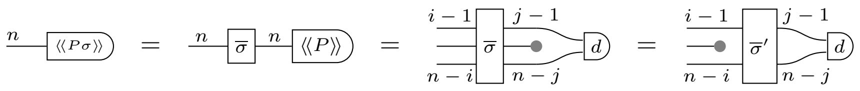
- For the rule

$$\frac{n \vdash P \quad degree(\sigma) \leq n}{n \vdash P\sigma}$$

we have $al(P\sigma) = \sigma[al(P)]$. Therefore $a_j \notin al(P)$ for $j := \sigma^{-1}(i)$. By the induction hypothesis, we can find d such that



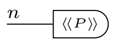
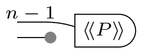
and conclude that

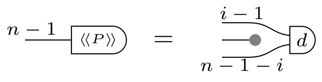


for some permutation $\sigma' : n-1 \to n-1$.

- For the rule

$$\frac{n+1 \vdash P}{n \vdash \nu a_{n+1}(P)}$$

we have  =  and, using the induction hypothesis we can find d such that



The conclusion follows immediately.

□

7. Related and Future Work

The terminology *Hoare and Milner synchronisation* is used in Synchronised Hyperedge Replacement (SHR) [**?** **?**]. 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 [**?**]. 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 **??** are also those of tile systems [**?**]. 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 [**?**].

Amongst the many other related models, it is worth mentioning bigraphs [**?**]. While also graphical, bigraphs can be nested hierarchically, a capability that we have not considered. Moreover, the behaviour functor $\mathcal{F}$ in § **??** 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 [**?**].

Appendix A. An Example of Bialgebraic Semantics: Non-Deterministic Automata

In this appendix, we illustrate bialgebraic semantics on a well-known case study, namely non-deterministic automata [**?**]. The intention is to provide in full a basic example that may serve as a roadmap for the approach to string diagrams proposed in the main text.

Deterministic automata as coalgebras. Let 2 be the set $\{0,1\}$ and A an alphabet of symbols. A deterministic automata (DA) is a pair $(S, \langle o, t\rangle)$ where S is a set of state and $\langle o, t\rangle \colon S \to 2 \times S^A$ consists of the *output function* $o \colon S \to 2$, defining whether a state $x \in S$ is accepting ($o(x) = 1$) or not ($o(x) = 0$), and the *transition function* $t \colon S \to S^A$ mapping each state $x \in S$ and each $a \in A$ the successor state $t(x)(a)$.

DAs are in one to one correspondence with coalgebras for the functor $\mathcal{F} \colon \mathsf{Set} \to \mathsf{Set}$ defined as $\mathcal{F}(X) = 2 \times X^A$. The set of all languages over the alphabet A, hereafter denoted by 2^{A^*}, carries a final coalgebra. For each deterministic automaton $(S, \langle o, t\rangle)$, there is a unique coalgebra homomorphism $[\![\cdot]\!] \colon S \to 2^{A^*}$ assigning to each state in x the language that it accepts (defined for all words $w \in A^*$ as $[\![x]\!](\epsilon) = o(x)$ and $[\![x]\!](aw) = [\![t(x)(a)]\!](w)$).

Non deterministic automata give rise to bialgebras. A non deterministic automata (NDA) is a pair $(S, \langle o, t\rangle)$ where S and o are like for deterministic automata, but the transition function t has now type $S \to \mathcal{P}_\omega S^A$, where $\mathcal{P}_\omega$ is the finite powerset functor. t maps each state $x \in S$ and each $a \in A$ to a (finite) set of possible successor states $t(x)(a)$.

Therefore NDA are coalgebras for the functor $\mathcal{F}\mathcal{P}_\omega$ where $\mathcal{F}$ is the functor for deterministic automata explained above. Traditionally the semantics of NDA is also defined in terms of languages, but the set 2^{A^*} does not carry a final coalgebra for $\mathcal{F}\mathcal{P}_\omega$. The solution is to view NDAs as DAs, by a construction that in automata theory is usually called determinisation (or powerset construction). Categorically, this amounts to transform an NDA $\beta \colon S \to 2 \times \mathcal{P}_\omega(S)^A$ into a DA $\beta^\sharp \colon \mathcal{P}_\omega(S) \to 2 \times \mathcal{P}_\omega(S)^A$, i.e. an $\mathcal{F}\mathcal{P}_\omega$-coalgebra into an $\mathcal{F}$-coalgebra. For the determinised NDA $\beta^\sharp$, there exists a unique $\mathcal{F}$-coalgebra homomorphism $[\![\cdot]\!] \colon \mathcal{P}_\omega(S) \to 2^{A^*}$. This yields language semantics $S \to 2^{A^*}$ for the original statespace S, by precomposing with the unit for the monad $\eta_S \colon S \to \mathcal{P}_\omega(S)$, as in the diagram below.

$$\begin{tikzcd}
S \arrow[r, "\eta"] \arrow[d, "\beta"'] & \mathcal{P}_\omega(S) \arrow[r, dashed, "{[\![\cdot]\!]}"] \arrow[dl, "\beta^\sharp"] & 2^{A^*} \arrow[d] \\
\mathcal{F}\mathcal{P}_\omega(S) \arrow[rr, "\mathcal{F}({[\![\cdot]\!]})"'] & & \mathcal{F}\mathcal{P}_\omega(2^{A^*})
\end{tikzcd} \tag{A.1}$$

In automata theory, there is a concrete way of constructing $\beta^\sharp$ from β. In the categorical abstraction, the general principle underpinning this construction is that defining $\beta^\sharp$ requires the introduction of a *distributive law* $\lambda\colon \mathcal{P}_\omega\mathcal{F} \Rightarrow \mathcal{F}\mathcal{P}_\omega$. For all sets X, $\lambda_X : \mathcal{P}_\omega(2 \times S^A) \to 2 \times \mathcal{P}_\omega(X)^A$ is defined for any $b_1, \ldots, b_n \in 2$ and for all $\phi_1, \ldots, \phi_n \in S^A$,

$$\lambda_X(\{\langle b_1, \phi_1\rangle, \ldots, \langle b_n, \phi_n\rangle\}) = \langle \bigsqcup_{i\in 1\ldots n} b_i, \bigsqcup_{i\in 1\ldots n} \phi_i\rangle \tag{A.2}$$

where the leftmost $\sqcup$ is the obvious join in 2 (regarded as the partial order $0 \sqsubseteq 1$), and the rightmost one is defined as the point-wise union: namely for all $a \in A$, $\bigsqcup_{i\in 1\ldots n} \phi_i(a) = \{\phi_1(a), \ldots, \phi_n(a)\}$. Observe that, in the particular case of the empty set $\emptyset$, $\alpha(\emptyset) = \langle 0, \phi_\emptyset\rangle$ where $\phi_\emptyset(a) = \emptyset$ for all $a \in A$.

Given λ, $\beta^\sharp$ is defined as $\mathcal{P}_\omega(S) \xrightarrow{\mathcal{P}_\omega\beta} \mathcal{P}_\omega\mathcal{F}\mathcal{P}_\omega(S) \xrightarrow{\lambda_{\mathcal{P}_\omega(S)}} \mathcal{F}\mathcal{P}_\omega\mathcal{P}_\omega(S) \xrightarrow{\mathcal{F}\mu} \mathcal{F}\mathcal{P}_\omega(S)$. The reader could also check that, with this definition, $\beta^\sharp$ is a λ-bialgebra, where the algebraic structure on the state space $\mathcal{P}_\omega(S)$ is given by the multiplication $\mu_S\colon \mathcal{P}_\omega\mathcal{P}_\omega(S) \to \mathcal{P}_\omega(S)$.

Algebraic presentation of $\mathcal{P}_\omega$. In concrete, determinisation is about obtaining the language semantics of NDAs. The essence of its categorical abstraction is moving from coalgebras to bialgebras, thus taking into account the *algebraic* structure of the statespace $\mathcal{P}_\omega(S)$. Indeed, it is well-known that the monad $\mathcal{P}_\omega$ is presented by the algebraic theory of join semilattice with bottom: these consist of a set S equipped with an operation $\otimes\colon S \times S \to S$ and an element $\mathbf{0}\colon 1 \to S$ such that

$$(x \otimes y) \otimes z = x \otimes (y \otimes z) \quad x \otimes y = y \otimes z \quad x \otimes x = x \quad x \otimes \mathbf{0} = x \tag{A.3}$$

for all $x, y, z \in S$. To make formal the link with $\mathcal{P}_\omega$, let us introduce the functor $\mathcal{T}_\mathcal{S}$ which maps every set S to the set $\mathcal{T}_\mathcal{S}(S)$ of terms built with $\otimes$, $\mathbf{0}$ and variables in S. This functor carries the structure of a monad, where the unit $\eta_S\colon S \to \mathcal{T}_\mathcal{S}(S)$is given by inclusion (each variable is a term) and the multiplication $\mu_S\colon \mathcal{T}_\mathcal{S}\mathcal{T}_\mathcal{S}(S) \to \mathcal{T}_\mathcal{S}(S)$ by substitution. The fact that $\mathcal{P}_\omega$ is presented by the algebraic theory of join semilattice with bottom means that $\mathcal{P}_\omega$ is (isomorphic to) the quotient of the monad $\mathcal{T}_\mathcal{S}$ by equations (**??**).

Modular construction of the distributive law. The monad $\mathcal{P}_\omega$ is associated with a rather elementary algebraic theory, which allows λ to be defined in a rather straightforward way. The monads that we use in the main text to encode categorical structures are way more involved. For this reason, it is important to modularise the task of finding a distributive law. Again, we are going to use $\lambda\colon \mathcal{P}_\omega\mathcal{F} \Rightarrow \mathcal{F}\mathcal{P}_\omega$ in (**??**) as a proof of concept, and show how its definition can be divided into different stages.

By the above discussion, $\mathcal{P}_\omega$ is the quotient of $\mathcal{T}_\mathcal{S}$ by equations (**??**), but we can decompose this even further: $\mathcal{T}_\mathcal{S}$ is the free monad (see § **??**) $(\mathcal{S}_1 + \mathcal{S}_2)^\dagger$ over the endofunctor expressing the semilattice signature: here $\mathcal{S}_1\colon S \mapsto \{\mathbf{0}\}$ stands for the element $\mathbf{0}$, and $\mathcal{S}_2\colon S \mapsto S \times S$ encodes the operation $\otimes$.

We can now divide the construction of λ in three stages:

Distributive law on the signature. The first step is to give separate distributive laws of $\mathcal{S}_1$ and $\mathcal{S}_2$ over $\mathcal{F}$. The first one is $\lambda^1\colon \mathcal{S}_1\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_1$.

Given a set S, we let $\lambda^1_S\colon \{\mathbf{0}\} \to 2 \times \{\mathbf{0}\}^A$ map $\mathbf{0}$ to the pair $\langle 0, !\rangle$ where $!$ is the unique function of type $A \to \{\mathbf{0}\}$. This distributive law can also be conveniently presented as the following pair of derivation rules. The first describes the element 0 (= $\mathbf{0}$ is not a terminating state) of the pair $\lambda^1_S\{\mathbf{0}\}$; the second describes the element $!\colon A \to \{0\}$ of the pair $\lambda^1_S\{\mathbf{0}\}$.

$$\frac{}{\mathbf{0}\not\downarrow} \qquad\qquad \frac{a \in A}{\mathbf{0} \xrightarrow{a} \mathbf{0}} \tag{A.4}$$

The second distributive law is $\lambda^1\colon \mathcal{S}_2\mathcal{F} \to \mathcal{F}\mathcal{S}_2$. Given a set S, we let $\lambda^2_S\colon (2\times S^A)\times(2\times S^A) \to 2\times(S\times S)^A$ be defined for all $\langle b_1,\phi_1\rangle, \langle b_2,\phi_2\rangle \in 2 \times S^A$ as $\lambda^2(\langle b_1,\phi_1\rangle, \langle b_2,\phi_2\rangle) = \langle b_1 \sqcup b_2, \langle \phi_1,\phi_2\rangle\rangle$. The representation in terms of derivation rules is:

$$\frac{s_1\downarrow}{s_1 \otimes s_2\downarrow} \qquad \frac{s_2\downarrow}{s_1 \otimes s_2\downarrow} \qquad \frac{s_1 \xrightarrow{a} s_1' \quad s_2 \xrightarrow{a} s_2'}{s_1 \otimes s_2 \xrightarrow{a} s_1' \otimes s_2'} \tag{A.5}$$

Distributive law on the free monad. We can now put together λ^1 and λ^2 to obtain a distributive law of type $\colon \mathcal{T}_\mathcal{S}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{T}_\mathcal{S}$. First, by post-composing λ^1 and λ^2 with the unit of the free monads $\eta\colon \mathcal{S}_i \Rightarrow \mathcal{S}_i^\dagger$ associated with endofunctors $\mathcal{S}_i$, one obtains two GSOS specifications $\overline{\lambda^1}\colon \mathcal{S}_1\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_1^\dagger$ and $\overline{\lambda^2}\colon \mathcal{S}_2\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}_2^\dagger$. As explained in Section **??**, one can take the coproduct of GSOS specification, so to obtain $\overline{\lambda}\colon \mathcal{S}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}^\dagger$ and thus, by Proposition **??** , a distributive law $\overline{\lambda}^\dagger\colon \mathcal{S}^\dagger\mathcal{F} \Rightarrow \mathcal{F}\mathcal{S}^\dagger$. Since $\mathcal{S}^\dagger = \mathcal{T}_\mathcal{S}$, this distributive law is effectively of the desired type $\mathcal{T}_\mathcal{S}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{T}_\mathcal{S}$.

Distributive law on the quotient. As $\mathcal{P}_\omega$ is a quotient of $\mathcal{T}_\mathcal{S}$ by equations (**??**), the last step is quotienting the $\overline{\lambda}^\dagger\colon \mathcal{T}_\mathcal{S}\mathcal{F} \Rightarrow \mathcal{F}\mathcal{T}_\mathcal{S}$ by such equations. There is a categorical approach, described in § **??**, which allows to turn $\overline{\lambda}^\dagger$ into a distributive law λ on the quotient of $\mathcal{T}_\mathcal{S}$, provided that $\overline{\lambda}^\dagger$ is compatible with equations (**??**). This is indeed the case for $\overline{\lambda}^\dagger$, which thus yields by Proposition **??** a distributive law $\lambda\colon \mathcal{P}_\omega\mathcal{F} \Rightarrow \mathcal{F}\mathcal{P}_\omega$ of the desired type. The definition (**??**) of λ given above can be equivalently described by the derivation rules (**??**)-(**??**), modulo the algebraic representation of $\mathcal{P}_\omega$ in terms of join semilattices.